

## About This eBook

ePUB is an open, industry-standard format for eBooks. However, support of ePUB and its many features varies across reading devices and applications. Use your device or app settings to customize the presentation to your liking. Settings that you can customize often include font, font size, single or double column, landscape or portrait mode, and figures that you can click or tap to enlarge. For additional information about the settings and features on your reading device or app, visit the device manufacturer's Web site.

Many titles include programming code or configuration examples. To optimize the presentation of these elements, view the eBook in single-column, landscape mode and adjust the font size to the smallest setting. In addition to presenting code and configurations in the reflowable text format, we have included images of the code that mimic the presentation found in the print book; therefore, where the reflowable format may compromise the presentation of the code listing, you will see a "Click here to view code image" link. Click the link to view the print-fidelity code image. To return to the previous page viewed, click the Back button on your device or app.

**IP Routing on Cisco IOS, IOS XE, and IOS XR**  
**An Essential Guide to Understanding and Implementing IP Routing**  
**Protocols**

**Brad Edgeworth, CCIE No. 31574**  
**Aaron Foss, CCIE No.18761**  
**Ramiro Garza Rios, CCIE No. 15469**

**Cisco Press**

800 East 96th Street  
Indianapolis, IN 46240



## **IP Routing on Cisco IOS, IOS XE, and IOS XR**

Brad Edgeworth, Aaron Foss, Ramiro Garza Rios

Copyright © 2015 Cisco Systems, Inc.

Cisco Press logo is a trademark of Cisco Systems, Inc.

Published by:

Cisco Press

800 East 96th Street

Indianapolis, IN 46240 USA

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the publisher, except for the inclusion of brief quotations in a review.

Printed in the United States of America

First Printing December 2014

Library of Congress Control Number: 2014957562

ISBN-13: 978-1-58714-423-3

ISBN-10: 1-58714-423-9

### **Warning and Disclaimer**

This book is designed to provide information about Cisco IOS, IOS XE, and IOS XR. Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.

The information is provided on an “as is” basis. The authors, Cisco Press, and Cisco Systems, Inc. shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the discs or programs that may accompany it.

The opinions expressed in this book belong to the author and are not necessarily those of Cisco Systems, Inc.

### **Feedback Information**

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the

unique expertise of members from the professional technical community.

Readers' feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book, or otherwise alter it to better suit your needs, you can contact us through email at [feedback@ciscopress.com](mailto:feedback@ciscopress.com). Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

### **Trademark Acknowledgments**

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Cisco Press or Cisco Systems, Inc. cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

**Publisher:** Paul Boger

**Business Operation Manager,**

**Cisco Press:** Jan Cornelssen

**Managing Editor:** Sandra Schroeder

**Project Editor:** Seth Kerney

**Technical Editors:** Richard Furr, Pete Lumbis

**Book Designer:** Gary Adair

**Composition:** Trina Wurst

**Proofreader:** Apostrophe Editing Services

**Associate Publisher:** Dave Dusthimer

**Acquisitions Editor:** Denise Lincoln

**Senior Development Editor:** Christopher Cleveland

**Copy Editor:** Keith Cline

**Editorial Assistant:** Vanessa Evans

**Cover Designer:** Mark Shirar

**Indexer:** Heather McNeill



## **Americas Headquarters**

Cisco Systems, Inc.

San Jose, CA

## **Asia Pacific Headquarters**

Cisco Systems (USA) Pte. Ltd.


Singapore

## **Europe Headquarters**

Cisco Systems International BV

Amsterdam, The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at [\*\*www.cisco.com/go/offices\*\*](http://www.cisco.com/go/offices).

 CCDE, CCENT, Cisco Eos, Cisco HealthPresence, the Cisco logo, Cisco Lumin, Cisco Nexus, Cisco StadiumVision, Cisco TelePresence, Cisco WebEx, DCE, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn and Cisco Store are service marks; and Access Registrar, Aironet, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, EtherFast, EtherSwitch, Event Center, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, iQuick Study, IronPort, the IronPort logo, LightStream, Linksys, MediaTone, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, PowerPanels, ProConnect, ScriptShare, SenderBase, SMARTnet, Spectrum Expert, StackWise, The Fastest Way to Increase Your Internet Quotient, TransPath, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0812R)

## About the Authors

**Brad Edgeworth**, CCIE No. 31574 (R&S & SP), has been with Cisco since 2011, working as a Systems Engineer and a Technical Leader. Brad is a distinguished speaker at Cisco Live, where he has presented on IOS XR. Before joining Cisco, Brad worked as a network architect and consultant for various Fortune 500 companies. Brad's other certifications include Cisco Certified Design Professional (CCDP) and Microsoft Certified Systems Engineer (MCSE). Brad has been working in the IT field for the past 18 years, with an emphasis on enterprise and service provider environments from an architectural and operational perspective. Brad holds a bachelor of arts degree in computer systems management from St. Edward's University in Austin, Texas.

**Aaron Foss**, CCIE No. 18761 (R&S & SP), is a High Touch Engineer with Cisco's Focused Technical Support (FTS) organization. He works with large service providers to troubleshoot issues relating to Multiprotocol Label Switching (MPLS), quality of service (QoS), and IP routing protocols. Aaron has more than 15 years of experience designing, deploying, and troubleshooting IP networks. He holds a bachelor of science degree in management information systems from Rochester Institute of Technology.

**Ramiro Garza Rios**, CCIE No. 15469 (R&S, SP, and Security), is a Senior Network Consulting Engineer with Cisco Advanced Services. His current role consists of planning, designing, implementing, and optimizing next-generation (NGN) service provider networks in the United States. He has been with Cisco for more than 8 years and has 14 years of networking industry experience. Before joining Cisco, Ramiro was a Network Consulting and Presales Engineer for a Cisco Gold Partner in Mexico, where he was involved in the planning, design, and implementation of many enterprise and service provider networks. He holds a bachelor of science degree in electronic engineering from the Instituto Tecnológico de Reynosa and lives with his wife and four children in Cary, North Carolina.

## About the Technical Reviewers

**Richard Furr**, CCIE No.9173 (R&S & SP), is a Technical Leader with Cisco's Technical Assistance Center (TAC). For the last 13 years, Richard has worked for Cisco TAC and High Touch Technical Support (HTTS) organizations, supporting service providers and large enterprise environments with a focus on troubleshooting routing protocols, MPLS, IP multicast and QoS.

**Pete Lumbis**, CCIE No. 28677 (R&S) and CCDE 20120003, is an expert in routing technologies including Border Gateway Protocol (BGP), MPLS, and multicast. He spent five years working in the Cisco TAC as the Routing Protocols Escalation Engineer supporting all of Cisco's customers. Most recently, Pete is focused on network design and architecture at Microsoft. Pete has been a distinguished speaker at Cisco Live on routing fast convergence and IOS routing internals.

## **Dedications**

This book is dedicated to my loving wife Tanya, who has endured and supported me through all my endeavors.

*-Brad*

I would like to dedicate this book to my supportive wife, Anne, and to my children, Ashley, Benny, and Clara, for giving up some weekend time with Dad so that I could write this book.

*-Aaron*

I would like to dedicate this book to my wonderful and beautiful wife, Mariana, and to my children Ramiro, Frinee, Felix, and Lucia for their love, patience, sacrifice, and support while writing this book.

To my parents, Ramiro Garza and Blanca Dolores Rios, for their continued support, love, encouragement, guidance, and wisdom.

And most importantly, I would like to thank God for all His blessings in my life.

*-Ramiro*

## **Acknowledgments**

### **Brad Edgeworth:**

A special thank you goes to Norm Dunn, Jocelyn Lau, Brett Bartow, and Denise Lincoln for making this book possible.

A debt of gratitude goes to my co-authors, Aaron and Ramiro. You accepted the challenge of helping me write this book. Little did you know that this project would become your second job. Some of the book's best chapters were supposed to be small, but exploded in to mini-novels to cover the topic properly. Your knowledge and dedication to this project are appreciated more than you will ever know.

To our technical editors, Richard and Pete: Thank you for finding all of our mistakes. In addition to your technical accuracy, your insight into the technologies needed by Cisco customers versus crazy ninja router tricks has kept the size of the book manageable and the content relevant.

Aaron, Ramiro, and I want to thank the Cisco Press team for their assistance and insight throughout this project. Chris Cleveland, you have been a pleasure to work with, and your attention to detail is simply amazing. It has been an educational experience for the three of us.

A special thanks to the Cisco HTTS RP and IOS XR teams, who continuously educate those about routing protocols. A special recognition to Hunter, Yigal, and Jimmy—you guys are rock stars!

Many people within Cisco have provided feedback and suggestions to make this a great book. Thanks to all who have helped in the process, especially Umair Arshad, Heather Bunch, Luc de Ghein, David Roehsler, Faraz Shamim, Craig Smith, and Mobeen Tahir.

### **Aaron Foss:**

I would like to thank my co-authors Brad and Ramiro for their amazing collaboration on this project. Brad, you have an extraordinary determination and drive that I admire greatly; and Ramiro, your technical knowledge and ability to make us laugh throughout the process of writing this book was much appreciated.

Finally, I want to acknowledge my manager, Zulfiqar Ahmed, for supporting me and encouraging me to undertake this book endeavor.

### **Ramiro Garza Rios:**

I would like to thank God for giving me the opportunity to work on this book. I would like to acknowledge my co-author Brad for the inception of this book and for being persistent until it became a reality. I would also like to acknowledge both of my co-authors, Aaron and Brad, for the great teamwork, dedication, and valuable input provided throughout the project.

## **Contents at a Glance**

Foreword

Introduction

### **Part I Network Fundamentals**

Chapter 1 Introduction to the Operating Systems

Chapter 2 IPv4 Addressing

Chapter 3 How a Router Works

### **Part II Routing Protocols**

Chapter 4 Static Routing

Chapter 5 EIGRP

Chapter 6 OSPF

Chapter 7 Advanced OSPF

Chapter 8 IS-IS

Chapter 9 Advanced IS-IS

Chapter 10 Border Gateway Protocol (BGP)

### **Part III Advanced Routing Techniques**

Chapter 11 Route Maps and Route Policy

Chapter 12 Advanced Route Manipulation

Chapter 13 Route Redistribution

### **Part IV Advanced BGP**

Chapter 14 Advanced BGP

Chapter 15 BGP Best Path Selection

### **Part V Multicast**

Chapter 16 IPv4 Multicast Routing



[Chapter 17 Advanced IPv4 Multicast Routing](#)

**[Part VI IPv6](#)**

[Chapter 18 IPv6 Addressing](#)

[Chapter 19 IPv6 Routing](#)

[Chapter 20 IPv6 Multicast Routing](#)

**[Part VII High Availability](#)**

[Chapter 21 High Availability Online](#)

**[Appendixes](#)**

[Appendix A Decimal to Hex to Binary Values Online](#)

[Appendix B BGP Attributes Online](#)

# **Contents**

## **Part I Network Fundamentals**

### **Chapter 1 Introduction to the Operating Systems**

IOS, IOS XE, and IOS XR Software Architecture

IOS

*Kernel and OS Scheduling*

*Memory Management*

*Software Packaging*

IOS XE

*Kernel and OS Scheduling*

*Memory Management*

IOS XR

*Kernel and OS Scheduling*

*Memory Management*

*Software Packaging*

Debugging

CLI and Configuration

IOS

*User Mode*

*Privileged Mode*

*Global Configuration Mode*

*Configuration Archiving*

*Configuration Replace*

IOS XR

[Viewing Changes in the SysDB](#)

[Commit Label](#)

[Commit Replace](#)

[Failed Commits](#)

[Configuration Rollback](#)

[Commit Confirmed](#)

[Multiple Commit Options](#)

[Loading Files for Changes](#)

[Hierarchical Configuration](#)

[PWD](#)

[Root](#)

[Summary](#)

[References in This Chapter](#)

## **[Chapter 2 IPv4 Addressing](#)**

[IP Fundamentals](#)

[Understanding Binary](#)

[Address Classes](#)

[Subnet Masks and Subnetting](#)

[Subnet Mask Purpose](#)

[Calculating Usable IP Addresses](#)

[Network Prefix Notation](#)

[Subnetting](#)

[Subnet Field](#)

[Subnet Math](#)

[Subnet Design](#)

Classless Interdomain Routing

Classful Versus Classless Routing

*Classful Routing*

*Classless Routing*

Variable-Length Subnet Masks

Summarization

Private IP Addressing

Special IP Addresses

IPv4 Address Configuration

Wildcard Subnet Masks

Summary

References in This Chapter

## **Chapter 3 How a Router Works**

IP Routing

Distance Vector Algorithms

Enhanced Distance Vector Algorithm

Link-State Algorithms

Path Vector Algorithm

Routing Table

*Prefix Length*

*Administrative Distance*

*Metrics*

Virtual Routing and Forwarding

IP Packet Switching

Process Switching

Cisco Express Forwarding

Software CEF

Hardware CEF

Planes of Operation

References in This Chapter

## **Part II Routing Protocols**

### **Chapter 4 Static Routing**

Connected Networks

Secondary Connected Networks

Static Routing Fundamentals

Point-to-Point Interfaces

Broadcast Interfaces

Default Route

Floating Static Routing

Recursive Lookup

Multihop Routing

*Single Recursive Lookup*

*Multiple Recursive Lookups*

Problems with Static Route Recursion

Null Interface

Static VRF Routes

References in This Chapter

### **Chapter 5 EIGRP**

EIGRP Fundamentals

EIGRP Neighbors

Inter-Router Communication

Forming EIGRP Neighbors

Classic EIGRP Autonomous System Configuration

IOS network Statement

IOS XR

Passive Interfaces

Sample Topology and Configuration

Confirmation of Interfaces

Verification of EIGRP Neighbor Adjacencies

Display of Installed EIGRP Routes

Router ID

EIGRP Terminology

Topology Table

Path Metric Calculation

Custom K Values

Interface Delay Settings

Load Balancing

EIGRP Wide Metrics

Failure Detection and Timers

Convergence

Stuck in Active

Stub

Design Considerations with EIGRP Stubs

Summarization

Interface-Specific Summarization

Summarization Metrics

Advertising a Default Route

Automatic Summarization

Authentication

Enabling Authentication on the interface

Key Chain Configuration

WAN Considerations

*IP Bandwidth Percent*

Split Horizon

Next-Hop Self

EIGRP Named Configuration

Address Family Instance Configuration

Address Family Interface Configuration

Address Family Topology Configuration

Summary

References in This Chapter

## **Chapter 6 OSPF**

OSPF Fundamentals

Inter-Router Communication

OSPF Hello Packets

Router ID

Neighbors

Forming OSPF Neighbor Adjacencies

Basic OSPF Configuration

*IOS network Statement*

IOS Interface Specific

IOS XR

Passive Interfaces

Sample Topology and Configuration

Confirmation of Interfaces

Verification of OSPF Neighbor Adjacencies

Verification of OSPF Routes

Designated Router and Backup Designated Router

Designated Router Elections

DR and BDR Placement

Failure Detection

Hello Timer

Dead Interval Timer

Verifying OSPF Timers

OSPF Fast Packet Hellos

OSPF Network Types

Broadcast

Non-Broadcast

Point-to-Point Networks

Point-to-Multipoint Networks

Loopback Networks

Review of OSPF Network Types

OSPF Adjacency with Different OSPF Network Types

Link Costs

Authentication



IOS Support for OSPF Authentication

IOS XR Support for OSPF Authentication

Summary

References in This Chapter

## **Chapter 7 Advanced OSPF**

Areas

Area ID

OSPF Route Types

*External OSPF Routes*

Link-State Announcements

LSA Age and Flooding

LSA Types

*LSA Type 1: Router Link*

*LSA Type 2: Network Link*

*LSA Type 3: Summary Link*

*LSA Type 5: External Routes*

*LSA Type 4: ASBR Summary*

*LSA Type 7: NSSA External Summary*

*LSA Type Summary*

OSPF Path Selection

Intra-Area Routes

Interarea Routes

External Route Selection

E1 and N1 External Routes

E2 and N2 External Routes

Equal Cost Multi-Path

Summarization of Routes

Interarea Summarization

External Summarization

Default Route

OSPF Stubby Areas

Stub Areas

Totally Stubby Areas

Not-So-Stubby Areas

Totally NSSA Areas

Virtual Links

Discontiguous Network

Multi-Area Adjacency

Prefix Suppression

Summary

References in This Chapter

## **Chapter 8 IS-IS**

IS-IS Fundamentals

Areas

OSI Addressing

*Inter-Domain Part*

*Domain Specific Part*

*NET Addressing*

Inter-Router Communication

IS Protocol Header

TLVs

IS PDU Addressing

Hello Packets

Link-State Packets

*LSP Lifetime*

*LSP ID*

*LSP Sequence*

*Attribute Fields*

*LSP Packet and TLVs*

IS-IS Neighbor

*Ethernet*

*Point-to-Point*

Basic IS-IS Configuration

IOS

IOS XR

Sample Topology and Configuration

Confirmation of IS-IS Interfaces

Verification of IS-IS Neighbor Adjacencies

Verification of IS-IS Routes

Designated Intermediate System

DIS Elections

DIS Placement

Point-to-Point Adjacency on Broadcast Media

Link State Packet Database

Viewing the LSPDB

Non-Pseudonode LSPs

Pseudonode LSPs

Building the Topology

Viewing the Topology

SPF Calculations

Passive Interfaces

Removal of Hello Padding

Failure Detection

Hello Timer

Hello Multiplier and Holding Timer

Authentication

IS-IS Hello Authentication

IS-IS LSP Authentication

Summary

References in This Chapter

## **Chapter 9 Advanced IS-IS**

Advanced IS-IS Routing

Route Leaking

Backbone Continuity

Loop Prevention

Router-Specific IS-IS Levels

Interface Specific IS-IS Levels

Path Selection

Equal Cost Multi-Path

Interface Metrics

Overload Bit

Summarization

Default Routes

Prefix Suppression

Summary

References in This Chapter

## **Chapter 10 Border Gateway Protocol (BGP)**

BGP Fundamentals

Autonomous System Numbers

Path Attributes

Loop Prevention

Address Families

Inter-Router Communication

Open Messages

*Hold Time*

*BGP Identifier*

Keepalive Messages

Update Messages

Notification Messages

BGP Sessions

BGP Neighbor States

Idle State

Connect State

Active State

OpenSent State

OpenConfirm State

Established State

Basic BGP Configuration

IOS

IOS XR

Verification of BGP Sessions

Prefix Advertisement

Receiving and Viewing Routes

iBGP

iBGP Full-Mesh Requirement

Peering via Loopback Addresses

eBGP

eBGP and iBGP Topologies

Next-Hop Manipulation

iBGP Scalability

Route Reflectors

*Loop Prevention in Route Reflectors*

*Out-of-Band Route Reflectors*

Confederations

Failure Detection

Security

eBGP Multihop

TTL Security

Summary

References in This Chapter

## **Part III Advanced Routing Techniques**

### **Chapter 11 Route Maps and Route Policy**

Access Control Lists

Standard ACLs

Extended ACLs

*IGP Network Selection*

*BGP Network Selection*

Prefix Matching

Prefix Lists

Prefix Sets

Regular Expressions

\_ (Underscore)

^ (Caret)

\$ (Dollar Sign)

[ ] (Brackets)

- (Hyphen)

[^] (Caret in Brackets)

() (Parentheses and | Pipe)

. (Period)

+ (Plus Sign)

? (Question Mark)

\* (Asterisk)

Looking Glass and Route Servers

AS\_Path Access List

IOS XR AS\_Path Selection Options

is-local

length

unique-length

passes-through

neighbor-is

originates-from

AS Path Set

Route Maps

Conditional Matching

*Multiple Conditional Match Conditions*

*Complex Matching*

Optional Actions

Continue

Route Map Examples

Routing Policy Language

Route Policy Structure

Match Statements

Attribute Modification

Common Route Policy Structure

Boolean Operators

*Negation*

*Conjunction*

*Disjunction*

*Order of Processing*

Comparing Prefix Sets to Prefix Lists



Parameterization

Route Policy Nesting

Original Value

Editors

RPL Examples

RPL Verification

*Redistribution RPL Verification*

*BGP RPL Verification*

References in This Chapter

## **Chapter 12 Advanced Route Manipulation**

Conditional Routing of Packets

Policy-Based Routing Configuration

Access-List-Based Forwarding Configuration

Local PBR

Administrative Distance

Modifying EIGRP AD

Modifying OSPF AD

Modifying IS-IS AD

Modifying BGP AD

Route Filtering and Manipulation

EIGRP Filtering by Prefix

EIGRP Filtering by Hop Count

EIGRP Offset Lists

OSPF Filtering (Local)

OSPF Filtering (Area)

IS-IS Filtering (Local)

BGP Filtering

Clearing BGP Connections

Summary

References in This Chapter

## **Chapter 13 Route Redistribution**

Redistribution Basics

Redistribution Is Not Transitive

Sequential Protocol Redistribution

Routes Must Exist in the RIB

Metrics

Protocol-Specific Configuration

Source-Specific Behaviors

*Connected Networks*

*IS-IS*

*BGP*

Destination-Specific Behaviors

*EIGRP*

*OSPF*

*IS-IS*

*BGP*

Challenges with Redistribution

Route Feedback

Suboptimal Routing

Invalid Routing Tables

Routing Loops

Methods to Avoid Routing Loops

*Prefix Filtering*

*Tagging*

*Increase Seed Metrics*

*Administrative Distance*

*Summarization on Redistributing Router*

Solutions to Redistribution Challenges

Summary

References in This Chapter

## **Part IV Advanced BGP**

### **Chapter 14 Advanced BGP**

BGP Communities

Enabling BGP Community Support

Well-Known Communities

*Internet*

*No\_Export*

*No\_Advertise*

*No\_Export\_SubConfed*

Conditionally Matching BGP Communities

*Community Set*

*Inline*

*Setting Private BGP Communities*

Route Summarization

Aggregate Address

Flexible Route Suppression

*Selective Prefix Suppression*

*Leaking Suppressed Routes*

Atomic Aggregate

Route Aggregation with AS\_SET

Route Aggregation with Selective Advertisement of AS\_Set

Default Route Advertisement

Default Route Advertisement Per Neighbor

Conditional Route Advertisement

Outbound Route Filtering

Backdoor Networks

Maximum Autonomous System

Maximum Prefix

Remove Private Autonomous System

Allow Autonomous System

Local Autonomous System

Configuration Scalability

IOS Peer Groups

IOS Peer Templates

IOS XR Configuration Templates

Summary

References in This Chapter

## **Chapter 15 BGP Best Path Selection**

BGP Best Path Overview

Weight

Local Preference

Locally Originated via Network or Aggregate Advertisement

Accumulated Interior Gateway Protocol

Shortest AS\_Path

Origin Type

Multi-Exit Discriminator

*Missing MED behavior*

*Always Compare Med*

*BGP Deterministic MED*

eBGP over iBGP

Lowest IGP Metric

Prefer the Oldest EBGP Path

Router ID

Minimum Cluster List Length

Lowest Neighbor Address

BGP ECMP

eBGP and iBGP Multipath

eiBGP Multipath

R1

R2

XR3

XR4

XR5

AS\_Path Relax

Suboptimal Routing with Route Reflectors

Additional Route Reflector

Shadow Route Reflector

Shadow Session Route Reflector

BGP Add-Path

Summary

Further Reading

## **Part V Multicast**

### **Chapter 16 IPv4 Multicast Routing**

Multicast Fundamentals

Multicast Addressing

Layer 2 Multicast Addresses

Internet Group Management Protocol

IGMP Snooping

IGMPv2

IGMPv3

Multicast Distribution Trees

Source Trees

Shared Trees

Protocol Independent Multicast

PIM Dense Mode

PIM Sparse Mode

*PIM Shared and Source Path Trees*

*Shared Tree Join*

*Source Registration*

*PIM SPT Switchover*

*Designated Routers*

Rendezvous Points

Static RP

Auto-RP

*Candidate RPs*

*RP Mapping Agents*

PIM Bootstrap Router

*Candidate RPs*

Reverse Path Forwarding

PIM Forwarder

Basic Multicast Configuration

Configure Rendezvous Points

*Static RP*

*Auto-RP*

*BSR*

Multicast Verification

Bidirectional PIM

Bidir-PIM Designated Forwarder

Summary

References in This Chapter

## **Chapter 17 Advanced IPv4 Multicast Routing**

Interdomain Multicast Routing

Multiprotocol BGP

Multicast Source Discovery Protocol

*MSDP Source Active Message Types*

*SA Messages*

*Keepalive Messages*

*MSDP Peers*

*MSDP Verification*

*MSDP Stub Networks*

Rendezvous Point Redundancy

Auto-RP with Multiple RPs

*Auto-RP Group Filtering*

BSR with Multiple RPs

*BSR Group Filtering*

*BSR RP Hash Algorithm*

Static RP with Multiple RPs

Anycast RP

Source Specific Multicast

SSM Mapping

*DNS SSM Mapping*

*Static SSM Mapping*

Multicast Security

Auto-RP Scoping

Multicast Boundaries

*Administratively Scoped Boundaries*

*Auto-RP Multicast Boundaries*

*BSR Multicast Boundaries*

Auto-RP Cisco-RP-Announce Message Filtering

PIM-SM Source Registration Filtering



[PIM-SM Accept RP](#)

[PIM Neighbor Control](#)

[PIM Register Rate Limit](#)

[Multicast Traffic Engineering](#)

[RPF Rules](#)

[Static Mroutes](#)

[MBGP](#)

[Static IGMP Joins](#)

[Multicast Troubleshooting](#)

[Mtrace](#)

[Summary](#)

[References in This Chapter](#)

## **[Part VI IPv6](#)**

### **[Chapter 18 IPv6 Addressing](#)**

[IPv6 Address Structure](#)

[Text Representation Address Abbreviation](#)

[IPv6 Hexadecimal to Binary Conversion](#)

[IPv6 Address Types](#)

[Unicast](#)

[Global Unicast](#)

[Unique Local Unicast](#)

[Link-Local Unicast](#)

[Anycast](#)

[Multicast](#)

[Special IPv6 Addresses](#)

Neighbor Discovery Protocol

Router, Prefix, and Parameter Discovery.

Redirect

IPv6 Stateless Address Autoconfiguration

*Extended Unique Identifier*

*SLAAC Router Configuration*

*RA Options for DNS*

Stateless DHCPv6

*IOS Stateless DHCPv6 Configuration*

*IOS XR Stateless DHCPv6 Configuration*

*Stateless DHCPv6 Verification*

Stateful DHCPv6, Relay Agent, and Relay Proxy

*IOS Relay Agent Configuration*

*IOS Relay Agent Verification*

*IOS XR Proxy Agent Configuration*

*IOS XR Proxy Agent Verification*

*IOS Stateful DHCPv6 Server Configuration*

*IOS XR Stateful DHCPv6 Server Configuration*

*Stateful DHCPv6 Server Verification*

IPv6 Address Resolution and Neighbor Unreachability Detection

Duplicate Address Detection

Summary

References in This Chapter

## **Chapter 19 IPv6 Routing**

Static Routing

[Static Route Configuration](#)

[Static Route Reference Chart for IPv6](#)

[EIGRPv6](#)

[EIGRPv6 Inter-Router Communication](#)

[EIGRPv6 Configuration](#)

[\*IOS EIGRPv6 Autonomous System Configuration \(Classic\)\*](#)

[\*IOS EIGRPv6 Hierarchical Configuration \(Named Mode\)\*](#)

[\*IOS XR EIGRPv6 Configuration\*](#)

[\*EIGRPv6 Verification\*](#)

[Summarization](#)

[Default Route](#)

[Route Filtering](#)

[EIGRP Configuration Command Reference Chart for IPv6](#)

[OSPFv3](#)

[OSPFv3 Inter-Router Communication](#)

[OSPFv3 Link-State Advertisement](#)

[OSPFv3 LSA Flooding Scope](#)

[OSPFv3 Configuration](#)

[\*IOS OSPFv3 Configuration\*](#)

[\*IOS XR OSPFv3 Configuration\*](#)

[\*OSPFv3 Verification\*](#)

[OSPFv3 Authentication](#)

[OSPFv3 Multiple Instances](#)

[OSPFv3 Configuration Command Reference Chart for IPv6](#)

[Integrated IS-IS for IPv6](#)

[IS-IS Inter-Router Communication](#)

[IS-IS Type-Length-Value](#)

[IS-IS Topology Modes](#)

[IS-IS Configuration](#)

[\*IOS Base Configuration\*](#)

[\*IOS XR Base Configuration\*](#)

[\*IOS Topology Mode Configuration\*](#)

[\*IOS XR Topology Mode Configuration\*](#)

[\*Verification\*](#)

[IS-IS Configuration Reference Chart for IPv6](#)

[Multiprotocol BGP for IPv6](#)

[Inter-Router Communication](#)

[BGP Configuration](#)

[\*IOS Base Configuration\*](#)

[\*IOS XR Base Configuration\*](#)

[BGP Verification](#)

[IPv6 over IPv4 BGP Sessions](#)

[BGP Configuration Command Reference Chart for IPv6](#)

[IPv6 Route Redistribution](#)

[Summary](#)

[References in This Chapter](#)

## **[Chapter 20 IPv6 Multicast Routing](#)**

[IPv6 Multicast Routing Overview](#)

[IPv6 Multicast Address Mapping into MAC Address](#)

[Enabling Multicast Routing](#)

[Multicast Listener Discovery](#)

[Protocol Independent Multicast](#)

[PIM Sparse Mode](#)

[\*Static RP\*](#)

[\*Bootstrap Router\*](#)

[\*Embedded RP\*](#)

[IPv6 Multicast Verification Commands](#)

[Reverse Path Forwarding](#)

[Multicast Boundary Scope](#)

[PIM Source Specific Multicast](#)

[Summary](#)

[References in This Chapter](#)

[\*\*Index\*\*](#)

[\*\*Part VII High Availability\*\*](#)

[\*\*Chapter 21 High Availability Online\*\*](#)

[\*\*Appendixes\*\*](#)

[\*\*Appendix A Decimal to Hex to Binary Values Online\*\*](#)

[\*\*Appendix B BGP Attributes Online\*\*](#)

## Icons Used in This Book



IOS Router



IOS XR Router



Layer 2 Switch



Optical Transport



Optical Cross Connect



Radio Tower



Printer



Workstation



Server



Regional Office



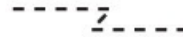
LAN Segment



Ethernet



Serial



Switched Circuit



Routing Domain

## Command Syntax Conventions

The conventions used to present command syntax in this book are the same conventions used in Cisco's Command Reference. The Command Reference describes these conventions as follows:

- Boldface indicates commands and keywords that are entered literally as shown. In actual configuration examples and output (not general command syntax), boldface indicates commands that are manually input by the user (such as a show command).
- Italics indicate arguments for which you supply actual values.
- Vertical bars (|) separate alternative, mutually exclusive elements.
- Square brackets [ ] indicate optional elements.
- Braces { } indicate a required choice.
- Braces within brackets [{ }] indicate a required choice within an optional element.

### Note

This book covers multiple operating systems, and a differentiation of icons and router names indicate the appropriate operating system that is being referenced. IOS and IOS XE use router names like R1 and R2 and are referenced by the IOS router icon. IOS XR routers will use router names like XR1 and XR2 and are referenced by the IOS XR router icon.

## Foreword

Service providers and even large, well-established enterprises, while they continue to sweat some legacy networking assets, they also realize the operational efficiencies gained by converging these disparate assets onto a common IP infrastructure. Furthermore, they generally understand the benefits of being able to offer new and innovative services with quicker time-to-market deployment with one unified converged IP backbone. Many service providers and enterprises have built out new IP backbones and are already realizing benefits of converged networking, but many have not, plus most have not realized the full potential of capability and revenue generation they can provide. This is where the need and demand for highly skilled IP network engineers becomes critical to the evolution of these IP network infrastructures, and where learning products such as Cisco career certifications and this Cisco Press resource shines in value.

This Cisco Press book is an excellent self-study resource to help aid candidates in preparing to pass exams associated with the CCNA Service Provider, CCNP Service Provider, and CCIE Service Provider career certifications. Exams associated with these Cisco certifications cover technology areas such as routing protocols (Enhanced Interior Gateway Routing Protocol [EIGRP], Open Shortest Path First [OSPF] Protocol, Intermediate System-to-Intermediate System [IS-IS] Protocol, and Border Gateway Protocol [BGP]), multicast, IPv6, and high availability. This book serves as a valuable aid in preparation in these areas. Furthermore, the book covers these topics across multiple Cisco operating system implementations, such as Cisco IOS and IOS XR, which are also covered within the noted Cisco career certifications. This resource can also aid in prepping candidates pursuing CCNA-CCNP-CCIE Routing and Switching certifications. Lastly, this book is valuable in general for learners looking to simply increase their technical understanding about how to configure routing protocols, multicast, IPv6, and high availability.

We hope and expect you'll find this book to be a valuable and frequently referenced technical aid, and a unique reference book for your personal library.

Norm Dunn

Senior Product Manager, [Learning@Cisco](mailto:Learning@Cisco)

Global Product Management, Service Provider Portfolio

Cisco Systems, Inc.



## Introduction

Within Cisco's Focused Technical Support (FTS) organization, a large number of questions about the IOS, IOS XE, and IOS XR operating systems are encountered on a daily basis. This book answers IP routing questions, in addition to covering the implementation and troubleshooting differences between the operating systems.

In alignment with the saying "a picture is worth a thousand words," multiple illustrations are included in the chapters to explain the various concepts. All protocols are presented conceptually, with applicable illustrations, configurations, and appropriate output. The scope of this book evolved to include the IOS and IOS XE operating systems so that non-IOS XR users could benefit from the explanations on the routing protocols. The book's structure explains a concept, and then provides the configuration commands and verification of the feature in small, digestible nuggets of information.

This book's content was created in alignment with [Learning@Cisco](#) to address the demand for more efficient self-study content for the Cisco Career Service Provider Certifications.

This book encompasses content spread across multiple sources and presents them in a different perspective while covering updated standards and features that are found in enterprise and service provider networks.

### WHO SHOULD READ THIS BOOK?

Network engineers, consultant, and students who want to understand the concepts and theory of EIGRP, OSPF, IS-IS, BGP, and multicast routing protocols on Cisco IOS, IOS XE, and IOS XR operating systems should read this book.

The book's content is relevant to network engineers in various stages of their career and knowledge. Every topic assumes minimal knowledge and explains the protocol from a ground-up perspective. For the advanced network engineers, relevant information on the routing protocol behavior is included. Differences in protocol behavior between IOS, IOS XE, and IOS XR are explicitly identified for each protocol.

### HOW THIS BOOK IS ORGANIZED

Although this book could be read cover to cover, it is designed to be flexible and allow you to easily move between chapters and sections of chapters to cover just the material that you need more work with. This book is organized into seven distinct sections.

Part I of the book provides a brief review of the operating systems, IP addressing, and networking fundamentals.

■ **Chapter 1, "Introduction to the Operating Systems:"** This chapter provides a high-level comparison of the network operating system architectures. An overview of the CLI configuration is provided so that users are comfortable with logging in and configuring the routers.

■ **Chapter 2, "IPv4 Addressing:"** This chapter explains the IPv4 addressing structure, the need for subnetting, and the techniques to differentiate a network address from a host address.

■ **Chapter 3, “How a Router Works:”** This chapter explains the reasons for using a routing protocol, the types of routing protocols, and the logic a router uses for forwarding packets.

Part II of the book explains static routing, EIGRP, OSPF, IS-IS, and BGP routing protocols.

■ **Chapter 4, “Static Routes:”** This chapter explains connected networks and static routes from the perspective of a router.

■ **Chapter 5, “EIGRP:”** This chapter explains the EIGRP routing protocol and how distance vector routing protocols work.

■ **Chapter 6, “OSPF:”** This chapter explains the basic fundamentals of the routing protocol, and its operational characteristics.

■ **Chapter 7, “Advanced OSPF:”** This chapter explains the reason for breaking an OSPF routing domain into multiple areas, techniques for optimization, and how to determine the best path.

■ **Chapter 8, “IS-IS:”** This chapter explains the history of the IS-IS routing protocol, along with the similarities and differences it has with OSPF.

■ **Chapter 9, “Advanced IS-IS:”** This chapter explains multilevel routing in an IS-IS domain, optimization techniques, and the path selection process.

■ **Chapter 10, “Border Gateway Protocol:”** This chapter explains the fundamental concepts of BGP sessions and route advertisement. The chapter covers the differences between external and internal peers.

Part III of the book explains the advanced routing concepts that involve routing policies and redistribution.

■ **Chapter 11, “Route Maps and Route Policy Language:”** This chapter explains prerequisite concepts such as matching networks prefixes with an access control list (ACL), prefix list or BGP advertisements with regex queries. This chapter also explains how IOS and IOS XE route maps can manipulate traffic. The chapter then discusses how IOS XR’s route policy language was designed to provide clarity and scalability.

■ **Chapter 12, “Advanced Route Manipulation:”** This chapter discusses policy-based routing, along with administrative distance manipulation, to modify route forwarding behavior. The chapter concludes by describing how to filter out specific routes from routing protocol participation.

■ **Chapter 13, “Route Redistribution:”** This chapter explains the ability to inject network prefixes learned from one routing protocol into another routing protocol. The chapter provides a thorough coverage on the rules of redistribution, problems associated with mutual redistribution, and methods for remediation.

Part IV of the book revisits BGP and describes how prefix lists, route maps, route policies, and redistribution can be used for traffic engineering.

■ **Chapter 14, “Advanced BGP:”** BGP communities, summarizations, and other router conservation techniques are explained in this chapter.

■ **Chapter 15, “BGP Best Path Selection:”** This chapter provides a through explanation of the best path selection algorithm and the ramifications that the selection has for other routers in the autonomous system. BGP route reflectors are examined, along with suboptimal routing due to path information loss. The chapter concludes with an overview of the various techniques available to optimize traffic flows when using route reflectors.

Part V of the book explains multicast traffic, the benefits of multicast, and configuration.

■ **Chapter 16, “IPv4 Multicast Routing:”** This chapter describes the benefits of multicast. Key multicast features such as Internet Group Management Protocol (IGMP), Protocol Independent Multicast (PIM), rendezvous points, multicast distribution trees are all discussed.

■ **Chapter 17, “Advanced IPv4 Multicast Routing:”** Large multicast networks require additional features to provide scalability and reachability between routing domains and autonomous systems. This chapter explains the advanced features: Multicast Source Discovery Protocol (MSDP), Source Specific Multicast (SSM), multicast boundaries, and multicast BGP.

Part VI of the book explains the IPv6 address structure, the changes to the routing protocols, and IPv6 multicast routing.

■ **Chapter 18, “IPv6 Addressing:”** This chapter describes the IPv6 address structure. The protocol stack’s neighbor discovery mechanisms are outlined, such as router advertisement messages, stateless address autoconfiguration, and duplicate address detection.

■ **Chapter 19, “IPv6 Routing:”** This chapter outlines the subtle command structure and protocol mechanics changes between the IPv4 and IPv6 routing protocols.

■ **Chapter 20, “IPv6 Multicast Routing:”** This chapter explains the fundamental differences between IPv4 and IPv6 multicast routing while emphasizing technologies like Multicast Listener Discovery (MLD), SSM, Embedded RP, and multicast boundaries.

Part VII, which can be found online at this book’s site, explains the concepts involved with improving the operational uptime of the network.

■ **Chapter 21, “High Availability:”** This chapter describes the techniques available to improve network availability and provide fast routing convergence.

## Final Words

This book is an excellent self-study resource to learn the routing protocols on Cisco IOS, IOS XE, and IOS XR operating systems. However, reading is not enough, and anyone who has obtained their CCIE will tell you that you must implement a technology to fully understand it. Our topologies are intentionally kept small to explain the routing concepts. We encourage the reader to re-create the topologies and follow along with the examples. A variety of resources are available that will allow you to practice the same concepts. Look online for the following:

- Online simulators at [Learning@Cisco](#)
- Online rack rentals
- Free demo versions of Cisco CSR 1000V (IOS XE)
- Free demo versions of Cisco IOS XRv (IOS XR)

Happy labbing!

# Part I: Network Fundamentals

# Chapter 1. Introduction to the Operating Systems

This chapter covers the following topics:

- IOS, IOS XE, and IOS XR software architecture
- Command-line interface (CLI) modes
- Configuration fundamentals

The Cisco Internetworking Operating System (IOS) is the software that powers millions of routers around the world. The IOS operating system (OS) has a long history of successfully meeting the constantly evolving challenges of multiprotocol routing. IOS XE and IOS XR are the latest evolutions of the software.

## IOS, IOS XE, AND IOS XR SOFTWARE ARCHITECTURE

This section begins with a high-level comparison of the network operating system's (NOS) software architecture. Every operating system is responsible for managing the system's hardware resources such as CPU, memory, and disks. Each NOS design has advantages and disadvantages, which affect the fundamental behavior of the operating system.

### IOS

IOS was developed in the 1980s when routers had limited memory (256 Kb) and low CPU processing power. The modularity within the IOS architecture has allowed it to grow with increased hardware capability and accommodate newer networking features requested by Cisco customers. IOS is as powerful today as when it was first created.

### Kernel and OS Scheduling

The IOS kernel follows a monolithic kernel model to accommodate for the limited system resources available at the time of its development. System processes and core functionality tightly interact, and all components have direct hardware access to conserve processing time.

IOS uses a priority *run to completion scheduler* for scheduling of processes. IOS considers each process as a single thread and assigns a priority value. High-priority processes run before lower-priority processes, but a high-priority process may not take CPU cycles from an already running low-priority process. The negative aspect to the run to completion scheduler is that an out of control process can potentially trigger a CPU hog condition and disrupt system stability.

Run to completion schedulers are CPU efficient because the system does not need to perform a context switch. A context switch is the capability for a single CPU to multitask between multiple processes. Context switching is CPU-intensive because switching back and forth between processes requires tracking and loading process thread state (context) information.

### Memory Management

IOS maps all the physical memory into one flat virtual address space, and the kernel does not perform any memory paging or swapping. Virtual address space is limited to the physical memory available but does allow *aliasing* of duplicate virtual memory contents to the same physical memory.

IOS does not implement memory protection between processes or memory pools. The advantage to this approach is that it improves system performance and minimizes the OS operational

overhead requirements. The trade-off to this approach is the increased system complexity and the potential for data corruption in one process to destabilize the entire system and potentially lead to a software-forced crash. A software-forced crash leads to a complete reload of the OS to recover.

### Software Packaging

IOS images are compiled into unique files for each platform. The feature set of the software image determines the *command-line interface* (CLI) commands and features available. The feature set name implies what technologies are available in the image. Upgrading feature sets requires using a new software image. Installing the new software image requires a software reload so that the new system image can be loaded.

Example 1-1 demonstrates how to determine the active software version and feature set of the router on IOS-based nodes with the command **show version**. Notice the highlighted system image with the feature set in bold.

### Example 1-1 IOS Version and Feature Identification

[Click here to view code image](#)

```
R1#show version
Cisco IOS Software, c7600rsp72043_rp Software (c7600rsp72043_rp-ADVIPSERVICES-M),
Version 15.2(2)S2, RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2012 by Cisco Systems, Inc.
Compiled Tue 07-Aug-12 09:17 by prod_rel_team
ROM: System Bootstrap, Version 12.2(33r)SRC3, RELEASE SOFTWARE (fc1)
BOOTLDR: Cisco IOS Software, c7600rsp72043_rp Software (c7600rsp72043_rp-ADVIPSER-
VICES-M), Version 15.2(2)S2, RELEASE SOFTWARE (fc1)
F340.10.08-7600-1 uptime is 9 minutes
Uptime for this control processor is 9 minutes
System returned to ROM by power cycle (SP by power on)
System image file is "disk0:c7600rsp72043-advipservices-mz.152-2.S2.bin"
Last reload type: Normal Reload
Last reload reason: power-on
```

#### Note

The easiest way to determine exactly what technology features are available in the image is to use the Cisco feature navigator tool on the Cisco website: <http://tools.cisco.com/ITDIT/CFN>

Cisco recently added another packaging model to address some of the impacts with upgrading feature sets. Newer IOS images offer the option of a universal image that simplifies the feature set packaging. Universal images contain a compilation of all the available features for the IOS version. Features are unlocked by activating a software license, thus minimizing the need for software reloads.

Example 1-2 demonstrates how to determine the active licenses on the router.

### Example 1-2 IOS Universal Image Version and License Identification

[Click here to view code image](#)

#### R1#show version

Cisco IOS Software, C2900 Software (C2900-UNIVERSALK9-M), Version 15.1(4)M2, RELEASE SOFTWARE (fc1)

Technical Support: <http://www.cisco.com/techsupport>

Copyright (c) 1986-2011 by Cisco Systems, Inc.

Compiled Mon 26-Sep-11 17:37 by prod\_rel\_team

ROM: System Bootstrap, Version 15.0(1r)M1, RELEASE SOFTWARE (fc1)

R1 uptime is 26 weeks, 4 days, 18 hours, 24 minutes

System returned to ROM by reload at 00:01:53 UTC Wed Jun 5 2013

System restarted at 20:01:53 EDT Tue Jun 4 2013

System image file is "flash:c2900-universalk9-mz.SPA.151-4.M2.bin"

Last reload type: Normal Reload

Last reload reason: Reload Command

! Output omitted for brevity

Technology Package License Information for Module:'c2900'

Technology	Technology-package Current	Technology-package Type	Technology-package Next reboot
ipbase	ipbasek9	Permanent	ipbasek9
security	securityk9	Permanent	securityk9
uc	uck9	Permanent	uck9
data	datak9	Permanent	datak9

Configuration register is 0x0

#### Note

Both of the software models, universal or named feature set, do not accommodate software patches for defects. A new binary file will be required along with the necessary reload.

## IOS XE

IOS XE utilizes key architectural components of IOS while improving upon the deficiencies of the IOS kernel. IOS XE's CLI and configuration is nearly identical to IOS. Network engineers familiar with IOS configuration will not notice a difference using the IOS XE operating system.

### Kernel and OS Scheduling

IOS XE adds stability through platform abstraction and the usage of a Linux kernel. The Linux kernel and drivers are the only component of the IOS XE architecture with direct access to the hardware providing additional platform stability.

The Linux kernel allows processes to execute across multiple CPUs. Application programming interfaces (API) allow applications to run on the same hardware as the router. An example application is WireShark on the Catalyst 4500 Supervisors.

All the routing protocols exist within a process called *IOSd*. Programming of the device hardware runs in a different process. Platform management can access all the necessary components of the IOS XE platform as part of the configuration management process.

### Memory Management

The Linux kernel provides virtual-memory support and supports a 64-bit architecture, allowing memory to exceed the 4-Gb limit associated with 32-bit architectures. The kernel allocates virtual memory to the IOSd process. IOSd then allocates the memory in configurable size and uses it for the various IOS functions. Memory protection mechanisms exist within the IOSd daemon to prevent memory corruption between processes.



Note

For the remainder of this book, IOS XE configuration should use the configuration listed for IOS.

## **IOS XR**

Cisco launched the IOS XR operating system in May 2004 with the CRS-1 router. At that time, multicore CPUs and gigabytes of RAM had realistic prices. Providing abundant system resources allowed IOS XR architects the opportunity to introduce additional safety mechanisms into the software design.

IOS XR's architecture is able to take advantage of the advances in hardware capabilities and provide a network operating system that provides 99.999 percent availability and scalability.

### **Kernel and OS Scheduling**

IOS XR is based on the third-party real-time operating system (RTOS) microkernel. The microkernel architecture has reduced functionality and focuses on essential services such as memory management, interprocess communication (IPC), and scheduling. System services such as device drivers, file systems, and protocols run outside the kernel as restartable processes.

The IOS XR OS allows for preemptive multitasking. High-priority processes run before lower-priority processes; however, if a high-priority process needs to run, it performs a context switch and temporarily removes the lower-priority process from the CPU. IOS XR scheduling also assigns specific time slices to processes of the same priority. This introduces a fairer scheduler and reduces the risk of a process triggering a CPU hog condition.

IOS XR microkernel architecture runs almost all system services and processes outside of the kernel. The processes run as user programs and can restart individually without requiring a full system reload. This allows for processes like authentication, authorization, and accounting (AAA), Border Gateway Protocol (BGP), and many others to be restarted without having to reload the entire router.

IOS XR performs health checks, and if a process is not running or is nonresponsive, the process is automatically restarted. To minimize data loss during the process restart, the memory state is collected by a checkpoint service. The process is restarted, and the memory state from the checkpoint service is made available to the process. This allows for minimal to zero loss of service during process recovery.

### **Memory Management**

IOS XR provides full memory protection for applications. The applications run on virtual memory instead of running directly on physical memory. A dedicated memory management unit (MMU) hardware controller manages the logical memory mappings between virtual and physical address space. Memory corruption caused by one process does not affect other processes. Virtual memory also allows memory to expand onto other storage media like flash disks and hard drives.

### **Software Packaging**

IOS XR packages utilize dynamic link libraries (DLLs). DLLs improve overall memory management on the system because DLLs allow memory sharing by all the applications that share the same library. In addition, if an application is no longer in use or not needed, the DLL is unloaded. For example, if BGP is not configured on the router, the libraries that support BGP do not need to be loaded into memory.

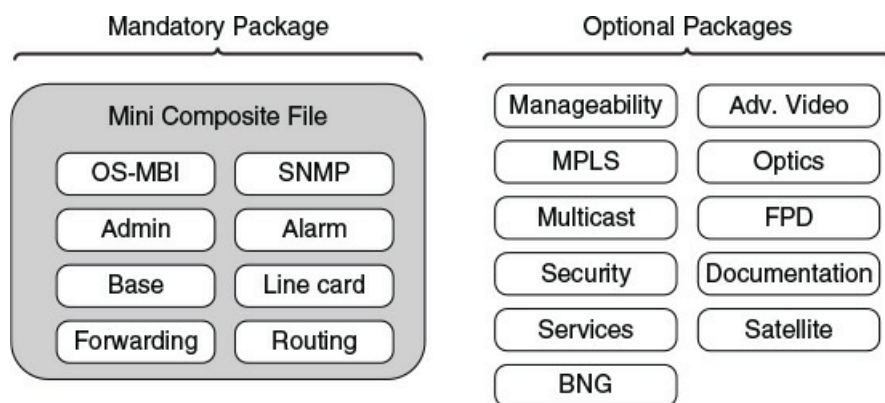
IOS XR uses software packages instead of feature sets. The modularity of the software architecture allows package installation, removal, or modification without reloading the entire

system.

Package installation envelopes (PIEs) are files that contain code for software features. The *mini* is the core composite file containing mandatory software components necessary for booting the router. The mini contains the kernel, memory management, and other core components. Additional functionality is optional and may be loaded with the appropriate PIE package. For example, multicast functionality resides in the optional multicast pie.

Software maintenance upgrades (SMUs) are PIEs that provide software patches. An SMU resolves a specific defect within a specific component of a package. SMUs provide software fixes for defects while allowing the router to stay on the same software release version.

**Figure 1-1** demonstrates IOS XR's software modularity. The mini is a mandatory package consisting of the kernel, base image, admin plane, routing, and forwarding components. Optional packages include Multiprotocol Label Switching (MPLS), multicast, Broadcast Network Gateway (BNG), and other features.



**Figure 1-1** IOS XR Modular Software Package

**Example 1-3** demonstrates how to determine the active software packages on the router with the command **show install active summary**.

### **Example 1-3** IOS XR show install active summary

[Click here to view code image](#)

```
XR1#show install active summary
Default Profile:
SDRs:
  Owner
Active Packages:
  disk0:asr9k-mini-px-4.3.2
  disk0:asr9k-px-4.3.2.CSCuj99070-1.0.0
  disk0:asr9k-px-4.3.2.CSCuj19861-1.0.0
  disk0:asr9k-fpd-px-4.3.2
  disk0:asr9k-mpls-px-4.3.2
  disk0:asr9k-mgbl-px-4.3.2
  disk0:asr9k-bng-px-4.3.2
  disk0:asr9k-doc-px-4.3.2
  disk0:asr9k-px-4.3.2.CSCuj22149-1.0.0
```

Table 1-1 provides a brief overview of the kernel, OS scheduling, memory management, and software packaging for Cisco IOS, IOS XE, and IOS XR operating systems.

	<b>Kernel and OS Scheduling</b>	<b>Memory Management</b>	<b>Software Packaging</b>
IOS	Monolithic kernel with priority run to completion scheduler	No virtual memory support, and potential for corruption through memory aliasing	Compiled into single, unique files for each platform and feature set.
IOS XE	Distributed Linux kernel that allows process execution on multiple CPUs	64-bit virtual memory support	Newer versions provide a universal image encompassing all features, and allow the activating of the features by software license.
IOS XR	Microkernel architecture that allows process execution on multiple CPUs with preemptive scheduling	Virtual memory support with dedicated MMU hardware controller to prevent memory corruption	Software is composed of DLLs that are packaged into a mini, PIEs, and SMUs. SMUs allow for patching of the IOS XR OS.

**Table 1-1** Summary of Cisco IOS, IOS XE, and IOS XR Kernel, Scheduling, Memory, and Software Packaging

## Debugging

Cisco IOS, IOS XE, and IOS XR provide debugging capabilities to identify why things are not working as anticipated. For a debug to be effective, the debugging needs to be enabled before the problem occurs. Preemptively turning on debugs in anticipation of a problem can overburden a router's CPU with unnecessary processing and present unnecessary syslog messages. Debugs may cause the router to be unstable during high load.

### Note

Impact on **debug** commands should be verified in a lab environment or under the direction of Cisco support engineers before using on production devices.

In addition to debug capability, IOS XR maintains detailed logs called *traces* that are enabled by default and do not affect the system stability or performance. Traces are a useful source of information when troubleshooting because they capture data before a problem occurs. Trace captures help support engineers identify the root cause of a problem. Traces are stored on the route processors (RPs) local storage and use circular logging, so time is of the essence when collecting trace files.

Example 1-4 shows an example of an Open Shortest Path First (OSPF) Hello trace.

## Example 1-4 IOS XR Traces

[Click here to view code image](#)

```
RP/0/0/CPU0:PE1#show ospf trace hello
Traces for OSPF 100 (Tue Jan 15 18:27:33)
Traces returned/requested/available: 2048/2048/2048
Trace buffer: hello
1 Jan 5 17:51:48.651 ospf_rcv_hello: intf Gi0/1/0/0 area.0 from 100.1.1.1
1.10.10.254
2 Jan 5 17:51:48.651 ospf_check_hello_events: intf Gi0/1/0/0 area.0 from
1.10.10.1
3 Jan 5 17:51:49.066 ospf_send_hello: area.0 intf Gi0/1/0/0 from 1.10.10.1
4 Jan 5 17:51:54.833 ospf_send_hello: area.0 intf Gi0/1/0/3 from 14.1.1.1
5 Jan 5 17:51:54.932 ospf_rcv_hello: intf Gi0/1/0/2 area.0 from 2.100.2.2
12.1.1.2
```

## CLI AND CONFIGURATION

This section provides a brief overview of the CLI and methodology for storing the router's configuration. Basic fundamentals for IOS and IOS XR have been provided so that you can apply configuration changes, as shown throughout this book.

### Note

The Cisco CLI recognizes an abbreviated command when the abbreviation contains enough characters to uniquely identify the command.

## IOS

IOS uses three CLI modes. Every CLI mode can use a variety of authentication or authorization mechanisms to ensure that users can use only the appropriate commands while in the CLI. The sections that follow detail CLI-related modes and features in more detail.

### User Mode

*User mode* is the first mode provided to a user upon logging in to a router. User mode provides some basic connectivity utilities and **show** commands. User mode prevents a user from restarting or making changes to a router.

User mode is indicated by the > prompt after the router name, as shown in [Example 1-5](#).

### Example 1-5 IOS User Mode

[Click here to view code image](#)

```

Router>
Router>?
Exec commands:
<1-99>          Session number to resume
clear           Reset functions
connect        Open a terminal connection
credential     load the credential info from file system
crypto         Encryption related commands.
disconnect     Disconnect an existing network connection
dot11         IEEE 802.11 commands
emm           Run a configured Menu System
enable        Turn on privileged commands
ethernet      Ethernet parameters
exit          Exit from the EXEC
help         Description of the interactive help system
ips          Intrusion Prevention System
lat          Open a lat connection
lig          LISP Internet Groper
login        Log in as a particular user
logout       Exit from the EXEC
modemui     Start a modem-like user interface
name-connection Name an existing network connection
ping        Send echo messages
radius      radius exec commands
release     Release a resource
renew       Renew a resource
set         Set system parameter (not config)
ssh        Open a secure shell client connection
tclquit    Quit Tool Command Language shell
telnet     Open a telnet connection
terminal   Set terminal line parameters
tn3270     Open a tn3270 connection
trm        Trend Registration Module

```

## Privileged Mode

*Privileged mode* is the second mode provided to a user. Entering privileged mode can be automatic at login or may require entering a second set of passwords or credentials. Within privileged mode, a user can restart a router, copy files to/from the router, and perform **show** commands. A user can enter privileged mode from user mode with the command **enable**, as shown in [Example 1-6](#).

### Example 1-6 IOS User Mode Going to Privileged Mode

[Click here to view code image](#)

```

Router>
Router>enable
Router#

```

## Global Configuration Mode

*Global configuration mode* is the third mode that a user can access. Quite simply, it allows a user to modify the configuration of a router. The command **configure terminal** places the user in the global configuration mode. Configuration commands may then be entered within the global configuration or under a specific configuration submode like interface or a routing protocol.

Parentheses after the router name indicate that a user is in configuration mode. Between the parentheses will indicate the context of the configuration mode, or if the context is under a submode.

Example 1-7 demonstrates moving from privileged mode to global configuration mode, and then to other various configuration submodes.

### Example 1-7 IOS Privileged Mode to Configuration Mode

[Click here to view code image](#)

```
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
! Global configuration prompt is shown below
Router(config)#
Router(config)#interface Gi0/0
! Interface configuration submode prompt is shown below
Router(config-if)#router eigrp 100
! Router configuration submode prompt is shown below
Router(config-router)#router bgp 100
! Router configuration submode prompt. Notice the prompt is the same even
! though the routing protocol is different
Router(config-router)#exit
Router(config)#
```

IOS devices store configuration files in two locations: *startup configuration* and *running configuration*. The running configuration refers to the router's active configuration and is stored in volatile memory. When the router reloads, the running configuration is lost. The startup configuration file is stored in nonvolatile memory, and the system references the startup configuration during boot to program the router's active (running) configuration.

The privileged command **copy running-config startup-config** saves system configurations across system reloads, and the privileged command **write erase** deletes the startup configuration. A write erase followed by a system reload will restore the system to factory default.

IOS processes configuration changes as a *single-stage commit*. Upon entering a command, the command is parsed for syntax, and then the change is applied instantly into the running configuration.

Example 1-8 demonstrates IOS's single-stage commit with a router name change. Notice how the router's name changes instantly.

### Example 1-8 IOS Single State Commit

[Click here to view code image](#)

```
Router(config)#hostname IOS-ROUTER
IOS-ROUTER(config)#
```

Single-stage commits may cause complications when processing configuration changes. A careful

order of operation must be followed when applying commands to prevent traffic loss or worse, being locked out of the router. Examples of router lockout include applying an access control list (ACL) to an interface before modifying the ACL to include your workstation's IP address.

#### Note

It is common to use the command **reload in time** on IOS devices so that the router reloads after the specified amount of time. Because the running configuration is not automatically saved through reloads, the system will restore to the original startup configuration upon boot. Although this is not a graceful back-out procedure, it is effective. If the configuration change is successful, however, the command **reload cancel** will cancel the timed reload.

## Configuration Archiving

IOS has a *configuration archiving* feature that maintains backup copies of the running configuration. Configuration archiving can happen at set time intervals or when the configuration is saved. Configuration archiving is set with the following steps:

**Step 1.** Enter the archive configuration submode with the command **archive**.

**Step 2.** Set the path for where the archive files will be stored using the command **path url**.

**Step 3.** Set the maximum number of copies that would like to be saved with the command **maximum number-copies**. This step is optional, and the default value is 10.

**Step 4.** Configure when the archive copy will be taken using either or both options:

**a.** Incremental time periods with the command **time-period minutes**.

**b.** Set the router to save to the archive when the running configuration is saved to startup configuration with the command **write-memory**.

Example 1-9 provides a sample configuration of IOS configuration archiving.

### Example 1-9 IOS Archival Configuration

[Click here to view code image](#)

```
R1 (config) #archive
R1 (config-archive) #path disk0:
R1 (config-archive) #time-period 1440
! Time period of 1440 (1 day) was selected to show changes, you will need to
! identify what is best for your organization.
R1 (config-archive) #write memory
R1 (config) #hostname IOS-Router1
IOS-Router1 (config) #exit
```

The command **show archive** provides the configuration archive status and filenames for each configuration. Example 1-10 provides sample output of the command.

### Example 1-10 Displaying IOS Configuration Archive

[Click here to view code image](#)

```
IOS-Router1#show archive
The maximum archive configurations allowed is 10.
There are currently 8 archive configurations saved.
The next archive file will be named disk0:CONFIG-<timestamp>-8
Archive # Name
1 disk0:CONFIG-Dec-16-21-42-33.356-0
2 disk0:CONFIG-Dec-17-21-43-33.410-1
3 disk0:CONFIG-Dec-18-21-44-33.439-2
4 disk0:CONFIG-Dec-19-21-45-33.466-3
5 disk0:CONFIG-Dec-20-21-46-30.840-4
6 disk0:CONFIG-Dec-21-21-46-33.496-5
7 disk0:CONFIG-Dec-22-21-47-33.548-6
8 disk0:CONFIG-Dec-23-21-47-40.742-7 <- Most Recent
```

Changes between the current running configuration and an archive file can be shown in a *Linux differential* format with the command **show archive config differences file-url**. [Example 1-11](#) shows an example for the changes made in [Example 1-9](#).

### Example 1-11 IOS Configuration Archive Differential

[Click here to view code image](#)

```
IOS-Router1#show archive config differences disk0:CONFIG-Dec-20-21-46-30.840-4
!Contextual Config Diffs:
+hostname R1
-hostname IOS-Router1
line con 0
-length 0
IOS-Router1#
```

### Configuration Replace

Some configuration changes in IOS need to be backed out of due to unintended circumstances. Trying to do a **copy startup-config running-config** does not work because the copy action acts as a merge. Some settings may need to be removed.

IOS provides a configuration replace feature that will replace the running configuration with a previously saved configuration. The saved configuration can come from the configuration archive or a local file containing a copy of the running configuration. The command **configure replace file-url** will initiate the process, as shown in [Example 1-12](#). Notice that the hostname on the router changed to the hostname in the identified file.

### Example 1-12 IOS Configuration Replace to Archived Configuration

[Click here to view code image](#)



```
IOS-Router1#configure replace disk0:PreSaved-Config.txt
This will apply all necessary additions and deletions
to replace the current running configuration with the
contents of the specified configuration file, which is
assumed to be a complete configuration, not a partial
configuration. Enter Y if you are sure you want to proceed. ? [no]: y
Total number of passes: 1
Rollback Done
R1#
*Dec 18 01:19:48.435: Rollback:Acquired Configuration lock.
R1#
```

#### Note

The **configure replace** process requires enough free memory to accommodate the running configuration and the source configuration file. In some rare cases, Cisco configuration commands cannot be removed from the running configuration without reloading a device. If this situation were to arise, an error message for each specific command is provided.

## IOS XR

EXEC mode is the default mode for users at login. The # after the router name indicates that the user is in EXEC mode, as shown in [Example 1-13](#).

### Example 1-13 IOS XR EXEC Mode Prompt

[Click here to view code image](#)

```
RP/0/RP0/CPU0:XR1#
```

#### Note

The concept of a user mode (>) does not exist for IOS XR. Privileges are assigned during login.

IOS XR contains an admin mode that provides access to IOS XR's administrative plane. The administrative plane controls software versions, PIES, SMUs, and the power state for line cards. Admin mode is reachable with the command **admin** and indicated by state of *admin*, as shown in [Example 1-14](#).

### Example 1-14 IOS XR Admin Mode Prompt

[Click here to view code image](#)

```
RP/0/RP0/CPU0:XR1#admin
RP/0/RP0/CPU0:XR1 (admin) #
```

#### Note

The full functionality of the admin mode is beyond the scope of this book.

All the relevant configuration for routing will be done within the global configuration of an IOS

XR node. The command **configure terminal** places the IOS XR router into global configuration mode. Configuration commands can then be entered within the global configuration or under a specific configuration submode like interface or a routing protocol.

Example 1-15 shows an example of entering the global configuration.

### Example 1-15 IOS XR Global Configuration Prompt

[Click here to view code image](#)

```
RP/0/RP0/CPU0:XR1#configure terminal
RP/0/RP0/CPU0:XR1 (config)#
```

IOS XR uses a *two-stage commit* for processing of configuration changes.

In the first stage, a *target configuration* is created where changes are entered. The target configuration contains only changes that have been entered and does not contain a full copy of the running configuration. Upon entering a command, IOS XR parses the configuration for syntax and then applies the command to the target configuration. Multiple configuration commands can be entered to the target configuration as needed.

In the second stage, IOS XR processes the target configuration into the running configuration with the **commit** command. Figure 1-2 shows IOS XR's two-stage configuration model.

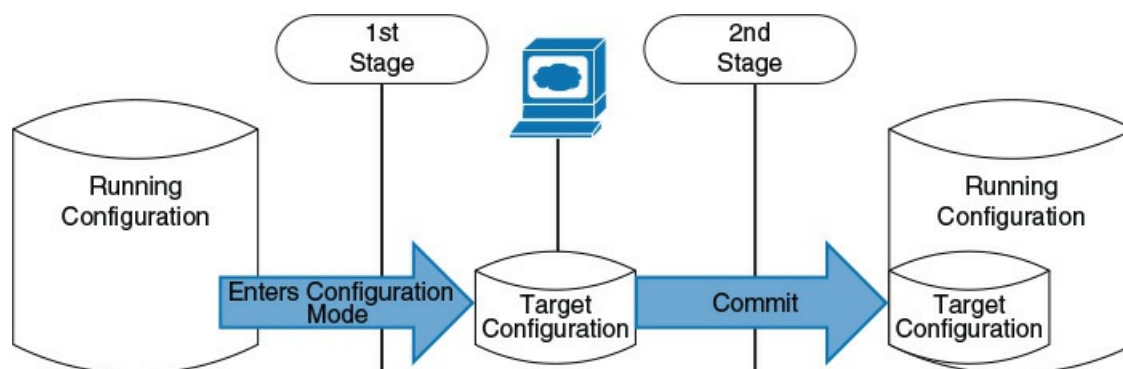


Figure 1-2 IOS XR Two-Stage Commit

The command **show running-config** displays a copy of the running configuration, as shown in Example 1-16.

### Example 1-16 IOS XR show running-config on a Fresh Install

[Click here to view code image](#)

```
RP/0/0/CPU0:ios#show running-config
Building configuration...
!! IOS XR Configuration
interface GigabitEthernet0/0/0/0
 shutdown
!
interface GigabitEthernet0/0/0/1
 shutdown
!
end
RP/0/0/CPU0:ios#
```

The command **show configuration** shows the configurations within the target configuration. [Example 1-17](#) demonstrates entering Cisco Discovery Protocol (CDP) configuration and then verifying the current target configuration.

### Example 1-17 IOS XR show configuration

[Click here to view code image](#)

```
RP/0/0/CPU0:ios#conf terminal
RP/0/0/CPU0:ios (config) #hostname XR1
RP/0/0/CPU0:ios (config) #cdp
RP/0/0/CPU0:ios (config) #int gigabitEthernet 0/0/0/0
RP/0/0/CPU0:ios (config-if) #cdp
RP/0/0/CPU0:ios (config-if) #int gigabitEthernet 0/0/0/1
RP/0/0/CPU0:ios (config-if) #cdp
RP/0/0/CPU0:ios (config-if) #show configuration
Building configuration...
!! IOS XR Configuration
hostname XR1
cdp
interface GigabitEthernet0/0/0/0
 cdp
!
interface GigabitEthernet0/0/0/1
!
End
```

#### Note

CDP helps map the physical network topology and is not enabled by default on IOS XR. [Example 1-17](#) demonstrates enabling it.

The command **show configuration merge** shows the target configuration merged with the running configuration.

In [Example 1-18](#), the target configuration from [Example 1-17](#) is combined with the running configuration from [Example 1-16](#).

### Example 1-18 IOS XR show configuration merge

[Click here to view code image](#)

```
RP/0/0/CPU0:ios(config-if)#show configuration merge
Building configuration...
!! IOS XR Configuration
!! Last configuration change at Fri Sep 20 15:42:00 2014 by xr
!
hostname XR1
cdp
interface GigabitEthernet0/0/0/0
  cdp
  shutdown
!
interface GigabitEthernet0/0/0/1
  cdp
  shutdown
!
End
```

IOS XR performs a final semantic check against the running configuration when the **commit** command is applied. Upon detecting configuration errors, the commit fails and the target configuration is not applied. If the syntax check passes, then the target configuration changes are accepted and processed against the running configuration. The target configuration is reset and ready for new input.

[Example 1-19](#) demonstrates the target configuration from [Example 1-16](#) being committed. Notice that the configuration change triggers a commit ID log message and that the router's hostname finally changes once the commit request is processed.

### Example 1-19 IOS XR commit

[Click here to view code image](#)

```
RP/0/0/CPU0:ios(config-if)#commit
RP/0/0/CPU0:Sep 21 00:26:25.360 : config[66391]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration committed by user 'JCHAMBR'. Use 'show configuration commit changes
1000000638' to view the changes.
RP/0/0/CPU0:XR1(config-if)#
```

IOS XR stores the running configuration within a database referred to as *SysDB*. As changes are committed to the system, a commit ID is associated to the configuration modification and entered into the SysDB.

### Viewing Changes in the SysDB

You can query the SysDB with the command **show configuration commit list**, which provides a complete listing of the changes in the SysDB. Changes are listed with the most recent being first, along with the commit ID (or commit label), username, method, and time stamp for the commit. The **show configuration commit list** is a helpful tool for reviewing configuration changes that have occurred on the router.

[Example 1-20](#) demonstrates the configuration commit list for an IOS XR router.

## Example 1-20 IOS XR show configuration commit list

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show configuration commit list
No. Label/ID      User      Line                Client  Time Stamp
~~~~ ~~~~~~      ~~~~      ~~~~                ~~~~~~ ~~~~~~
 1  1000000038  JCHAMBR  vty3:node0_0_CPU0  CLI     Fri Sep 13 11:06:35 2014
 2  1000000037  KJOHNS   vty3:node0_0_CPU0  CLI     Fri Sep 13 11:05:33 2014
 3  1000000036  BEDGEW   vty3:node0_0_CPU0  CLI     Fri Sep 13 11:00:41 2014
 4  1000000035  RAMIROG  vty3:node0_0_CPU0  CLI     Fri Sep 13 10:59:39 2014
 5  1000000034  JCHAMBR  vty3:node0_0_CPU0  CLI     Tue Aug 27 15:08:04 2014
 6  1000000033  KJOHNS   vty1:node0_0_CPU0  CLI     Tue Jul 16 15:32:27 2014
 7  1000000032  RAMIROG  vty3:node0_0_CPU0  CLI     Mon Jul 15 16:22:54 2014
 8  1000000031  BEDGEW   vty3:node0_0_CPU0  CLI     Mon Jul 15 16:21:14 2014
 9  1000000030  JCHAMBR  vty3:node0_0_CPU0  CLI     Mon Jul 15 16:02:07 2014
10  1000000029  KJOHNS   vty4:node0_0_CPU0  CLI     Mon Jul 8 16:06:58 2014
11  1000000028  RAMIROG  vty4:node0_0_CPU0  CLI     Mon Jul 8 16:04:55 2014
12  1000000027  BEDGEW   vty4:node0_0_CPU0  CLI     Mon Jul 8 15:55:08 2014
13  1000000026  JCHAMBR  vty4:node0_0_CPU0  CLI     Mon Jul 8 15:54:03 2014
14  1000000025  KJOHNS   vty4:node0_0_CPU0  CLI     Mon Jul 8 15:49:53 2014
15  1000000024  FOSS     vty4:node0_0_CPU0  CLI     Mon Jul 8 15:48:49 2014
```

The command **show configuration commit changes** {*commit-id* | *label*} displays the commit ID's configuration modification. [Example 1-21](#) demonstrates how to see the changes for a specific commit ID.

## Example 1-21 IOS XR show configuration commit for Specific Change

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show configuration commit changes 1000000025
Building configuration...
!! IOS XR Configuration
!
no route-policy RPL-L3-IPv4-IN-BETA
end
```

To see the complete delta of changes over multiple commits, the command **show configuration commit changes last** *number-of-commits* is used.

[Example 1-22](#) shows an example of looking at the changes within the last three commits.

## Example 1-22 IOS XR show configuration commit changes Incremental

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show configuration commit changes last 3
Building configuration...
!! IOS XR Configuration
no cdp
!
no interface Loopback0
!
no router ospf 1
end
```

## Commit Label

During the commit process, a unique label can be assigned using the command **commit label** *label-name*. Labels are a great method for providing a description for configuration changes. [Example 1-23](#) demonstrates applying a configuration commit label and then viewing the configuration commit list history.

### Example 1-23 IOS XR commit label

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#configure terminal
RP/0/0/CPU0:XR1 (config) #hostname XR1-ROUTER
RP/0/0/CPU0:XR1 (config) #commit label ROUTERNAMECHANGE
RP/0/0/CPU0:XR1-ROUTER (config) #show configuration commit list
```

SNo.	Label/ID	User	Line	Client	Time Stamp
~~~~	~~~~~	~~~~	~~~~	~~~~~	~~~~~
1	ROUTERNAME	JCHAMBR	con0_0_CPU0	CLI	Fri Sep 20 16:02:22 2014
2	1000000014	JCHAMBR	con0_0_CPU0	CLI	Fri Sep 20 15:48:35 2014
3	1000000013	JCHAMBR	con0_0_CPU0	CLI	Fri Sep 20 15:48:25 2014

#### Note

Some organizations assign a label that matches the change control number associated within an organization's IT processes.

## Commit Replace

A **commit replace** completely replaces the running configuration with the target configuration. Take extra care when using this option because performing a commit replace on an empty target configuration will, in essence, erase the router configuration instead of merging the configuration like a normal commit action.

[Example 1-24](#) demonstrates a configuration change along with simulating the running configuration merged with the target configuration.

### Example 1-24 IOS XR Change with show configuration merge

[Click here to view code image](#)

```

! Change of the Hostname
RP/0/0/CPU0:XR1#conf terminal
RP/0/0/CPU0:XR1(config)#hostname XR1-COMMITREPLACE
! Examine the target configuration
RP/0/0/CPU0:XR1(config)#show configuration
Building configuration...
!! IOS XR Configuration
hostname XR1-COMMITREPLACE
end
! Examine the target configuration merged with the running configuration
RP/0/0/CPU0:XR1(config)#show configuration merge
Building configuration...
!! IOS XR Configuration
!! Last configuration change at Fri Sep 20 16:07:55 2014 by xr
!
hostname XR1-COMMITREPLACE
cdp
interface GigabitEthernet0/0/0/0
  cdp
  shutdown
!
interface GigabitEthernet0/0/0/1
  cdp
  shutdown
!
end

```

In [Example 1-25](#), a **commit replace** was performed, and examining the running configuration shows that it does not match the simulated output in [Example 1-23](#). The target configuration replaced the running configuration. Notice how the interfaces statuses change to up because the target configuration did not include a **shutdown** command in the target configuration.

### Example 1-25 IOS XR commit replace

[Click here to view code image](#)

```

RP/0/0/CPU0:XR1(config)#commit replace
This commit will replace or remove the entire running configuration. This
operation can be service affecting.
Do you wish to proceed? [no]: y
RP/0/0/CPU0:Sep 20 16:08:57.284 : ifmgr[220]: %PKT_INFRA-LINK-3-UPDOWN :
Interface GigabitEthernet0/0/0/1, changed state to Down
RP/0/0/CPU0:XR1-COMMITREPLACE(config)#
RP/0/0/CPU0:Sep 20 16:08:57.284 : ifmgr[220]: %PKT_INFRA-LINK-3-UPDOWN :
Interface GigabitEthernet0/0/0/0, changed state to Down
RP/0/0/CPU0:Sep 20 16:08:57.314 : ifmgr[220]: %PKT_INFRA-LINK-3-UPDOWN :
Interface GigabitEthernet0/0/0/1, changed state to Up
RP/0/0/CPU0:Sep 20 16:08:57.314 : ifmgr[220]: %PKT_INFRA-LINK-3-UPDOWN :
Interface GigabitEthernet0/0/0/0, changed state to Up
! Examine the running-configuration now that the change has been committed.
RP/0/0/CPU0:XR1-COMMITREPLACE(config)#show running-config
Building configuration...
!! IOS XR Configuration
!
hostname XR1-COMMITREPLACE
end

```

## Failed Commits

If the semantic check fails during a commit process, the target configuration does not merge with the running configuration. The command **show configuration failed** displays the reason for the failed semantic check during the commit.

**Example 1-26** demonstrates an attempt to configure more than one BGP process. A router will not support two BGP processes, so the system does not apply the configuration and reports a failed to commit error message. The command **show configuration failed** provides details on which command in the target configuration failed the commit parser check.

## Example 1-26 IOS XR show configuration failed

[Click here to view code image](#)

```

RP/0/0/CPU0:XR1#conf terminal
RP/0/0/CPU0:XR1(config)#router bgp 100
RP/0/0/CPU0:XR1(config-bgp)#commit
RP/0/0/CPU0:XR1(config-bgp)#router bgp 200
RP/0/0/CPU0:XR1(config-bgp)#commit
% Failed to commit one or more configuration items during a pseudo-atomic
operation. All changes made have been reverted. Please issue 'show configuration
failed' from this session to view the errors
RP/0/0/CPU0:XR1(config-bgp)#show configuration failed
!! SEMANTIC ERRORS: This configuration was rejected by
!! the system due to semantic errors. The individual
!! errors with each failed configuration command can be
!! found below.
router bgp 200
!!% The process 'bpm' rejected the operation but returned no error
!
End

```



## Configuration Rollback

IOS XR allows one or multiple changes to be rolled back through the usage of the SysDB. Upon rolling back a configuration change, the operator can specify the number of committed changes to rollback with the command **rollback configuration last** *number-of-changes*.

### Note

Performing a rollback counts as a SysDB change and should be considered when you do a second rollback. It is recommended to look at the change list to make sure that you are selecting the correct commit ID or change count.

In [Example 1-27](#), the configuration was rolled back three changes and was verified with the **show configuration commit list** command. Notice that the client will show *Rollback* rather than *CLI*.

### Example 1-27 IOS XR Configuration of a Rollback of a Specific Number of Changes

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1-COMMITREPLACE#rollback configuration last 3
Loading Rollback Changes.
Loaded Rollback Changes in 1 sec
Committing..
10 items committed in 2 sec (4)items/sec
Updating.
Updated Commit database in 1 sec
Configuration successfully rolled back 3 commits.
RP/0/0/CPU0:XR1#show configuration commit list
Fri Sep 20 16:37:20.748 UTC
SNo. Label/ID      User      Line      Client      Time Stamp
~~~~ ~~~~~~      ~~~~      ~~~~      ~~~~~~      ~~~~~~
 1   1000000021   JCHAMBR   con_0_CPU0   Rollback    Fri Sep 20 16:37:10 2014
 2   1000000020   JCHAMBR   con_0_CPU0   CLI         Fri Sep 20 16:08:57 2014
 3   1000000019   JCHAMBR   con_0_CPU0   CLI         Fri Sep 20 16:07:55 2014
 4   ROUTERNAME   JCHAMBR   con_0_CPU0   CLI         Fri Sep 20 16:07:14 2014
```

It is also possible to rollback to a specific commit ID or label. This removes any potential error in counting the number of changes needed to roll back to. The command **rollback configuration to** *{commit-id | label}* takes the router back to the specific configuration change.

[Example 1-28](#) provides an example of rolling back to a specific commit ID.

### Example 1-28 IOS XR Rollback to a Specific Change ID

[Click here to view code image](#)

```

RP/0/0/CPU0:XR1#rollback configuration to ROUTERNAME
Loading Rollback Changes.
Loaded Rollback Changes in 1 sec
Committing.
13 items committed in 1 sec (12)items/sec
Updating.
Updated Commit database in 1 sec
Configuration successfully rolled back to 'ROUTERNAMECHANGE'.
RP/0/0/CPU0:XR1#show configuration commit list

```

SNo.	Label/ID	User	Line	Client	Time Stamp
1	1000000022	JCHAMBR	con0_0_CPU0	Rollback	Fri Sep 20 16:47:08 2014
2	1000000021	JCHAMBR	con0_0_CPU0	Rollback	Fri Sep 20 16:37:10 2014
3	1000000020	JCHAMBR	con0_0_CPU0	CLI	Fri Sep 20 16:08:57 2014
4	1000000019	JCHAMBR	con0_0_CPU0	CLI	Fri Sep 20 16:07:55 2014

### Commit Confirmed

A configuration change can sometimes lead to traffic loss. The commit confirmed feature will enable the changes but will initiate a configuration rollback at the end of the duration specified. If the change is sufficient, a second commit is required to prevent the rollback and permanently apply the changes.

The command syntax is **commit confirmed 30-65535\_seconds**.

In [Example 1-29](#), a commit confirmed is performed without a secondary commit to confirm the changes. Notice the highlighted time stamps and that the rollback initiates automatically after 30 seconds.

### Example 1-29 IOS XR commit confirmed Without Confirmation

[Click here to view code image](#)

```

RP/0/0/CPU0:XR1(config)#hostname XR1-COMMIT-CONFIRM
RP/0/0/CPU0:XR1(config)#commit confirmed 30
RP/0/0/CPU0:Sep 16 13:46:53.374 : config[66625]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration committed by user 'BEDGEW'. Use 'show configuration commit changes
1000000042' to view the changes.
RP/0/0/CPU0:XR1-COMMIT-CONFIRM(config)#
RP/0/0/CPU0:Sep 16 13:47:24.075 : cfgmgr_trial_confirm[66653]: %MGBL-CONFIG-6-DB_
COMMIT : Configuration committed by user 'BEDGEW'. Use 'show configuration commit
changes
1000000043' to view the changes.
RP/0/0/CPU0:XR1-COMMIT-CONFIRM#show configuration commit list
Mon Sep 16 13:59:44.908 EDT

```

SNo.	Label/ID	User	Line	Client	Time Stamp
1	1000000043	BEDGEW	vty3:node0_0_CPU0	Rollback	Mon Sep 16 13:47:23 2014
2	1000000042	BEDGEW	vty3:node0_0_CPU0	CLI	Mon Sep 16 13:46:53 2014

[Example 1-30](#) demonstrates a commit confirmed with a secondary commit to confirm the changes. Notice that the configuration commit list shows only one commit performed and that the time stamp for the change is the first commit.

## Example 1-30 IOS XR commit confirmed with Confirmation

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1(config)#hostname XR1-COMMIT-CONFIRM
RP/0/0/CPU0:XR1(config)#commit confirmed 30
RP/0/0/CPU0:Sep 16 13:51:47.414 : config[66850]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration
  committed by user 'BEDGEW'. Use 'show configuration commit changes 1000000044' to
  view
  the changes.
RP/0/0/CPU0:XR1-COMMIT-CONFIRM(config)#
RP/0/0/CPU0:XR1-COMMIT-CONFIRM(config)#commit
Mon Sep 16 13:51:55.705 EDT
% Confirming commit for trial session.
RP/0/0/CPU0:XR1-COMMIT-CONFIRM(config)#exit
RP/0/0/CPU0:XR1-COMMIT-CONFIRM#show configuration commit list
```

SNo.	Label/ID	User	Line	Client	Time Stamp
1	1000000044	BEDGEW	vty3:node0_0_CPU0	CLI	Mon Sep 16 13:51:47 2014
4	1000000043	BEDGEW	vty3:node0_0_CPU0	Rollback	Mon Sep 16 13:47:23 2014
5	1000000042	BEDGEW	vty3:node0_0_CPU0	CLI	Mon Sep 16 13:46:53 2014

### Multiple Commit Options

Up until this point, the **commit** command has been shown with only one optional parameter. The **commit** command allows multiple parameters to be executed on the same command at one time.

[Example 1-31](#) show the **commit** command with two parameters: **label** and **confirmed**.

## Example 1-31 IOS XR Multiple Commit Options

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1(config)#commit label MULTIPLE confirmed 30
RP/0/0/CPU0:Sep 20 16:59:44.006 : config[65704]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration committed by user 'BEDGW'. Use 'show configuration commit changes
1000000023' to view the changes.
RP/0/0/CPU0:XR-MULTIPLE(config)#commit
RP/0/0/CPU0:Sep 20 17:00:18.423 : config[65704]: %MGBL-SYS-5-CONFIG_I : Configured
from console by BEDGW% Confirming commit for trial session.
```

### Loading Files for Changes

IOS XR's two-stage commit allows for changes to be created in advance and stored as text files on the local storage. The configuration files can be loaded into the target configuration with the **load filesystem:filename** command. The change is then committed like any other change. This feature provides the opportunity to allow any change to be peer reviewed in advance of its implementation.

[Example 1-32](#) shows the loading of a configuration file.

## Example 1-32 IOS XR Loading Configuration Files

[Click here to view code image](#)

```

RP/0/0/CPU0:XR1#conf terminal
RP/0/0/CPU0:XR1 (config)#load bootflash:PRECONFIG-FILE.txt
Loading.
105 bytes parsed in 1 sec (104)bytes/sec
RP/0/0/CPU0:XR1 (config)#show configuration
Building configuration...
!! IOS XR Configuration
hostname XR1-PRECONFIG
cdp
interface GigabitEthernet0/0/0/0
  ipv4 address 10.12.1.1 255.255.255.0
!
!
route-policy PASS-ALL
  pass
end-policy
!
End

```

### Hierarchical Configuration

On IOS-based nodes, the routing protocol configuration may exist under the router process, under the interface configuration submode, or in the global process. IOS XR's configuration is 100 percent hierarchical. For example, all features and options for a routing protocol reside within the routing protocol configuration. Parameters that are set at the global process for the routing protocol are inherited by the more specific components of the protocol.

Figure 1-3 depicts a timer set to a value of 10 seconds at the global process for the OSPF routing protocol. The 10-second setting is inherited at the area level and then down to the interfaces within that area so that all interfaces will have a timer value of 10 seconds.

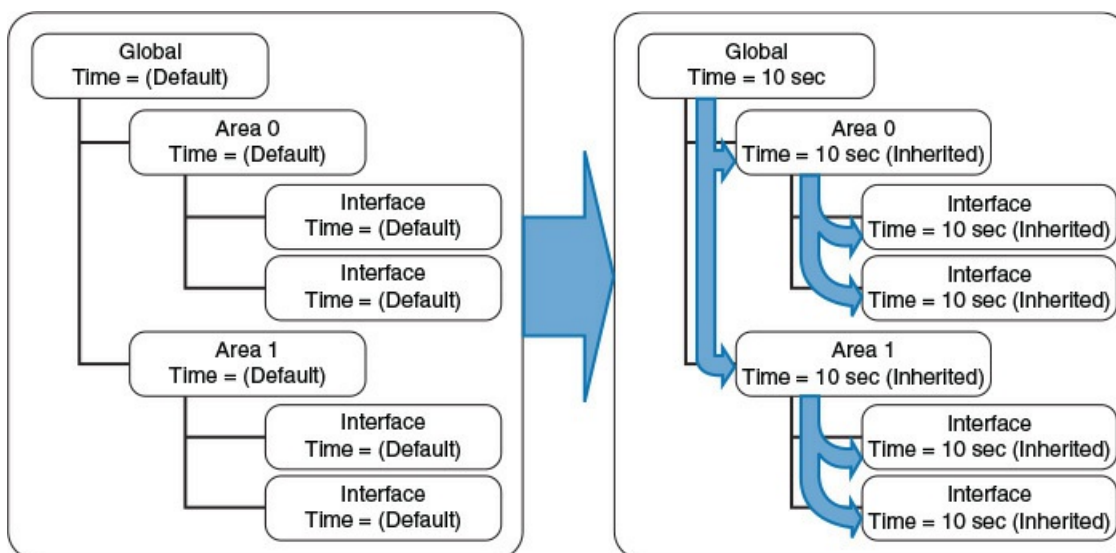


Figure 1-3 Global Inheritance

Hierarchical configurations allow for exceptions by allowing lower-level components to preempt inheritance. For example, if all interfaces should be set to 10 seconds but Area 1 needs to be 60 seconds, the inheritance flow will look like Figure 1-4.

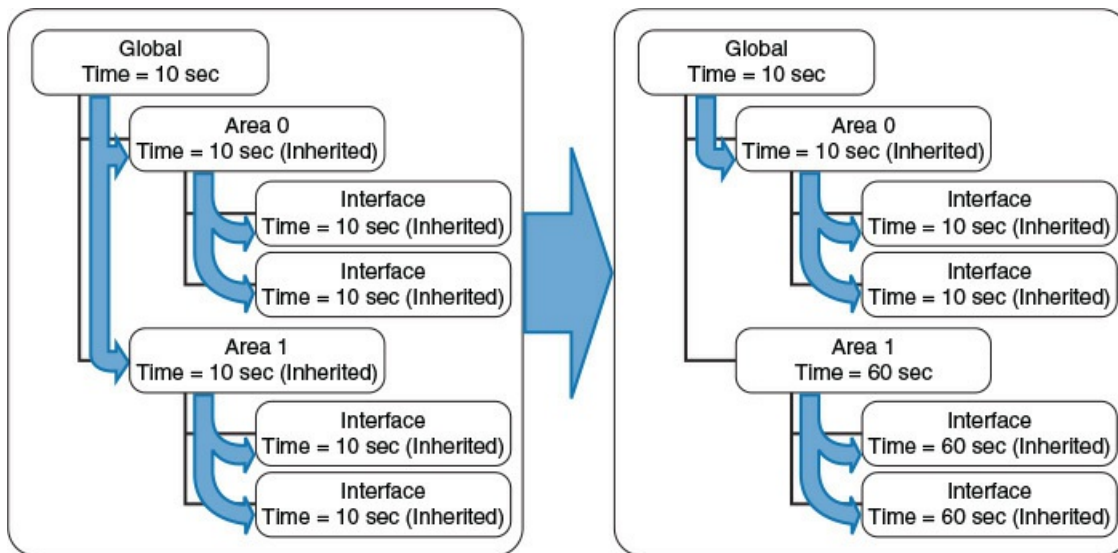


Figure 1-4 Global Inheritance with Lower-Level Preemption

This notion of preemption can be taken to a specific interface. It might seem simpler to place all the settings under the individual network interface, but if the router has 200 interfaces, that could be a lot of typing.

## PWD

IOS and IOS-XR configuration prompts change depending upon the submode while configuring the router. On IOS, the configuration prompt will mainly be under the interface `IOS-R2(config-if)#` or a router process `IOS-R2(config-router)#` submodes. Rarely does the IOS configuration require three levels of submode configuration.

IOS-XR configurations can be multiple submodes deep because of the hierarchical nature of the OS. To locate the submode of the configuration prompt, use the **pwd** command, which will display the context of the configuration prompt.

In [Example 1-33](#), the configuration requires three submodes deep into the BGP configuration. The command **pwd** is issued to clarify the exact CLI configuration mode.

### Example 1-33 Example of **pwd** on IOS XRs

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#conf terminal
RP/0/0/CPU0:XR1(config)#router bgp 100
RP/0/0/CPU0:XR1(config-bgp)#neighbor 1.1.1.1
RP/0/0/CPU0:XR1(config-bgp-nbr)#address-family ipv4 unicast
RP/0/0/CPU0:XR1(config-bgp-nbr-af)#pwd
router bgp 100 \n neighbor 1.1.1.1 \n address-family ipv4 unicast
```

## Root

Moving between subconfigurations in IOS XR OS may require going back to the global configuration before changing configuration submodes. Instead of typing **exit** multiple times to take you to the global process, the command **root** may be used.

[Example 1-34](#) demonstrates the manual **exit** command method.

### Example 1-34 Example Usage of Manually Exiting a Nested Configuration

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1 (config-bgp-nbr-af) #exit  
RP/0/0/CPU0:XR1 (config-bgp-nbr) #exit  
RP/0/0/CPU0:XR1 (config-bgp) #exit  
RP/0/0/CPU0:XR1 (config) #
```

[Example 1-35](#) demonstrates using the **root** command.

### Example 1-35 Example Usage of root

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1 (config-bgp-nbr-af) #root  
RP/0/0/CPU0:XR1 (config) #
```

## SUMMARY

After reading this chapter, you should have a fundamental understanding of the architectures for the IOS, IOS XE, and IOS XR network operating systems. The operating systems described have their own advantages and disadvantages affecting the fundamental behavior within the operating system.

This chapter has provided you with the necessary information to log in to a router and apply sample configurations that are provided throughout this book. Features around viewing historical configurations have been provided, along with the techniques to restore a prior configuration should you make a mistake while learning how to configure these operating systems.

For more information on IOS and IOS XR software architecture, the books *Inside Cisco IOS Software Architecture* and *Cisco IOS XR Fundamentals* provide comprehensive coverage of the operating systems.

## REFERENCES IN THIS CHAPTER

Tahir, Mobeen, et al. *Cisco IOS XR Fundamentals*. Indianapolis: Cisco Press, 2009. Print.

Bollapragada, Vijay, et al. *Inside Cisco IOS Software Architecture*. Indianapolis: Cisco Press, 2000. Print.

Khan, Muhammad Afaq. *Building Service-Aware Networks: The Next-Generation WAN/MAN*. Indianapolis: Cisco Press, 2010. Print.

Cisco. Cisco IOS Software Configuration Guides. <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides. <http://www.cisco.com>

## Chapter 2. IPv4 Addressing

This chapter covers the following topics:

- IPv4 addressing
- Subnet masks and subnetting
- Private IP addressing
- Special IP addresses
- IPv4 address configuration
- Wildcard subnet mask

All devices on the Internet communicate with each other using the Internet protocol suite, more commonly known as *Transmission Control Protocol/Internet Protocol* (TCP/IP). The numeric identifier that TCP/IP assigns to every device in the world is called an *IP address*. IP has evolved significantly over the years to allow for the incredible growth of the Internet. This chapter covers IP addressing fundamentals, as well as the protocol enhancements that have been introduced over time to conserve address space and improve route table scalability. The advancements in IP are covered in chronological order to provide a historical perspective and to show how each new feature builds upon the last.

### IP FUNDAMENTALS

Three key components allow communication between devices using TCP/IP: the IP address, subnet mask, and a default gateway.

The *IP address* is a 32-bit long unique value that represents a host on a network. The 32 bits of the IP address are commonly expressed in dot-decimal notation, such as 192.168.0.10. Each decimal value represents 8 binary bits of the IP address. The four decimal numbers that form the IP address may have a value ranging from 0 to 255, and they are always separated by a period. Later in this chapter the binary representation of an IP address is explained.

An IP address is divided into two components: a *network identifier* and a *host identifier*.

IP Address = Network Identifier + Host Identifier

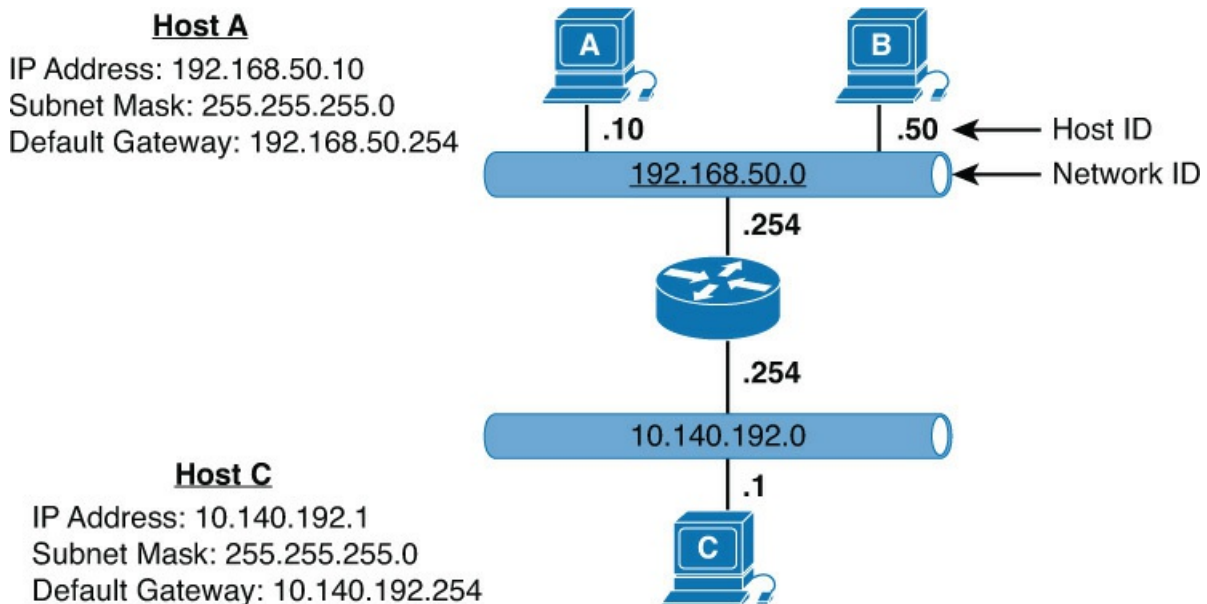
The network ID is the portion of the IP address that will be shared by all devices in the same network, and the host ID is the value that will be unique to the device for that network. A host's IP address, along with its associated subnet mask, is required to determine the network ID and host ID. If a host needs to send packets to a destination address that has the same network ID, it can do so directly without the need for a gateway. If a host needs to send packets to a destination that has a different network ID, the device immediately knows that it will need to send the packet to the local gateway so that a router can forward the packet off of the local network.

Delivering an IP packet is, in many respects, similar to delivering the mail. A mailman needs an address in order to know where to deliver a letter, just as a router needs an IP address to know where to deliver a packet. Every house on the same street shares a common street name; similarly, every device attached to the same local network will have identical network IDs. By taking this analogy one step further, every device has a unique host ID, just as every house has a unique street address.

The *subnet mask* is a 32-bit-long value used to identify IP address network boundaries. The purpose of a subnet mask is to clarify the IP address network ID and host ID.

The *default gateway* IP address is used by the host to identify the network exit point. The gateway address will be the IP address of the local router that provides connectivity to the other networks. If two hosts do not have the same network ID, they require a router to communicate.

In [Figure 2-1](#), host A (192.168.50.10) and host B (192.168.50.50) are connected to the same network 192.168.50.0/24. Because the hosts are members of the same network, they may communicate directly with each other. Host C (10.140.192.1) is a member of a different network, 10.140.192.0/24. For host A and host B to communicate with host C, a gateway router is required.



**Figure 2-1** IP Address Components

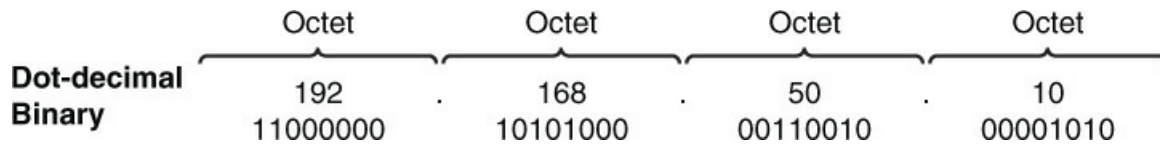
Note

The term *gateway* or *default gateway* often refers to a router from a host's standpoint.

## UNDERSTANDING BINARY

Computers read an IP address in binary form, but reading an IP address in binary is difficult to read quickly, so the IP address is referenced in dot-decimal notation. An example of a dot-decimal IP address is 192.168.50.10, which looks like 11000000 10101000 00110010 00001010 in binary notation. [Figure 2-2](#) displays the IP address in both formats, dot-decimal and binary. Every 8 bits of binary is called a *byte*, but when referring to an IP address, this byte is represented as a decimal number and called an *octet*. The 32-bit-long address is converted into four decimal numbers that are separated by a period.





**Figure 2-2** IP Address

Binary numbers can have only two values, represented as either a 1 or a 0. If the binary bit value is 1, it is considered “on” and counts toward the computed decimal value of the IP address. If the bit is 0, it is considered “off” and it does not count toward the final decimal value tally. [Table 2-1](#) displays the value for each binary bit.

<b>Bit</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>
Value	128	64	32	16	8	4	2	1

**Table 2-1** Binary Bit Values

Binary base 2 calculations:

Bit 1 Value =  $2^7 = 128$

Bit 2 Value =  $2^6 = 64$

Bit 3 Value =  $2^5 = 32$

Bit 4 Value =  $2^4 = 16$

Bit 5 Value =  $2^3 = 8$

Bit 6 Value =  $2^2 = 4$

Bit 7 Value =  $2^1 = 2$

Bit 8 Value =  $2^0 = 1$

Note

Binary bit values are power of 2s, starting from right to left.

If all the bits in an octet are set to 1, the on position, the dot-decimal value for the octet adds up to 255.

$$128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$$

It is easy to see that if all the bits are set to off (0), the dot-decimal value will equal 0.0.0.0.

Binary = 00000000 00000000 00000000 00000000

Dot-decimal = 0.0.0.0

If all the binary bits are set to on (1), the dot-decimal value will equal 255.255.255.255.

Binary = 11111111 11111111 11111111 11111111

Dot-decimal = 255.255.255.255

With the knowledge of binary math, it is possible to convert an IP address or subnet mask from dot-decimal to binary and vice versa. Figure 2-3 illustrates the full conversion of the IP address 10.140.192.1 to binary. Notice that each octet is independently converted. To start, compare the decimal value of the IP address to the binary bit column value. If the decimal value is higher than the bit column value, the bit can be flipped to on (1), and the binary value subtracted. The new value is compared in the next column. If the value is smaller than the binary value, the bit stays 0 or off. Repeat this exercise until the decimal value has reached 0 to complete the binary translation of the dot-decimal IP address.

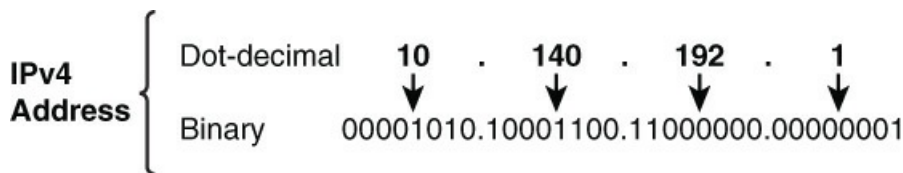


Figure 2-3 10.140.192.1 Decimal to Binary Conversion

Figures 2-4 through 2-7 demonstrate how to calculate the binary on bits for each octet of the IP address 10.140.192.1.

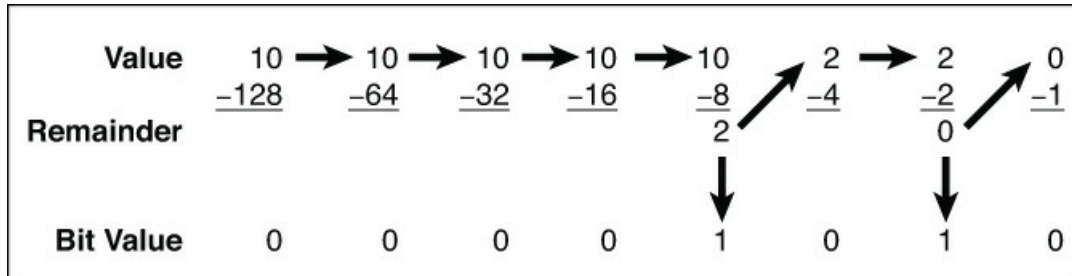


Figure 2-4 Decimal to Binary Calculation for First Octet

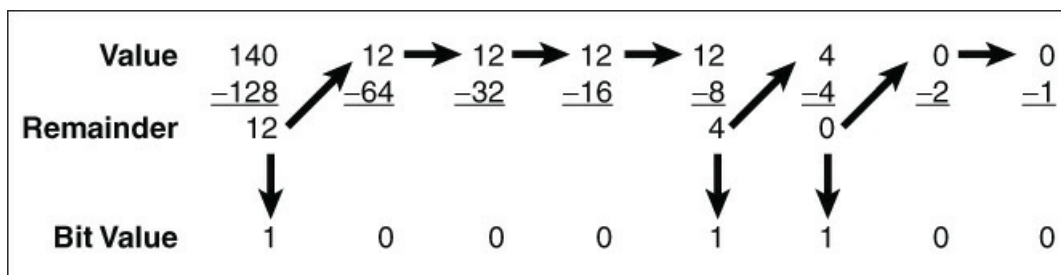


Figure 2-5 Decimal to Binary Calculation for Second Octet

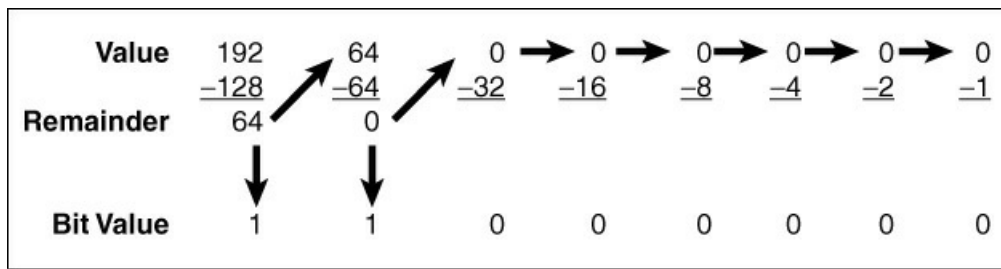


Figure 2-6 Decimal to Binary Calculation for Third Octet

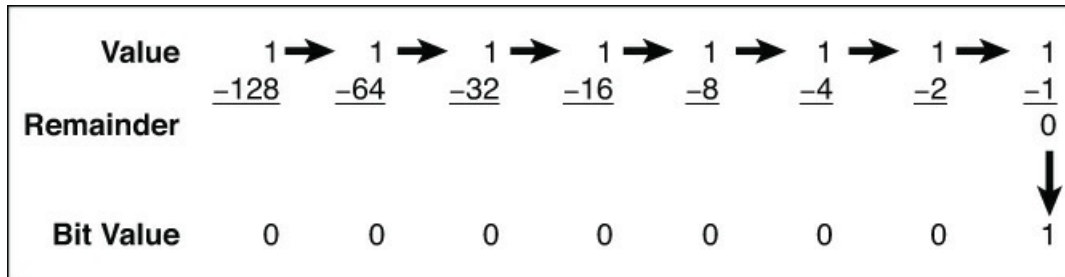


Figure 2-7 Decimal to Binary Calculation for Fourth Octet

To convert binary to decimal involves lining up the bit with the corresponding decimal value and adding up the values to create the decimal value. Figure 2-8 demonstrates converting the binary value 11000010 to decimal 194.

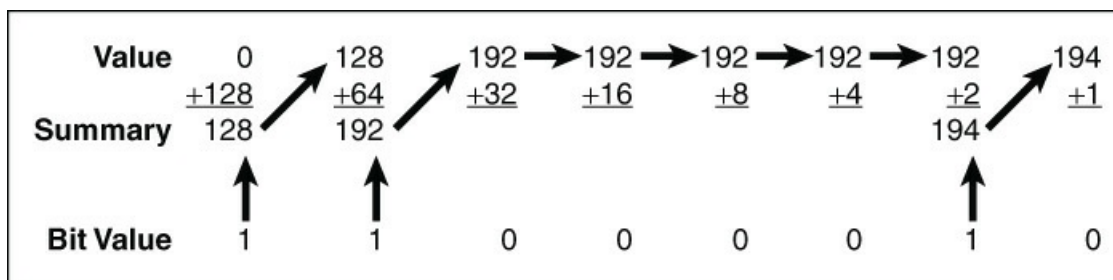


Figure 2-8 Binary to Decimal Conversion for 11000011

## ADDRESS CLASSES

A brief history of the Internet and the development of IP is helpful for understanding the rationale for address classes.

When IP was in its early development the concept of subnet masks did not exist. The first 8 bits were considered the network number (network ID), and the remaining 3 octets were called the rest field (host ID). The limitation of using 8 bits for the network block meant that the early Internet was not very scalable. At that time, 8 bits ( $2^8$ ) would allow a maximum of only 256 networks!

In 1981, RFC 791 temporarily addressed this scalability problem by introducing the concept of address classes so that a multitude of new networks could be formed of various sizes. The first 4 leading bits of the IP address were designated for identifying the class of network. The first binary bits in the IP address are called various names depending on the source. In this text, the bits are referred to as the *leading bits*, but the bits may also be called the *most significant bits* or *high-order bits*. The leading bit indicates the class of network to which the IP address belongs.

Table 2-2 outlines the number of available hosts in each IP address class along with the leading binary bit and address range.

Class	Leading Bits	Available Networks	Available Hosts per Network	Address Range	Derived Network Mask
Class A	0	128 ( $2^7$ )	$16,777,214 - (2^{24}) - 2$	0.0.0.0 – 127.255.255.255	255.0.0.0
Class B	10	16,384 ( $2^{14}$ )	$65,534 - (2^{16}) - 2$	128.0.0.0 – 191.255.255.255	255.255.0.0
Class C	110	$2,097,152 - (2^{21})$	$254(2^8) - 2$	192.0.0.0 – 223.255.255.255	255.255.255.0
Class D (Multicast)	1110	—	—	224.0.0.0 – 239.255.255.255	—
Class E (Reserved)	1111	—	—	240.0.0.0 – 255.255.255.255	—

Table 2-2 IP Address Classes

Class A networks start with a binary bit value 0 and can have a dot-decimal range of 0.0.0.0 to 127.255.255.255. The network number bit field is 7 bits, and the remaining 24 bits are for local host addressing. This class provides an extremely large number of IP addresses:  $16,777,214 (2^{24} - 2)$  usable IP addresses per network. A total of  $128 (2^7)$  networks are available in this class. Only extremely large institutions like governments were assigned a full Class A network block. Figure 2-9 displays the IP address fields of a Class A IP address.

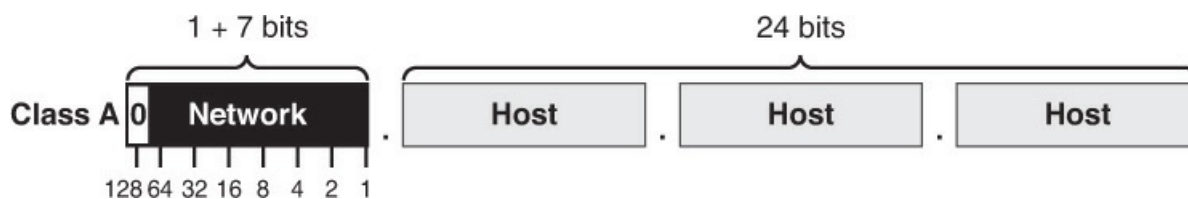


Figure 2-9 Class A IP Address

Class B networks start with a leading binary bit value of 10 and have a dot-decimal range of 128.0.0.0 to 191.255.255.255. This network block provides 14 bits for networks and 16 bits for hosts. Class B networks were assigned to medium to large organizations. A total of  $16,384 (2^{14})$  networks are within the class, with  $65,534 (2^{16} - 2)$  usable IP addresses per network. Figure 2-10 displays the IP address fields of a Class B IP address.

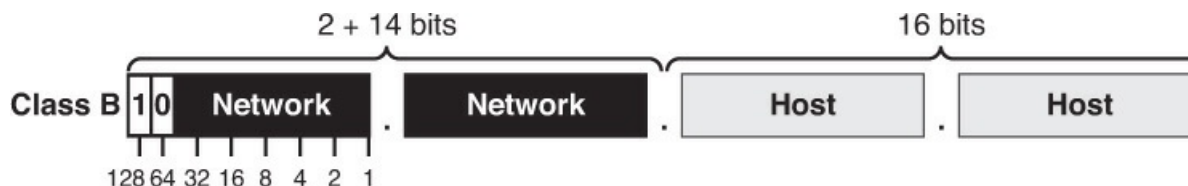


Figure 2-10 Class B Address

Class C networks start with a leading order bit value of 110 and have a dot-decimal range of 192.0.0.0 to 223.255.255.255. The network number bit field is 21 bits long, and the remaining 8 bits are for hosts. The Class C address block was dedicated to small organizations. A total of  $2,097,152 (2^{21})$  networks are available in this class, with  $254 (2^8 - 2)$  usable IP addresses per network. Figure 2-11 displays the IP address fields of a Class C IP address.

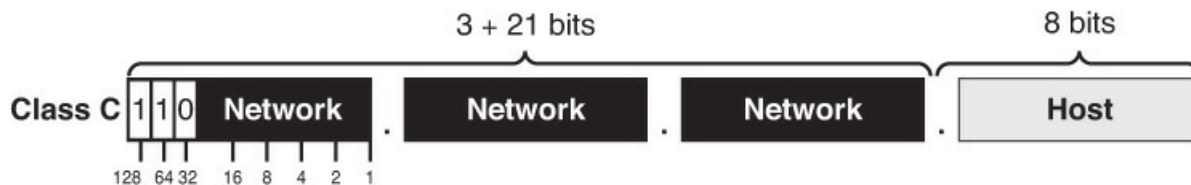


Figure 2-11 Class C Address

Class D addresses are defined with leading bits 1110 and reserved for multicast routing. The Class D dot-decimal range is 224.0.0.0 to 239.255.255.255. The final class is E and is reserved for future use. A Class E address starts with leading binary value of 1111 and is in the dot-decimal range of 240.0.0.0 to 255.255.255.255.

Note

Classful routing and Class A, Class B, and Class C address classes are considered legacy and are no longer commonly used in modern networks. Today's networks are classless and do not follow strict classful network boundaries. Classless interdomain routing (CIDR) and its benefits are covered later in this chapter.

## SUBNET MASKS AND SUBNETTING

Subnet masks and subnetting improve address conservation by eliminating the rigid network boundaries that are present with classic address classes. Classful networks use the leading bits of the IP address to determine the network ID and thus the size of the network. Subnetting, however, uses the IP address along with a subnet mask to determine the network ID, allowing for smaller subnetworks (subnets) and improved address utilization.

### Subnet Mask Purpose

Together, the subnet mask and IP address are used to identify the boundary between the network ID and host ID. Host A has an IP address of 192.168.50.10 with a subnet mask of 255.255.255.0.

Figure 2-12 shows the IP address and subnet mask converted to binary and aligned under each other. The subnet mask is used to “mask” the bits that are used to determine the network ID in an IP address. The subnet mask’s binary bits that are in the on position, value of 1, indicate the bits of the IP address that will be used to create the network ID. The subnet mask bits that are off, value of 0, indicate the host ID of the address. In this instance, the boundary is between the 24th and 25th bit. The first 24 bits form the network ID 192.168.50, and the last 8 bits form the host ID 10.

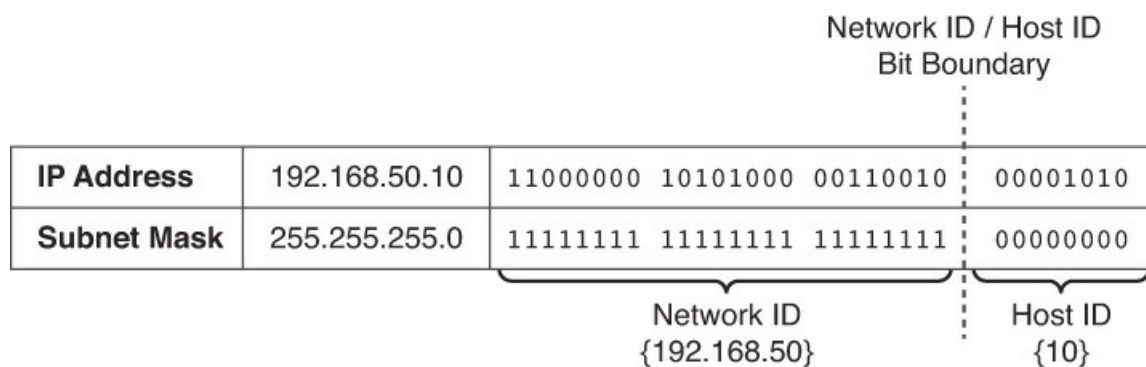


Figure 2-12 Network ID and Host ID Boundary

In the network topology from Figure 2-1, host A (192.168.50.10) wants to send traffic to host B (192.168.50.50). The source IP address and subnet mask are used to determine the local network ID and host range. The network ID for host A is 192.168.50.0, which matches the first 24 bits of

the binary address for host B, as indicated in [Table 2-3](#). Because the bits match, the two hosts have the same network ID and a router is not necessary for the two to communicate.

Source IP (Host A)	192.168.50.10	11000000 10101000 00110010 00001010
Subnet Mask	255.255.255.0	11111111 11111111 11111111 00000000
Network ID	192.168.50	11000000 10101000 00110010 00000000
Destination IP (Host B)	192.168.50.50	11000000 10101000 00110010 00110010
Action	The two hosts have the same network ID, so a router is not necessary for the two hosts to communicate.	

**Table 2-3** *Sender's Network ID Matches Destination Packet*

[Table 2-4](#) provides an example where host A wants to communicate with host C (10.140.192.1). The network ID does not match, so a router is required to transmit the packet to the network that contains host C.

Source IP(Host A)	192.168.50.10	11000000 10101000 00110010 00001010
Subnet Mask	255.255.255.0	11111111 11111111 11111111 00000000
Network ID	192.168.50.	11000000 10101000 00110010 00000000
Destination IP(Host C)	10.140.192.1	00001010 10001100 11000000 00000001
Action	The two hosts have a different network ID, so the source needs to forward the packet to the local gateway router.	

**Table 2-4** *Sender's Network ID Does Not Match Destination Packet*

### Calculating Usable IP Addresses

It is important to note that a device cannot be assigned a host ID that has a binary value of all 0s or all 1s.

The all-0s host address is reserved by the network ID as a way to reference the network. For example, the correct way to reference the network ID used by 192.168.0.10 is to say that the 192.168.0.10 address is in the 192.168.0.0 255.255.255.0 network.

The all-1s host address is reserved for broadcast. The broadcast address is a special address that can be used to communicate with all the devices on the local network at the same time. For example, the broadcast address for the 192.168.0.0 255.255.255.0 network is 192.168.0.255.

To calculate the number of usable IP addresses within a network, the formula is  $2^n - 2$ , where  $n$  is the number of host bits available and 2 is subtracted to account for the all-0s and broadcast address.

### How many available hosts formula:

$$2^n - 2$$

$n$  = Number of Host Bits

Consider the network 192.168.0.0 255.255.255.0, which has 8 host bits available:

$$\text{Total Hosts} = 2^8 - 2 = 254$$

Network Address = 192.168.0.0

Broadcast Address = 192.168.0.255

Note

In some older texts, you may see that the same all-0s and all-1s rules applied for calculating the number of usable subnets. Originally, RFC 950 strongly recommended avoiding subnet 0 and the all-1s subnet. Using subnet 0 for addressing was discouraged because the binary math for the subnet mask made it difficult to infer the network ID from the subnet address. This is no longer true, and the practice of avoiding subnet 0 and the all-1s subnet was made obsolete by RFC 1878. The `ip subnet-zero` command was introduced in IOS 12.0 to overcome this restriction. Today, IP subnet 0 is the default behavior for both IOS and IOS XR. IOS XR cannot disable IP subnet 0! The key takeaway is that the IP subnet 0 and all-1s subnet rule no longer applies to modern networks.

### Network Prefix Notation

The prefix length represents the number of leading binary bits in the subnet mask that are in the on position. The prefix notation is referenced quite often in this book because it is the shorthand way of expressing the subnet mask. For example, the prefix length /24 is a much simpler way of expressing subnet mask 255.255.255.0. With network prefix notation, the 192.168.50.0 255.255.255.0 network may be rewritten as 192.168.50.0/24.

Figure 2-13 demonstrates the prefix length notation for a Class A, B, and C network.

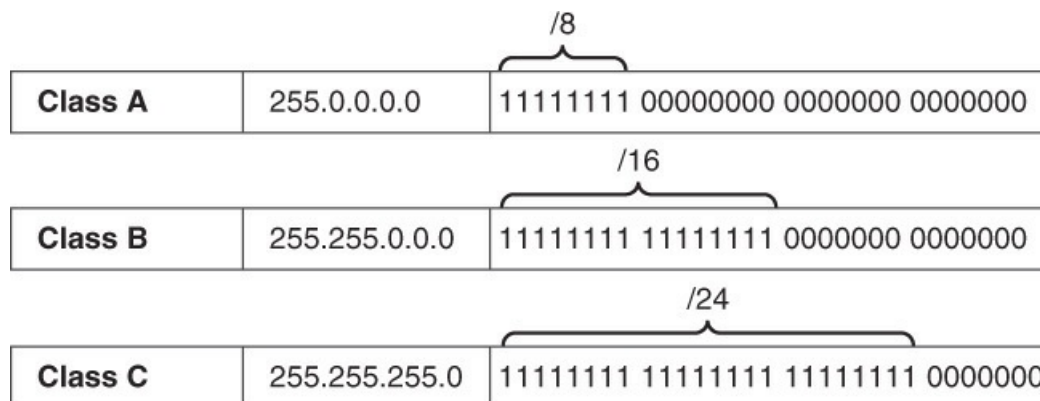


Figure 2-13 Class A, B, C Prefix Notation

Note

Network prefix notation may also be called *classless interdomain routing (CIDR)* notation. A more thorough overview of CIDR is covered later in this chapter.

### Subnetting

*Subnetting* is a method for partitioning a classful IP network into smaller subnetworks (subnets). The original intent of subnetting was to help scale the Internet and curtail the rapid depletion of IPv4 addresses. Over time, subnetting has also become a useful method for improving network performance by containing broadcast traffic because a broadcast will not cross a Layer 3 network boundary (router).

Figure 2-14 illustrates classful IP addressing for three LANs. Each LAN has 500 devices that require an IP address. A Class C network will be too small to accommodate all the hosts on each LAN because a Class C network can accommodate only 254 host IP addresses per network.



segment. The next largest class is a Class B, which provides 65,534 available host addresses. The IP addressing scheme uses three Class B networks (172.16.0.0/16, 172.17.0.0/16, and 172.18.0.0/16) to support the connectivity requirements.

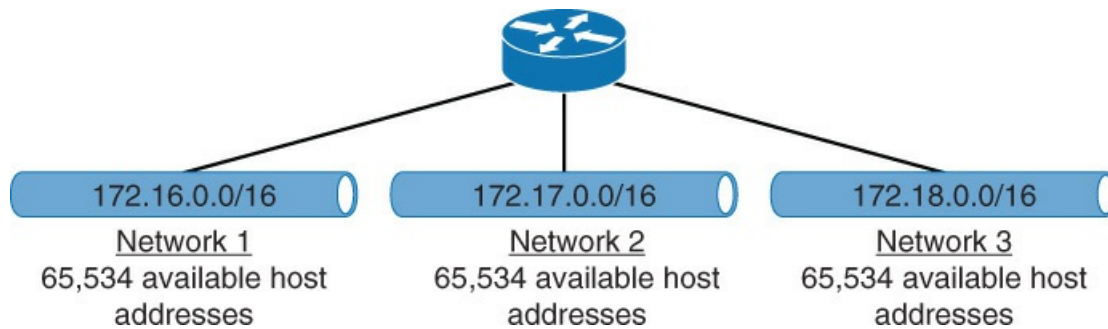


Figure 2-14 Network Topology Using Class B Networks

Unfortunately, classful IP addressing prevents the conservation of IP address space. In Figure 2-14, more than 99 percent of the IP addresses are not being utilized.

IP subnetting is used to subdivide a single classful network into smaller networks so that a more efficient utilization of the IP addresses can be achieved. In Figure 2-15, the Class B network 172.16.0.0 is subdivided into three subnets (172.16.0.0/23, 172.16.2.0/23, and 172.16.4.0/23) to provide connectivity for the three LANs. The remaining unused addresses (172.16.6.0–172.16.255.255) in the Class B network are conserved for future use.

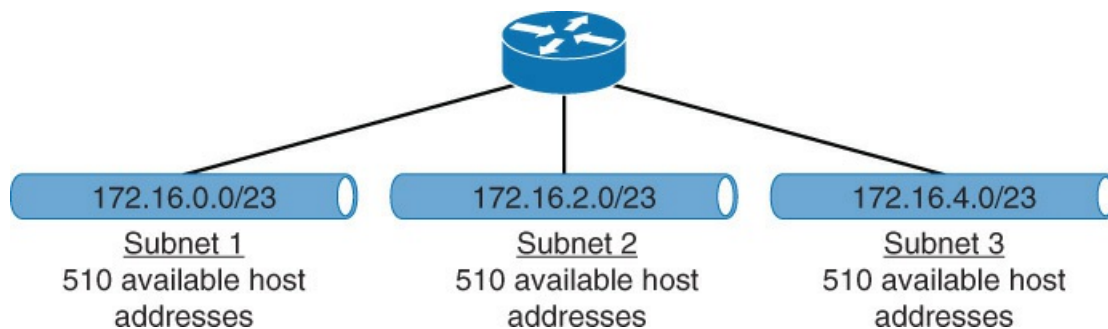


Figure 2-15 Network Topology Using Subnets

### Subnet Field

RFC 950 describes the mechanism for dividing a network into subnetworks through the use of a new subnet field in the IP address. The new field is located between the network mask and the host ID. Subnetting works by borrowing bits from the host ID portion of the IP address to extend the length of the network ID. Figure 2-16 demonstrates the placement of the subnet field in the IP address.

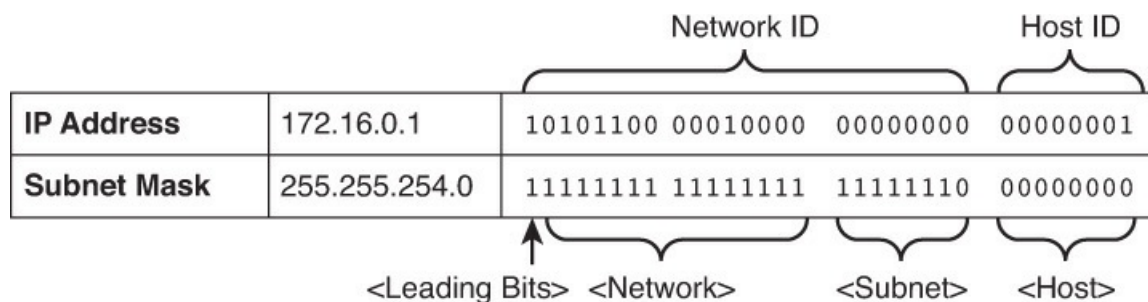


Figure 2-16 Subnetting IP Address Fields

Reassigning bits from the host ID to the network ID allows for more networks, but it also reduces



the number of available hosts per network. Table 2-5 provides a reference for the number of available subnets and usable hosts based on the subnet mask bit length.

<b>Class A Network</b>							
Prefix Length	Subnet Mask	Subnets ( $1 \times 2^n$ )	Usable Hosts ( $1 \times 2^n - 2$ )				
/8	255.0.0.0	0	16,777,214				
/9	255.128.0.0	2	8,388,606				
/10	255.192.0.0	4	4,194,302				
/11	255.224.0.0	8	2,097,150				
/12	255.240.0.0	16	1,048,574				
/13	255.248.0.0	32	524,286				
/14	255.252.0.0	64	262,142	<b>Class B Network</b>			
/15	255.254.0.0	128	131,070	Subnets ( $1 \times 2^n$ )	Usable Hosts ( $1 \times 2^n - 2$ )		
/16	255.255.0.0	256	65,534	0	65,534		
/17	255.255.128.0	512	32,766	2	32,766		
/18	255.255.192.0	1024	16,382	4	16,382		
/19	255.255.224.0	2046	8190	8	8,190		
/20	255.255.240.0	4096	4094	16	4,094		
/21	255.255.248.0	8192	2046	32	2,046		
/22	255.255.252.0	32,768	1022	64	1,022	<b>Class C Network</b>	
/23	255.255.254.0	65,536	510	128	510	Subnets ( $1 \times 2^n$ )	Usable Hosts ( $1 \times 2^n - 2$ )
/24	255.255.255.0	131,072	254	256	254	0	254
/25	255.255.255.128	262,144	126	512	126	2	126
/26	255.255.255.192	524,288	62	1024	62	4	62
/27	255.255.255.224	104,8576	30	2046	30	8	30
/28	255.255.255.240	2,097,152	14	4096	14	16	14
/29	255.255.255.248	4,194,304	6	8192	6	32	6
/30	255.255.255.252	8,388,608	2	32,768	2	64	2
/31	255.255.255.254	33,554,432	2	65,536	2	128	2

**Table 2-5** Subnet Mask Reference Chart

Note

/31 networks should be reserved for point-to-point router links.

## Subnet Math

One of the challenges of subdividing a classful network into multiple smaller subnetworks is identifying where the new IP address range starts and ends. The focus of this section is on how to calculate the subnet address numbers:

- Network address
- First usable host address
- Last usable host address
- Broadcast address

### Bitwise AND Operation

To calculate the network boundary, the host IP address and the subnet mask must first be converted binary. The two numbers are compared, and a logical AND operation is performed. To perform the bitwise AND operations, the binary values of the address and mask are lined up side by side. The AND operation result is a 1 if both binary numbers are 1 in the side-by-side comparison. If the bit values are any other combination, the AND result is 0. Figure 2-17 demonstrates how to calculate the network boundary for IP address 10.140.15.1 with subnet mask 255.255.252.0 using the bitwise AND method.

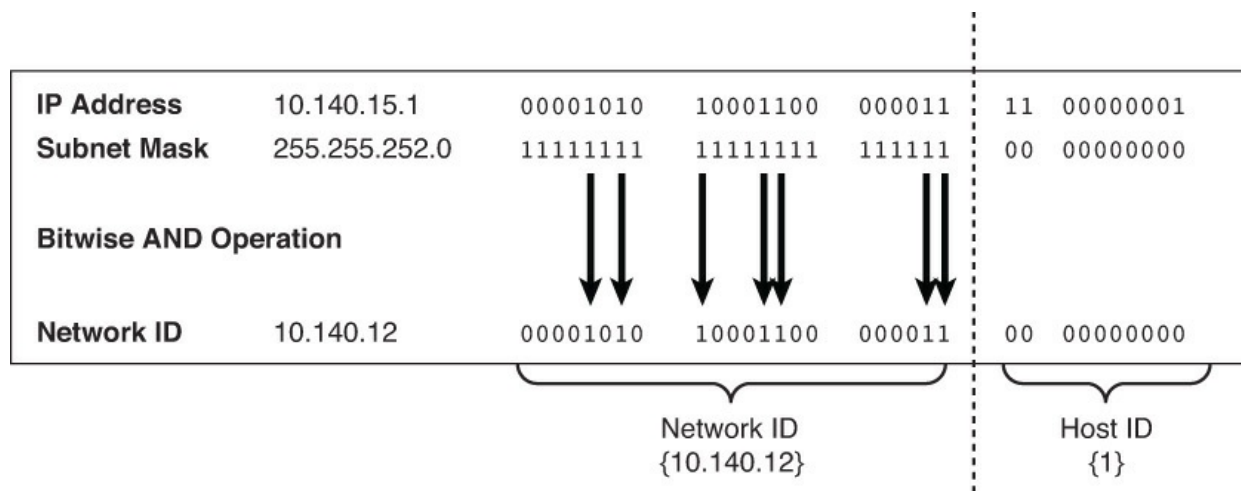
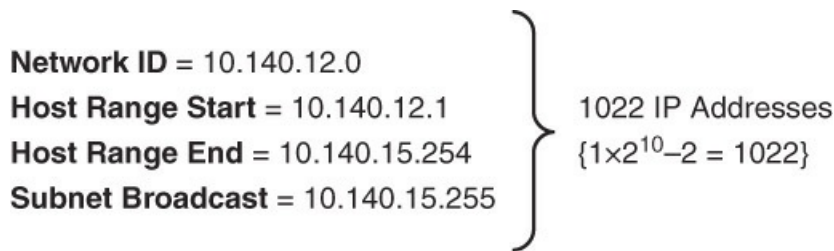


Figure 2-17 Bitwise AND Operation for 10.140.15.1 255.255.252.0

The computed network ID for 10.140.15.1 255.255.252.0 is 10.140.12.0. Any packet that the host sends to an IP address that has the same network ID will be sent to the local host directly. If a packet does not have the same network ID, the host forwards the packet to the local gateway router for transmission off-net. Binary math can be used to determine the range of hosts that will share the same network ID. The host ID is 10 bits long, so the binary math calculation is  $2^{10} = 1024$  hosts. Each octet of an IP address holds a total of 256 addresses, 0 to 255, so the host range for the network is 10.140.12.1 to 10.14.15.254. The first address, host all 0s, is reserved for the network ID; and the last address, host all 1s, is reserved for the broadcast address. Figure 2-18 provides the calculated host range for the subnet 10.140.12.0/22.



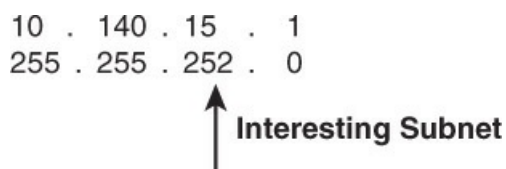
**Figure 2-18** 10.140.12.0/22 Calculated Host Range

### Magic Number Method

Calculating subnets using the bitwise AND method can be quite time-consuming. Another approach that the network administrator may use is called the *magic number method*. The magic number method does not first require converting numbers to binary; instead, the calculation is based on an astute examination of the IP address and subnet mask.

The IP address 10.140.15.1 and subnet mask 255.255.252.0 is used again to demonstrate the magic number method.

The first step is to examine the octet values in the subnet mask 255.255.252.0. The octet of interest is the first non-255 numbered octet. This value in this octet is called the *interesting number*. [Figure 2-19](#) highlights the interesting subnet octet.



**Figure 2-19** Interesting Subnet Octet

The third octet of the subnet mask is 252 instead of 255. This is enough information to determine that the network ID must start with 10.140 because the first two octets are 255.255, which is an all-1s binary combination that will mask the two octets. Next, examine the third octet of the subnet mask, which has a value of 252. The 8 bits in an octet add up to a total value of 256. Subtract the third octet’s value 252 from 256 and call the result the *magic number*.

### Calculating Magic Number:

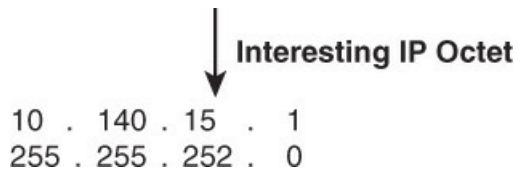
$$256 - \text{Interesting Subnet} = \text{Magic Number}$$

$$256 - 252 = 4$$

Note

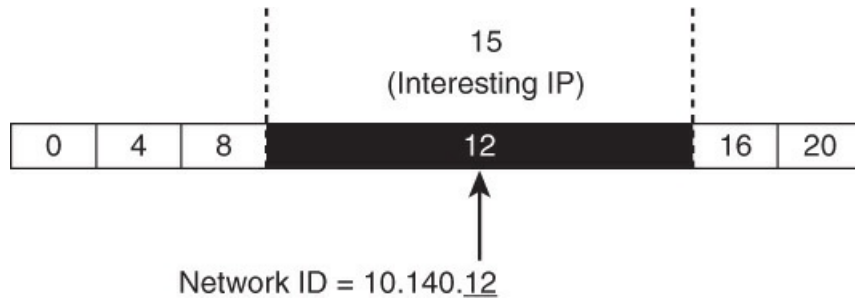
You may be wondering why the subnet mask states 255 if the 8 bits in an octet add up to a value of 256. The mask is reported as 255 because counting starts at 0, instead of at 1.

The magic number is 4. The host range is going to take up to 4 decimal values in the third octet. To determine the network ID, the third octet of the IP address 10.140.15.1 is scrutinized. The network ID will be the closest magic number multiple that is less than or equal to the interesting IP octet. [Figure 2-20](#) highlights the interesting IP address octet.



**Figure 2-20** Interesting IP Address Octet

The interesting IP address is 15, which falls between 12 and 16. 12 is chosen because it is the magic number multiple (4) that is less than or equal to the interesting IP address 15. [Figure 2-21](#) displays the magic number multiples of 4 and highlights which number is chosen for the network ID's third octet.



**Figure 2-21** Closest Magic Number Multiple That Is Less Than or Equal to Interesting IP Octet

Using the 12 for the third octet, the calculated network's address is 10.140.12.0.

To determine the first usable host address in the subnet, add 1 to the network address:

$$\text{First Usable Host Address} = 10.140.12.0 + 1 = 10.140.12.1$$

To determine the broadcast address in the subnet, add the magic number to the network ID, and then subtract 1 from the result:

$$\text{Broadcast Address} = (10.140.12 + 4) - 1 = 10.140.15.255$$

To determine the last usable host address in the subnet, add the magic number to the network ID, and then subtract 2 from the result:

$$\text{Last Usable Host Address} = (10.140.12 + 4) - 2 = 10.140.15.254$$

[Figure 2-22](#) demonstrates the full magic number calculation for determining the first usable address, last usable address, and subnet broadcast address.

Network-ID = 10.140.12.0  
Magic Number = 4

$$\begin{array}{r} \text{First Usable Host Address} \quad \rightarrow \quad 10.140.12.0 \\ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad +1 \\ \hline \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \mathbf{10.140.12.1} \end{array}$$

$$\begin{array}{r} \text{Last Octet in Subnet} \quad \rightarrow \quad 10.140.12 \\ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad +4 \\ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \hline \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad -1 \\ \hline \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \mathbf{10.140.15} \end{array}$$

$$\begin{array}{r} \text{Last Usable Host Address} \quad \rightarrow \quad 10.140.15.256 \\ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad -2 \\ \hline \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \mathbf{10.140.15.254} \end{array}$$

$$\begin{array}{r} \text{Subnet Broadcast Address} \quad \rightarrow \quad 10.140.15.256 \\ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad -1 \\ \hline \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \mathbf{10.140.15.255} \end{array}$$

**Figure 2-22** Subnet Address Range Calculations

To calculate the subnet mask addresses using the magic number method, use the following steps:

**Step 1.** Subnet Mask (Interesting Number)

Identify the interesting octet in the subnet mask. This will be the first octet that is not 255.

**Step 2.** IP Address (Interesting Number)

Identify the interesting octet in the IP address. This will be the same octet as the interesting subnet mask.

**Step 3.** Magic Number

Subtract the interesting subnet mask number from 256 to determine the magic number.

**Step 4.** Network ID

Decipher the network ID by comparing the multiples of the magic number to the interesting IP address. The network ID will be the closest multiple that is less than or equal to the interesting IP octet.

**Step 5.** First Address

The first usable IP address will be one number higher than the network ID.

**Step 6.** Last Address

To calculate the last address in the subnet, add the magic number to the network ID, and then subtract 2.

### Step 7. Broadcast Address

To calculate the broadcast address in the subnet, add the magic number to the network ID, and then subtract 1.

Table 2-6 through Table 2-8 provide three more examples of the magic number method for calculating the address range for a subnet.

	<b>Address</b>	<b>Notes</b>
<b>Subnet Mask</b>	255.240.0.0	Interesting subnet address = 240
<b>IP Address</b>	10.55.200.33	Interesting IP address = 55
<b>Network ID</b>	10.48.0.0	Magic number = $256 - 240 = 16$ Interesting IP address = 55 55's closest multiple of 16 $\leq 48$
<b>First Address</b>	10.48.0.1	Add 1 to the network ID
<b>Last Address</b>	10.63.255.254	Add the magic number to the network ID and subtract 2 from the network $10.48.0.0 + 16 = 10.64.0.0$ $10.64.0.0 - 2 = 10.63.255.254$
<b>Broadcast</b>	10.63.255.255	Add the magic number to the network ID and subtract 1 from the network $10.48.0.0 + 16 = 10.64.0.0$ $10.64.0.0 - 1 = 10.63.255.255$

**Table 2-6** 10.55.200.33/12 Subnet Address Range Calculations

	<b>Address</b>	<b>Notes</b>
<b>Subnet Mask</b>	255.255.254.0	Interesting subnet address = 254
<b>IP Address</b>	172.16.2.3	Interesting IP address = 2
<b>Network ID</b>	172.16.2.0	Magic number = $256 - 254 = 2$ Interesting IP address = 2 2's closest multiple of $2 \leq 2$
<b>First Address</b>	172.16.2.1	Add 1 to the network ID
<b>Last Address</b>	172.16.3.254	Add the magic number to the network ID and subtract 2 from the network $172.16.2.0 + 2 = 172.16.4.0$ $172.16.4.0 - 2 = 172.16.3.254$
<b>Broadcast</b>	172.16.3.255	Add the magic number to the network ID and subtract 1 from the network $172.16.2.0 + 2 = 172.16.4.0$ $172.16.4.0 - 1 = 172.16.3.255$

**Table 2-7** 172.16.2.3/23 Subnet Address Range Calculations

	<b>Address</b>	<b>Notes</b>
<b>Subnet Mask</b>	255.255.255.192	Interesting subnet address = 192
<b>IP Address</b>	192.168.100.5	Interesting IP address = 5
<b>Network ID</b>	192.168.100.0	Magic number = $256 - 192 = 64$ Interesting IP address = 64 5's closest multiple of $64 \leq 0$
<b>First Address</b>	192.168.100.1	Add 1 to the network ID
<b>Last Address</b>	192.168.100.62	Add the magic number to the network ID and subtract 2 from the network $192.168.100.0 + 64 = 192.168.100.64$ $192.168.100.64 - 2 = 192.168.100.62$
<b>Broadcast</b>	192.168.100.63	Add the magic number to the network ID and subtract 1 from the network $192.168.100.0 + 64 = 192.168.100.64$ $192.168.100.64 - 1 = 192.168.100.63$

**Table 2-8** 192.168.100.5/26 Subnet Address Range Calculations

### Subnet Design

A common task for a network administrator is to plan IP address allocation schemes for networks. Typically, the goal is to subdivide the network so that IPs can be conserved, while still allowing for future network growth.

The local-area networks (LANs) in [Figure 2-23](#) require IP addressing. The challenge is to subdivide a single Class C network, 192.168.100.0/24, into subnets to fulfill the host requirements for each LAN.

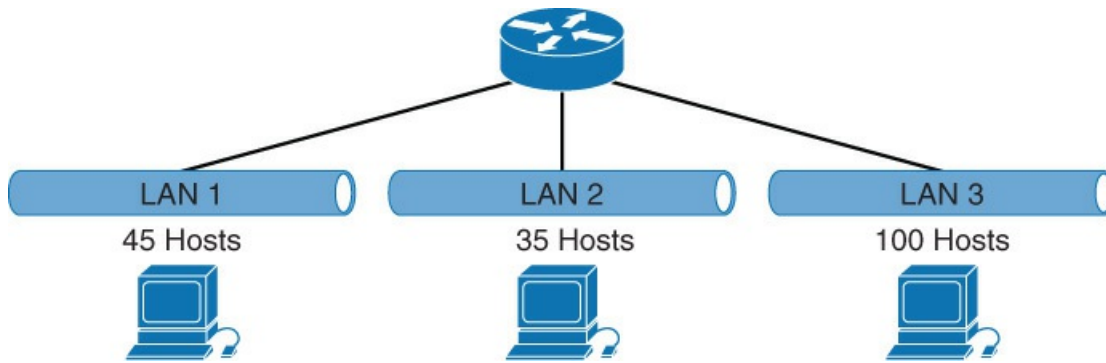


Figure 2-23 192.168.100.0/24 Subnet Challenge Topology

To determine the custom subnet masks for the network, it is helpful to start by creating a table for the available subnets and hosts. The 192.168.100.0/24 network prefix has a subnet mask of 255.255.255.0, so only the 8 bits in the last octet are available for subnetting.

The first task for creating the table is listing out the binary bit values for an octet. The binary bit values are used later to calculate the subnet mask. Figure 2-24 provides a binary bit values table for the 8 available subnetting bits for the network address 192.168.100.0.

192.168.100.	128	64	32	16	8	4	2	1	<b>Binary bit values</b>
--------------	-----	----	----	----	---	---	---	---	--------------------------

Figure 2-24 Subnetting Table: Binary Bit Values

The second step is to list the available host values from right to left. To calculate the number of usable IP addresses within a network the formula is  $2^n - 2$ , starting right to left, where  $n$  is the number of host bits available. Figure 2-25 indicates the number of available hosts IP addresses available based on the number of binary bits available for the host ID.

192.168.100.	128	64	32	16	8	4	2	1	<b>Binary bit values</b>
	254	126	62	30	14	6	2	1	
	← <b>Available hosts</b>								

Figure 2-25 Subnetting Table: Binary Bit Values + Available Hosts

The third and final step is to list the available subnets. The subnets are calculated by counting the number of bits, from left to right, and using the formula  $2^n$ , where  $n$  is the number of subnet bits. Figure 2-26 indicates the number of available subnets for network 192.168.100.0 based on the number of binary bits provided to the network ID.

	→ <b>Available Subnets</b>								
	2	4	8	16	32	64	128	256	
192.168.100.	128	64	32	16	8	4	2	1	<b>Binary bit values</b>
	254	126	62	30	14	6	2	1	
	← <b>Available hosts</b>								

Figure 2-26 Subnetting Table: Binary Bit Values + Available Hosts + Available Subnets

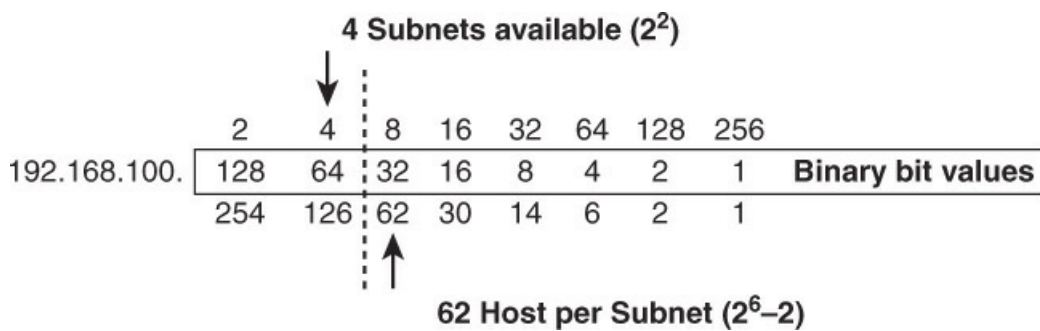
Now that the table is completed, the job of picking the correct number of bits for the subnet mask is simplified.



The example network in [Figure 2-23](#) includes three LANs. The IP address requirements for each LAN are outlined here for reference.

- **LAN 1:** 45 hosts
- **LAN 2:** 35 hosts
- **LAN 3:** 100 hosts

LAN 1 and 2 will require more than 30 hosts, but fewer than 62. Borrowing 2 host bits will generate 4 subnets, each having 62 hosts per subnet. [Figure 2-27](#) demonstrates that extending the subnet mask by 2 bits allows for the creation of 4 subnets. The remaining 6 bits are available for the host ID, allowing for 62 hosts per subnet.



**Figure 2-27** /26 Subnet

[Table 2-9](#) provides an overview of the current subnet assignments.

	<b>Address</b>	<b>Notes</b>
Subnet mask	255.255.255.192	128 + 64 = 192 (/26)
Subnet 1	192.168.100.0	Assign to LAN 1
Subnet 2	192.168.100.64	Assign to LAN 2
Subnet 3	192.168.100.128	—
Subnet 4	192.168.100.192	—

**Table 2-9** Subnetting Challenge Table: LAN 1 + LAN 2

LAN 1 and 2 do not require the addresses 192.168.100.128–192.168.100.254, so the remaining addresses can be allotted for creating a larger subnet to support LAN 3’s requirements of 100 IP addresses. [Figure 2-28](#) demonstrates that borrowing 1 bit from the host ID will allow for 2 subnets, each having 126 usable IP addresses per subnet.

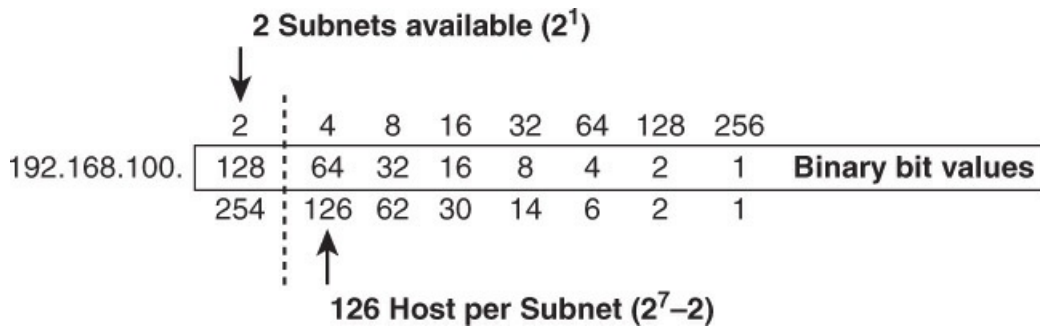


Figure 2-28/25 Subnet Table

The first subnet is already accounted for by LAN 1 and LAN 2. LAN 3 can claim the second subnet to fulfill the network requirements. Table 2-10 indicates that the address range 192.168.100.0 to 192.168.100.127 (192.168.100.0/25) is already consumed by LAN 1 (192.168.100.0/26) and LAN 2 (192.168.100.64.0/26).

	Address	Notes
Subnet mask	255.255.255.128	0 + 128 = 128 (/25)
Subnet 1	192.168.100.0	Already consumed by LAN 1 + 2
Subnet 2	192.168.100.128	Assign to LAN 3

Table 2-10 Subnetting Challenge Table: LAN 3

Table 2-11 and Figure 2-29 present the final address allocation scheme for the network. LAN 1 and LAN 2 consume the first half of the 192.168.100.0/24 Class C address block with /26 subnets. The remaining addresses are assigned to LAN 3 with a larger /25 subnet.

	LAN 1	LAN 2	LAN 3
Network	192.168.100.0	192.168.100.64	192.168.100.128
Subnet Mask	255.255.255.192	255.255.255.192	255.255.255.128
First Address	192.168.100.1	192.168.100.65	192.168.100.129
Last Address	192.168.100.62	192.168.100.126	192.168.100.254
Broadcast	192.168.100.63	192.168.100.127	192.168.100.255

Table 2-11 Subnetting Challenge Table: Final

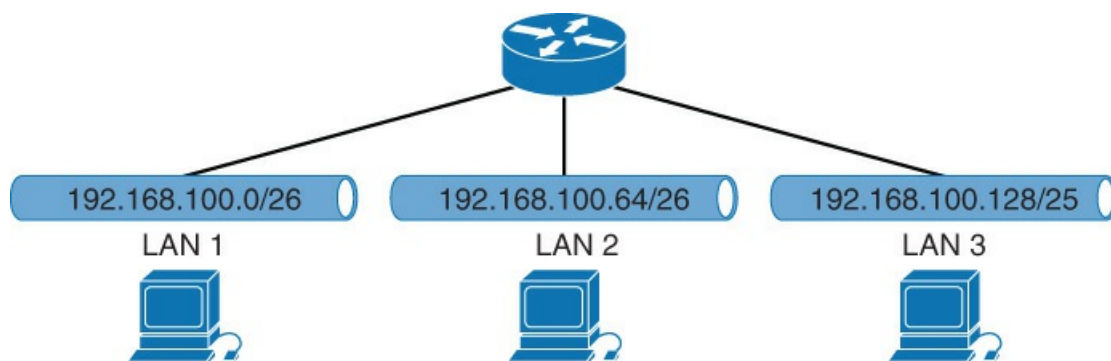


Figure 2-29 192.168.100.0/24: Completed Subnet Challenge Topology

## CLASSLESS INTERDOMAIN ROUTING

In 1993, RFCs 1517 through 1520 introduced classless interdomain routing (CIDR). CIDR provides for three very important scalability enhancements to IP: classless routing, variable-length subnet masks (VLSMs), and route summarization.

### Classful Versus Classless Routing

Earlier in this chapter, you learned how IP addresses are divided into classes based on the leading bit. Classes A, B, and C each have fixed network and host bit boundaries. In addition, subnetting may be used to subdivide the classful network into smaller subnets by borrowing bits from the hosts range and allocating them to the network ID. Unfortunately, there were still some limitations with implementing subnets due to the nature of classful routing.

### Classful Routing

In classful routing, the subnet mask is not included in the routing update. Therefore, the router receiving the update needs to make a local decision on how to apply a route mask for the IP address.

The classful routing local decision process is as follows:

**Step 1.** Assign the same mask to the route as what is configured on the interface that received the update.

**Step 2.** Apply a subnet mask that matches the classful network. For example, Class A is 255.0.0.0, Class B is 255.255.0.0, and Class C is 255.255.255.0.

The router will select choice 1 if the routing update is received on an interface that is configured with the same matching classful network. The router will select choice 2 if the route is on an interface that uses a different classful network. Routing Information Protocol Version 1 (RIPv1) and Interior Gateway Routing Protocol (IGRP) are examples of classful routing protocols.

The following example shows how classful routing makes it extremely difficult for network administrators to implement subnet masks. In [Figure 2-30](#), router R1 is advertising network 10.0.1.0 255.255.255.0.



**Figure 2-30** Classful Routing Update

Classful routing protocols do not include the subnet mask information in routing updates. [Figure 2-31](#) is a packet capture from router R1 and confirms that the subnet mask is not included in the routing update.

```

+ Frame 11: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
+ Ethernet II, Src: c0:00:03:a0:00:00 (c0:00:03:a0:00:00), Dst: Broadcast
+ Internet Protocol Version 4, Src: 10.0.12.1 (10.0.12.1), Dst: 255.255.2
+ User Datagram Protocol, Src Port: router (520), Dst Port: router (520)
- Routing Information Protocol
  Command: Response (2)
  Version: RIPv1 (1)
  - IP Address: 10.0.1.0, Metric: 1
    Address Family: IP (2)
    IP Address: 10.0.1.0 (10.0.1.0)
    Metric: 1

```

Classful routing protocol advertisements  
do not include subnet mask

Figure 2-31 Classful Routing Update (RIPv1)

Router R2 receives the advertisement for network 10.0.1.0 from router R1. R2 then compares the address in the update to the IP address on the receiving interface, Fast Ethernet 0/0. Fast Ethernet 0/0 is assigned IP address 10.0.12.2, which is in the same classful network as the update. R2 determines that the 10.0.1.0 address should be imported into the route table and assigned the same subnet mask as the receiving interface 255.255.255.0. The 10.0.1.0 network is advertised to R3 because the IP address of the outgoing interface is 10.0.23.2/24, which is part of the same Class A network (10.0.0.0/8). [Example 2-1](#) highlights that R1's 10.0.1.0 network is successfully added to R2's routing table.

### Example 2-1 R2's Routing Table

[Click here to view code image](#)

```

R2#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

10.0.0.0/24 is subnetted, 5 subnets
C       10.0.12.0 is directly connected, FastEthernet0/0
R       10.0.3.0 [120/1] via 10.0.23.3, 00:00:04, FastEthernet0/1
R       10.0.1.0 [120/1] via 10.0.12.1, 00:00:22, FastEthernet0/0
C       10.0.23.0 is directly connected, FastEthernet0/1
R       10.0.34.0 [120/1] via 10.0.23.3, 00:00:04, FastEthernet0/1

```

Router R2 and router R3 receive the routing advertisement for network 10.0.1.0. R2 does not assign a prefix or mask to routes 10.0.1.0 and 10.0.3.0. Again, the subnet masks for the attached interfaces are used when inserting the route into the route table.

Now observe what happens when the subnet masks are non-uniform across the network. In [Figure 2-32](#), the links between routers R1<>R2 and R2<>R3 have been modified to /30 prefixes.

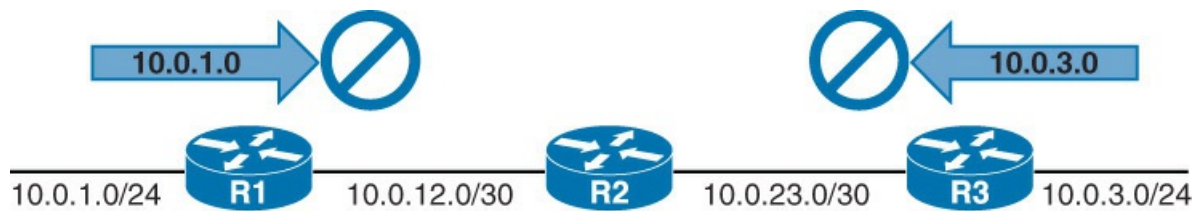


Figure 2-32 Classful Routing: Variable-Length Subnet Masks

The router interconnections are now using small 255.255.255.252 subnet masks to conserve IP addressing. Using small subnets on router point-to-point links is common in modern networks. When used in classful networks, however, nonuniform subnet masks can be very hazardous. The classful routing protocol on the edge routers, R1 and R3, will not advertise the locally attached /24 subnets out the /30 IP addressed point-to-point links. R1 and R3 are suppressing any advertisements for the attached /24 networks because they are not the same subnet mask length as the point-to-point connection to the upstream router R2. [Example 2-2](#) demonstrates that router R1 and R3 are not advertising the 10.0.1.0 or 10.0.3.0 networks to R2. R2's routing table only includes entries for directly attached networks.

### Example 2-2 Classful Routing: VLSM Broken Network

[Click here to view code image](#)

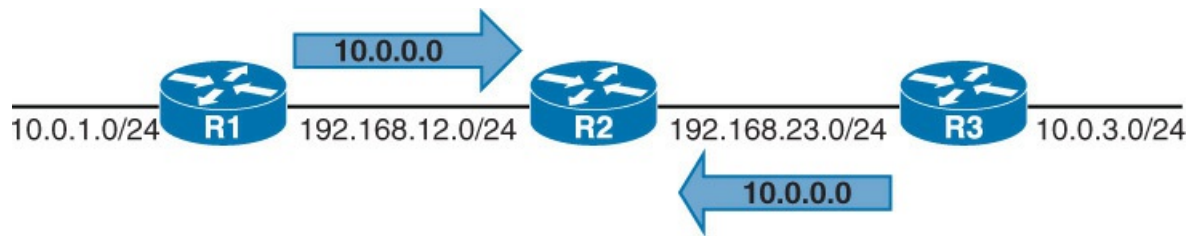
```
R2#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

10.0.0.0/30 is subnetted, 2 subnets
C       10.0.12.0 is directly connected, FastEthernet0/0
C       10.0.23.0 is directly connected, FastEthernet0/1
```

The key takeaway is that all subnet masks need to be the same length for full IP reachability in classful routing environments. If the masks are not the same, static and default routes are a necessity.

Now observe classful routing behavior when there are two subnets of the same network that are not contiguous. The router interconnections in the topology have been modified to use a different network addressing block. In [Figure 2-33](#), R1 and R2 are interconnected using the Class C network 192.168.12.0/24, and R2 and R3 are interconnected using Class C network 192.168.23.0/24. The Class A network 10.0.0.0/8 has been subnetted so that 10.0.1.0/24 is connected to R1 and 10.0.3.0/24 is connected to R3.



**Figure 2-33** Classful Routing: Discontiguous Network

The rules for classful routing dictate that router R1 must summarize the 10.0.1.0 subnet to a Class A network boundary before advertising the route to R2 because the interconnection between the two routers is on a different network. The route is modified to 10.0.0.0 because the Class A major network mask is 255.0.0.0. The same action is performed when the 10.0.3.0 network is advertised by router R3. This introduces a serious problem because there are now two equal-cost paths for the 10.0.0.0/8 network. [Example 2-3](#) demonstrates that router R2 router does not know whether R1 or R3 is the best path for 10.0.1.0.

### Example 2-3 Classful Routing Best Path Confusion

[Click here to view code image](#)

```
R2#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.12.0/24 is directly connected, FastEthernet0/0
R    10.0.0.0/8 [120/1] via 192.168.23.3, 00:00:04, FastEthernet0/1
      [120/1] via 192.168.12.1, 00:00:02, FastEthernet0/0
C    192.168.23.0/24 is directly connected, FastEthernet0/1
```

To further compound this problem, the edge routers R1 and R3 are not receiving any updates for each other's LAN subnets. As this example demonstrates, it is not possible to have discontinuous networks when using classful routing protocols and achieve full connectivity.

### Classless Routing

CIDR allows for subnets to be included in the routing update. RIPv2, Enhanced Interior Gateway Routing Protocol (EIGRP), Open Shortest Path First (OSPF), Intermediate System-to-Intermediate System (IS-IS) Protocol, and Border Gateway Protocol (BGP) are examples of classless routing protocols. In [Figure 2-34](#), R1 is advertising the network 10.0.1.0 255.255.255.0 and has disabled classful boundary auto summarization using the command **no auto-summary** under the routing process.





Figure 2-34 Classless Routing Update

A classless routing update includes both the network and subnet mask. The packet capture in Figure 2-35 confirms that router R1's routing protocol advertisement includes both the 10.0.1.0 network along with the subnet mask 255.255.255.0.

```

Frame 7: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
Ethernet II, Src: c0:00:18:bc:00:00 (c0:00:18:bc:00:00), Dst: IPv4mcast
Internet Protocol Version 4, Src: 192.168.12.1 (192.168.12.1), Dst: 224.
User Datagram Protocol, Src Port: router (520), Dst Port: router (520)
Routing Information Protocol
  Command: Response (2)
  Version: RIPv2 (2)
  IP Address: 10.0.1.0, Metric: 1
    Address Family: IP (2)
    Route Tag: 0
    IP Address: 10.0.1.0 (10.0.1.0)
    Netmask: 255.255.255.0 (255.255.255.0)
    Next Hop: 0.0.0.0 (0.0.0.0)
    Metric: 1
  
```

Classless routing protocol advertisements include network + mask

Figure 2-35 Route R1 Classless Routing Update (RIPv2)

The inclusion of the subnet mask has corrected router R2's view of the network. It correctly knows that it should traverse the path via R1 to reach the 10.0.1.0/24 network and the path via R3 to reach the 10.0.3.0/24 network. Example 2-4 highlights R2's routing table entries for network 10.0.3.0 and 10.0.1.0. The routing updates are properly being received from R1 and R3.

### Example 2-4 Classless Routing Best Path Calculation

[Click here to view code image](#)

```

R2#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.12.0/24 is directly connected, FastEthernet0/0
     10.0.0.0/24 is subnetted, 2 subnets
R       10.0.3.0 [120/1] via 192.168.23.3, 00:00:27, FastEthernet0/1
R       10.0.1.0 [120/1] via 192.168.12.1, 00:00:16, FastEthernet0/0
C    192.168.23.0/24 is directly connected, FastEthernet0/1
  
```

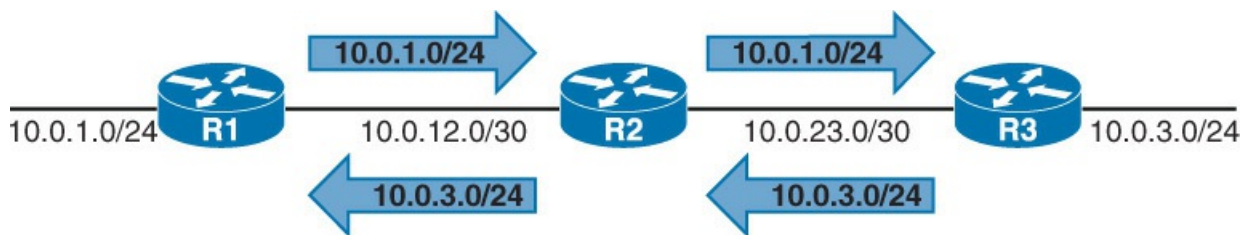
In today's modern networks, classless routing is the de facto standard. The IOS operating system

has used classless routing as the default mode of operation since the late 1990s with the 11.3 release. The IOS command to disable classless routing is **no ip classless**. In IOS XR, classless routing is the only mode supported.

### Variable-Length Subnet Masks

CIDR supports variable-length subnet masks (VLSMs), eliminating the need to maintain the same subnet mask length throughout the network. Network segments that do not require many IP addresses, such as a point-to-point router links, may be assigned a small subnet without concern for breaking the routing topology due to nonuniform subnet masks.

Let's reexamine the topology from [Figure 2-32](#), but this time instead of using the classful routing protocol, a classless routing protocol is deployed. In [Figure 2-36](#), R1 and R3 are no longer suppressing the local network advertisements to R2 because it does not matter that the point-to-point link to R2 has a different mask than the locally attached network.



**Figure 2-36** VLSM: Classless Routing Update

[Example 2-5](#) demonstrates that R2 has an accurate view of the network topology.

### Example 2-5 VLSM: Successful Route Convergence

[Click here to view code image](#)

```
R2#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

    10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C       10.0.12.0/30 is directly connected, FastEthernet0/0
R       10.0.3.0/24 [120/1] via 10.0.23.1, 00:00:27, FastEthernet0/1
R       10.0.1.0/24 [120/1] via 10.0.12.1, 00:00:20, FastEthernet0/0

C       10.0.23.0/30 is directly connected, FastEthernet0/1
```

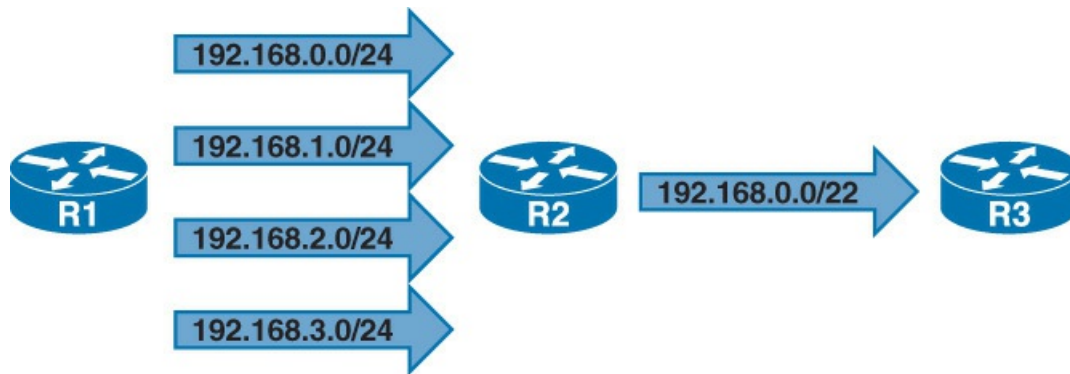
### Summarization

The last improvement introduced with CIDR is the ability to perform route summarization. Route summarization is combining many routes together so that they appear as one route in the route table.

In the early 1990s, there were legitimate concerns that the explosive growth of the Internet was going to overwhelm router resources because of the growing number of route table entries. To



reduce route table entries, the ability to perform route summarization was introduced. [Figure 2-37](#) demonstrates how route summarization can be used to reduce the size of the routing table for downstream routers.



**Figure 2-37** Summarization

Router R2's route table is displayed in the output of [Example 2-6](#). Router R1 is not performing route summarization, so the table includes all four networks: 192.168.0.0/24, 192.168.1.0/24, 192.168.2.0/24, and 192.168.3.0/24.

### Example 2-6 No Summarization

[Click here to view code image](#)

```
R2#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is 10.11.12.1 to network 0.0.0.0

    10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C       10.0.12.0/24 is directly connected, GigabitEthernet0/1
L       10.0.12.2/32 is directly connected, GigabitEthernet0/1
C       10.0.23.0/24 is directly connected, GigabitEthernet0/2
L       10.0.23.2/32 is directly connected, GigabitEthernet0/2
D       192.168.0.0/24 [90/130816] via 10.0.12.1, 00:00:47, GigabitEthernet0/1
D       192.168.1.0/24 [90/130816] via 10.0.12.1, 00:00:47, GigabitEthernet0/1
D       192.168.2.0/24 [90/130816] via 10.0.12.1, 00:00:47, GigabitEthernet0/1
D       192.168.3.0/24 [90/130816] via 10.0.12.1, 00:00:47, GigabitEthernet0/1
```

Router R2 performs route summarization and advertises only a single summary route for the 192.168.0.0/22 network to router R3. The summarization reduces the size of R3's route table. [Example 2-7](#) displays R1's route table.

### Example 2-7 Route Summarization

[Click here to view code image](#)

```
R3#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
D    10.0.12.0/24 [90/3072] via 10.0.23.2, 00:02:07, GigabitEthernet0/1
C    10.0.23.0/24 is directly connected, GigabitEthernet0/1
L    10.0.23.3/32 is directly connected, GigabitEthernet0/1
D    192.168.0.0/22 [90/131072] via 10.0.23.2, 00:00:18, GigabitEthernet0/1
```

Note

Route summarization may also be called hierarchical routing, aggregation, or supernetting.

To determine the summary prefix for route summarization, the networks are converted to binary and lined up on top of each other so that you can easily identify the common leading binary bits. These indicate the prefix that should be used to create the summary.

Figure 2-38 displays the four prefixes from Figure 2-37 in binary format. 192.168.0.0/22 is determined to be the summary prefix for the networks because the bit boundary is common across all four prefixes: 192.168.0.0, 192.168.1.0, 192.168.2.0, and 192.168.3.0.

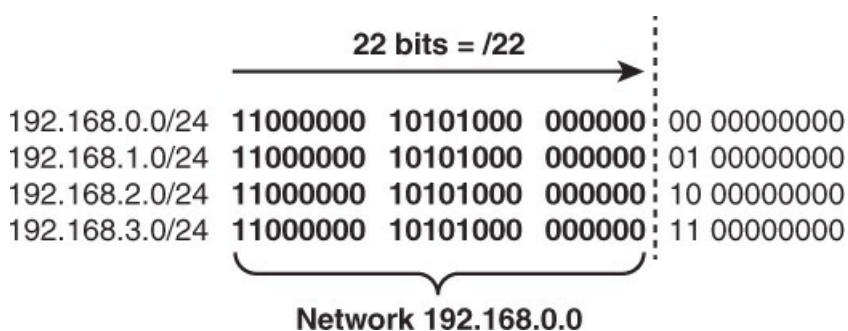


Figure 2-38 192.168.0.0/22 Bit Boundary

## PRIVATE IP ADDRESSING

In 1996, in an attempt to further slow down IP address exhaustion, the famous RFC 1918 was released that defines private address space. A private IP address is an address that is not routable on the global Internet. A private IP address is assigned locally within an organization, and because it is not shared with the global Internet community, there is not a risk of overlapping IP addresses.

Note

Private IP addresses behave like normal public IP addresses and are completely routable within an organization. Internet service providers (ISPs) filter the private IP address routes from being received or advertised so that they do not appear on the global route tables.

The Internet Assigned Numbers Authority (IANA) has reserved the following three IP blocks for private networks: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16. [Table 2-12](#) provides the start and end range for the private IPv4 address blocks.

<b>Network Prefix</b>	<b>Start</b>	<b>End</b>	<b>Total IP Addresses</b>
10.0.0.0/8	10.0.0.0	10.255.255.255	16,777,216
172.16.0.0/12	172.16.0.0	172.31.255.255	1,048,576
192.168.0.0/16	192.168.0.0	192.168.255.255	65,536

**Table 2-12** Private IPv4 Address Ranges

Note

Enterprises use Network Address Translation (NAT) to provide Internet connectivity to computers that are assigned a private IP address. NAT allows for one or many private IP addresses to share a globally routable IP address.

## SPECIAL IP ADDRESSES

The Internet Engineering Task Force (IETF) and the Internet Assigned Numbers Authority (IANA) have reserved certain IP addresses for special purposes. [Table 2-13](#) provides a list of the IANA IPv4 reserved address ranges, along with the RFC that documents the purpose of the reservation.

Address Block	Name	Routed on Internet	RFC	
0.0.0.0/8	“This host on this network”	No	RFC 1122	
10.0.0.0/8	Private-Use	No	RFC 1918	
100.64.0.0/10	Carrier-grade NAT	No	RFC 6598	
127.0.0.0/8	Loopback	No	RFC 1122	
169.254.0.0/16	Link Local	No	RFC 3927	
172.16.0.0/12	Private-Use	No	RFC 1918	
192.0.0.0/24	IETF Protocol Assignments	No	RFC 6890	
192.0.0.0/29	DS-Lite	No	RFC 6333	
192.0.0.170/32	192.0.0.171/32	NAT64/DNS64 Discovery	No	RFC-ietf-behave-nat64-discovery-heuristic-17
192.0.2.0/24	Documentation (Test-NET-1)	No	RFC 5737	
192.88.99.0/24	6to4 Relay Anycast	Yes	RFC 3068	
192.168.0.0/16	Private-Use	No	RFC 1918	
198.18.0.0/15	Benchmarking	No	RFC 2544	
198.51.100.0/24	Documentation (TEST-NET-2)	No	RFC 5737	
203.0.113.0/24	Documentation (TEST-NET-3)	No	RFC 5737	
224.0.0.0/4	Multicast	Yes	RFC 5771	
240.0.0.0/4	Reserved	No	RFC 1112	
255.255.255.255/32	Limited Broadcast	No	RFC 919	

Table 2-13 IPv4 Special-Purpose Addresses

## IPV4 ADDRESS CONFIGURATION

To enable IPv4 protocol stack processing on an interface, it must have an IP address assigned to it. An IP address, along with the subnet mask, is needed for the interface configuration.

Configuring an IP address is done with the command **ip address** *ip-address subnet-mask* on IOS nodes and with the command **ipv4 address** *ip-address subnet-mask* on IOS XR nodes, as demonstrated in [Example 2-8](#).

### Example 2-8 Assigning an IP Address to an Interface

[Click here to view code image](#)

```
IOS
interface GigabitEthernet0/1
 ip address 10.0.0.1 255.255.255.0
```

**IOS XR**

```
interface GigabitEthernet0/0/0/0
  ipv4 address 10.0.0.2 255.255.255.0
```

Optionally, it is possible to configure an IP address in IOS XR using the network mask prefix rather than the full subnet mask by using the command **ipv4 address ipv4-address prefix-length [secondary]**, as shown in [Example 2-9](#). Operationally, the two configuration methods are the same and achieve the same result.

### **Example 2-9** Assigning an IP Address to an Interface Using CIDR Notation

[Click here to view code image](#)

**IOS XR**

```
interface GigabitEthernet0/0/0/0
  ipv4 address 10.0.0.2/24
```

Cisco routers support assigning multiple addresses to a network interface. An interface may have one primary IP address and multiple *secondary addresses*. Secondary addresses are useful in many situations, such as when renumbering the network to allow for a graceful transitioning between address schemes.

Note

Control packets originating from the router always use the primary address.

Configuring secondary IP address is done by adding the suffix **secondary** after the IP address statement, as shown in [Example 2-10](#).

### **Example 2-10** Assigning a Secondary IP Address to an Interface

[Click here to view code image](#)

**IOS**

```
interface GigabitEthernet0/1
  ip address 10.0.0.1 255.255.255.0
  ip address 192.168.0.1 255.255.255.0 secondary
```

**IOS XR**

```
interface GigabitEthernet0/0/0/0
  ipv4 address 10.0.0.2 255.255.255.0
  ipv4 address 192.168.0.2 255.255.255.0 secondary
```

The IOS command **show ip interface [interface-type interface-number] [brief]** and the IOS XR command **show ipv4 interface [interface-type interface-number] [brief]** provide detailed Layer 3 interface configuration information along with the status of the IP protocol. [Example 2-11](#) demonstrates the output of the command.

## Example 2-11 Viewing Specific IPv4 Interfaces

[Click here to view code image](#)

### IOS

```
R1# show ip interface GigabitEthernet 0/1
GigabitEthernet0/1 is up, line protocol is up
  Internet address is 10.0.0.1/24
  Broadcast address is 255.255.255.255
  Address determined by setup command
  MTU is 1500 bytes
  Helper address is not set
  Directed broadcast forwarding is disabled
  Secondary address 192.168.0.1/24
  Outgoing access list is not set
  Inbound access list is not set
  Proxy ARP is enabled
  Local Proxy ARP is disabled
! Output omitted for brevity
```

### IOS XR

```
P/0/RSP0/CPU0:XR1#show ipv4 interface GigabitEthernet0/0/0/0
GigabitEthernet0/0/0/0 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 10.0.0.2/24
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound access list is not set
  Proxy ARP is disabled
! Output omitted for brevity
```

[Example 2-12](#) displays an IPv4 summary of each interface. Notice that only the primary IP address for each interface is reported, along with the operational status of the interface.

## Example 2-12 Viewing a Summary of All IPv4 Interfaces

[Click here to view code image](#)

### IOS

```
R1#show ip interface brief
Interface                IP-Address      OK? Method Status          Protocol
GigabitEthernet0/0      unassigned      YES NVRAM    administratively down down
GigabitEthernet0/1      10.0.0.1        YES NVRAM    Up              Up
GigabitEthernet0/2      unassigned      YES NVRAM    administratively down down
! Output omitted for brevity
```

## IOS XR

```
RP/1/RSP0/CPU0:XR1#show ip interface brief
```

Interface	IP-Address	Status	Protocol
MgmtEth0/RSP0/CPU0/0	unassigned	Shutdown	Down
MgmtEth0/RSP0/CPU0/1	unassigned	Shutdown	Down
GigabitEthernet0/0/0/0	unassigned	Shutdown	Down
GigabitEthernet0/0/0/1	10.0.0.2	Up	Up
GigabitEthernet0/0/0/2	unassigned	Shutdown	Down

```
! Output omitted for brevity
```

## WILDCARD SUBNET MASKS

Wildcard subnet masks are used to perform a matching condition against an IP address. They may be used to match a single host, subnet, or an entire range of networks. In this text, wildcard masks are used to reference subnets in access control lists (ACL) and during initial Interior Gateway Protocol (IGP) routing protocol configuration with the use of the **network** statement.

Wildcard masks are expressed in dot-decimal notation. The match condition is based on the binary value of the wildcard mask.

### Wildcard Mask Rules:

0 = Match

1 = Does not match or “do not care”

A wildcard mask can be considered the inverted form of a subnet mask. The network 192.168.50.0/24 is used to demonstrate the concept. [Figure 2-39](#) displays the IP address and subnet mask in binary format for network 192.168.50.0/24.

		Network ID			Host ID
IP Address	192.168.50.0	11000000	10101000	00110010	00000000
Subnet Mask	255.255.255.0	11111111	11111111	11111111	00000000

**Figure 2-39** 192.168.50.0/24 Subnet Mask

The bits of the subnet mask are flipped to create the wildcard subnet mask. [Figure 2-40](#) includes the wildcard mask for network 192.168.50.0/24 in binary format.

		Network ID			Host ID
<b>Subnet Mask</b>	255.255.255.0	11000000	10101000	00110010	00000000
<b>IP Address</b>	192.168.50.0	11000000	10101000	00110010	00000000
<b>Wildcard Mask</b>	0.0.0.255	00000000	00000000	00000000	11111111
		Exact Match			Wildcard (Match Any)

Figure 2-40 192.168.50.0/24 Wildcard Mask

The wildcard mask 192.168.50.0 0.0.0.255 indicates that the first 24 bits must exactly match (192.168.50). The remaining 8 bits may match any value (wildcard).

A simpler method for a network administrator to calculate the wildcard subnet mask is to perform dot-decimal octet subtraction. Every octet will have a value ranging from 0 to 255. To calculate the inverse value of a subnet mask, the subnet mask's octet value is subtracted from 255.

Figure 2-41 through Figure 2-43 demonstrate how to calculate wildcard masks for matching increasingly larger ranges of IP addresses.

$$\begin{array}{r}
 255 \quad 255 \quad 255 \quad 255 \\
 - 255 \quad . \quad 255 \quad . \quad 255 \quad . \quad 255 \quad . \quad 255 \quad \text{Subnet mask} \\
 \hline
 0 \quad . \quad 0 \quad . \quad 0 \quad . \quad 0 \quad \text{Wildcard mask}
 \end{array}$$

Figure 2-41 Match Single IP Host

Figure 2-41 demonstrates the wildcard math calculation for a single IP host 172.16.31.50. A host route has a subnet mask value of 255.255.255.255. To calculate the wildcard mask, the subnet mask is subtracted from the 4-byte 255 value. The resulting wildcard inverse match for the single IP host is 172.16.31.50 0.0.0.0.

Figure 2-42 demonstrates the wildcard math calculation for the network 172.16.31.0/24. A /24 prefix has a subnet mask of 255.255.255.0. To calculate the wildcard mask, the subnet mask is subtracted from the 4-byte 255 value. The resulting wildcard inverse match for the /24 network is 172.16.31.0 0.0.0.255.

$$\begin{array}{r}
 255 \quad 255 \quad 255 \quad 255 \\
 - 255 \quad . \quad 255 \quad . \quad 255 \quad . \quad 0 \quad \text{Subnet mask} \\
 \hline
 0 \quad . \quad 0 \quad . \quad 0 \quad . \quad 255 \quad \text{Wildcard mask}
 \end{array}$$

Figure 2-42 Match /24 Network

Figure 2-43 demonstrates the wildcard math calculation for the entire summary route 172.16.0.0/12. A /12 prefix has a subnet mask 255.240.0.0. To calculate the wildcard mask, the subnet mask is subtracted from the 4-byte 255 value. The resulting wildcard inverse match for the IP range is 172.16.0.0 0.15.255.255.



255	255	255	255	
255	. 240	. 0	. 0	<b>Subnet mask</b>
0	. 15	. 255	. 255	<b>Wildcard mask</b>

Figure 2-43 Match IP Range

## SUMMARY

Although IPv4 has been an unparalleled success and will no doubt be around for the foreseeable future, there are not enough available IP addresses to enable the continued growth of the Internet. The address-conservation techniques covered in this chapter, such as subnetting and private IP addressing, are slowing the depletion rate but will not be enough to prevent regional Internet registries (RIRs) from eventually running out of IP addresses. The “next generation” Internet protocol, IPv6, was developed to circumvent this problem. IPv6 avoids the address-exhaustion problem by vastly increasing the available address pool size from 4.3 billion to 340 undecillion! A complete coverage of IPv6 routing is covered in Chapter 18, “IPv6 Addressing.”

## REFERENCES IN THIS CHAPTER

IANA IPv4 Special-Purpose Address Registry, <http://www.iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml>

**RFC 791**, *Internet Protocol*, J. Postel, IETF, <http://www.ietf.org/rfc/rfc791.txt>, September 1981

**RFC 950**, *Internet Standard Subnetting Procedure*, J. Mogul, J. Postel, IETF, <http://www.ietf.org/rfc/rfc1985.txt>, August 1985

**RFC 1878**, *Variable Length Subnet Table For IPv4*, T. Pummill, B. Manning, IETF, <http://www.ietf.org/rfc/rfc1878.txt>, December 1995

**RFC 1517**, *Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR)*, R. Hinden, IETF, <http://tools.ietf.org/html/rfc1517>, September 1993

**RFC 1518**, *An Architecture for IP Address Allocation with CIDR*, Y. Rekhter, T. Li, IETF, <http://tools.ietf.org/html/rfc1518>, September 1993

**RFC 1519**, *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*, V. Fuller, T. Li, J. Yu, IETF, <http://tools.ietf.org/html/rfc1519>, September 1993

**RFC 1520**, *Exchanging Routing Information Across Provider Boundaries in the CIDR Environment*, Y. Rekhter, C. Topolcic, IETF, <http://tools.ietf.org/html/rfc1520>, September 1993

**RFC 4632**, *Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan*, V. Fuller, T. Li, IETF, <http://tools.ietf.org/html/rfc4632>, August 2006

**RFC 1918**, *Address Allocation for Private Internets*, Y. Rekhter, B. Moskowitz, D. Karrenberg, G.J. de Groot, E. Lear, IETF, <http://tools.ietf.org/html/rfc1918>, February 1996

Wendell Odom, *CCNA ICND Exam Certification Guide*. Indianapolis: Cisco Press, 2004. Print.

## Chapter 3. How a Router Works

This chapter covers the following topics:

- IP routing
- IP packet switching
- Planes of operation

The previous chapters described that a router is necessary to transmit packets between network segments. This chapter explains the process a router uses to accomplish this task. By the end of this chapter, you should have a good understanding of how a router performs IP routing and IP packet forwarding between different network segments.

### IP ROUTING

A router's primary function is to move an IP packet from one network to a different network. A router learns about nonattached networks through static configuration or through dynamic IP routing protocols.

Dynamic IP routing protocols distribute network topology information between routers and provide updates without intervention when a topology change in the network occurs. Design requirements or hardware limitations may restrict IP routing to static routes, which do not accommodate topology changes very well, and can burden network engineers depending on the size of the network. Routers try to select the best loop-free path in a network that forwards a packet to its destination IP address.

A network of interconnected routers and related systems managed under a common network administration is known as an *autonomous system*. The Internet is composed of thousands of autonomous systems spanning the globe.

The common dynamic routing protocols found in networks today are as follows:

- RIPv2 (Routing Information Protocol Version 2)
- EIGRP (Enhanced Interior Gateway Routing)
- OSPF (Open Shortest Path First) Protocol
- IS-IS (Intermediate System-to-Intermediate System) Protocol
- BGP (Border Gateway Protocol)

With the exception of BGP, the protocols in the preceding list are designed and optimized for routing within an autonomous system and are known as *internal gateway protocols* (IGPs). External gateway protocols (EGPs) route between autonomous systems. BGP is an EGP protocol but can also be used within an autonomous system. If BGP exchanges routes within an autonomous system, it is known as an *internal BGP* (iBGP) session. If it exchanges routes

between different autonomous systems, it is known as an *external BGP* (eBGP) session.

Figure 3-1 shows an illustration of how one or many IGPs as well as iBGP can be running within an autonomous system and how eBGP sessions interconnect the various autonomous systems together.

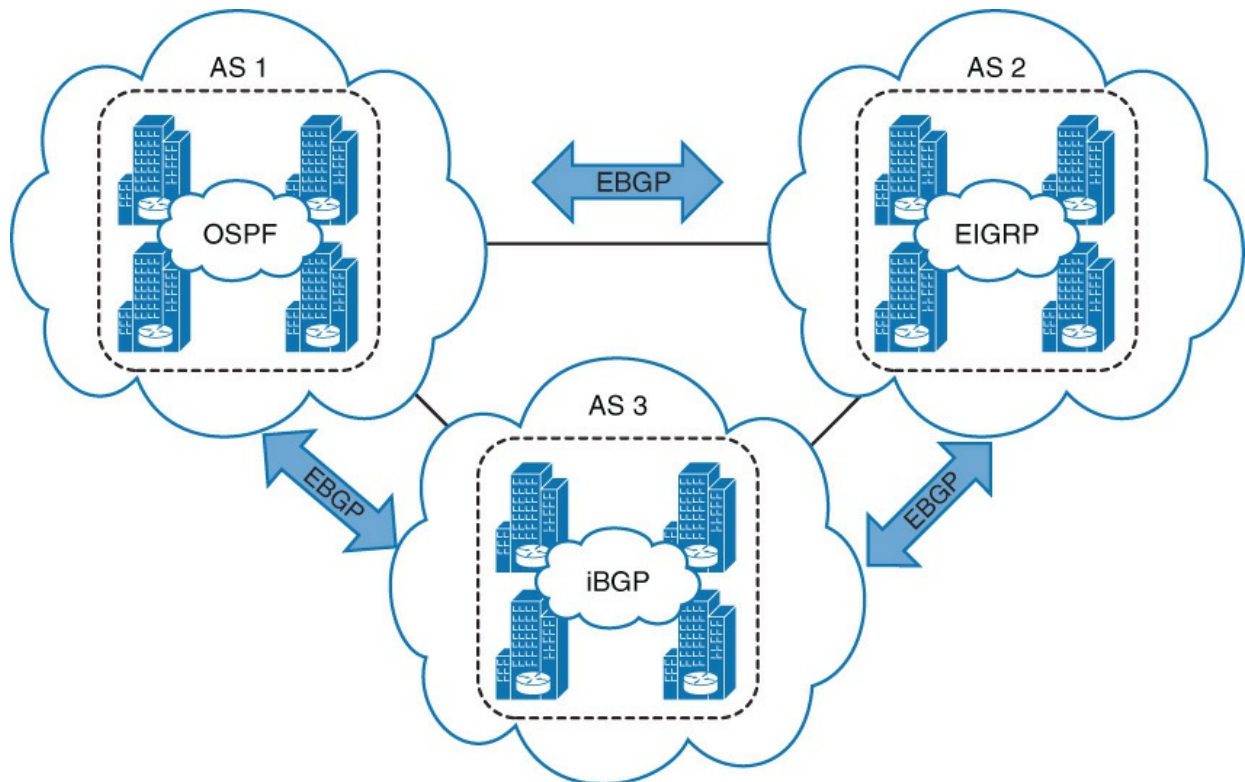


Figure 3-1 Autonomous Systems and How They Interconnect

EGPs and IGPs use different algorithms for path selection and are discussed in the following sections.

### Distance Vector Algorithms

Distance vector routing protocols, such as RIP, advertise routes as vectors (distance, vector), where distance is a metric (or cost) such as hop count and vector is the next-hop router's IP used to reach the destination:

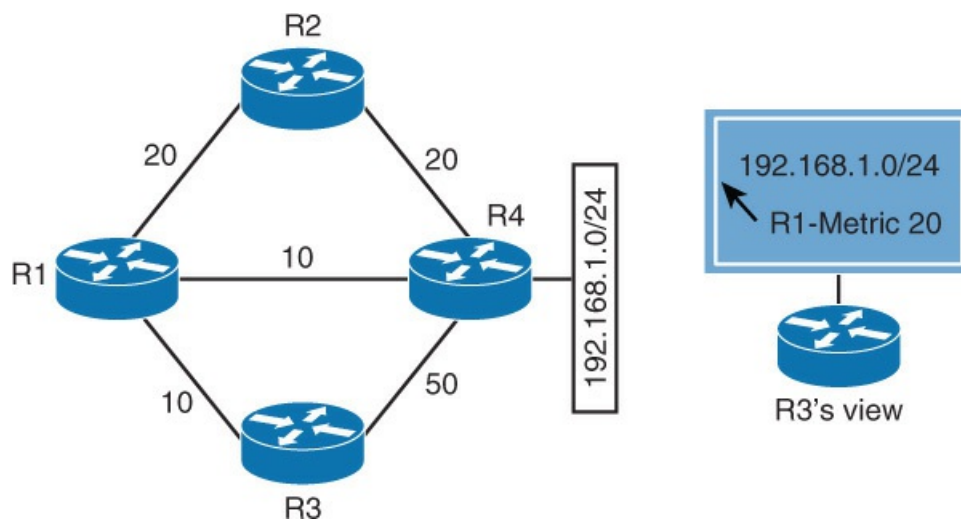
- **Distance:** The distance is the route metric to reach the network.
- **Vector:** The vector is the interface or direction to reach the network.

When a router receives routing information from a neighbor, it stores it in a local routing database as it is received and the distance vector algorithm (also known as *Bellman-Ford* and *Ford-Fulkerson* algorithms) is used to determine which paths are the best loop-free paths to each reachable destination. Once the best paths are determined, they are installed into the routing table and are advertised to each neighbor router.

Routers running distance vector protocols advertise the routing information to their neighbors from their own perspective, modified from the original route that it received. For this reason, distance vector protocols do not have a complete map of the whole network; instead, their database reflects that a neighbor router knows how to reach the destination network and how far

the neighbor router is from the destination network. They do not know how many other routers are in the path toward any of those networks. The advantage of distance vector protocols is that they require less CPU and memory and can run on low-end routers.

An analogy commonly used to describe distance vector protocols is that of a road sign at an intersection that indicates the destination is 20 miles to the west; this information is trusted and blindly followed, without really knowing whether there is a shorter or better way to the destination or if the sign is even correct. [Figure 3-2](#) illustrates how a router using a distance vector protocol views the network and the direction that R3 needs to go to reach the 192.168.1.0/24 subnet.



**Figure 3-2** Distance Vector Protocol View of the Network

### Enhanced Distance Vector Algorithm

The Diffused Update Algorithm (DUAL) is an enhanced distance vector algorithm that EIGRP uses to calculate the shortest path to a destination within a network. EIGRP advertises network information to its neighbors as other distance vector protocols do, but it has some enhancements as its name suggests. Some of the enhancements introduced into this algorithm compared to other distance vector algorithms are the following:

- Rapid convergence time for changes in the network topology.
- Only sends updates when there is a change in the network. It does not send full routing table updates in a periodic fashion like distance vector protocols.
- It uses hellos and forms neighbor relationships just like link-state protocols.
- It uses bandwidth, delay, reliability, load, and maximum transmission unit (MTU) size instead of hop count for path calculations.
- It has the option to load balance traffic across equal or unequal metric cost paths.

EIGRP is sometimes referred to as a *hybrid routing protocol* because it has characteristics of both distance vector and link-state protocols, as shown in the preceding list (for example, forming adjacencies with neighbor routers and relying on more advanced metrics such as bandwidth other than hop count for its best path calculations).

## Link-State Algorithms

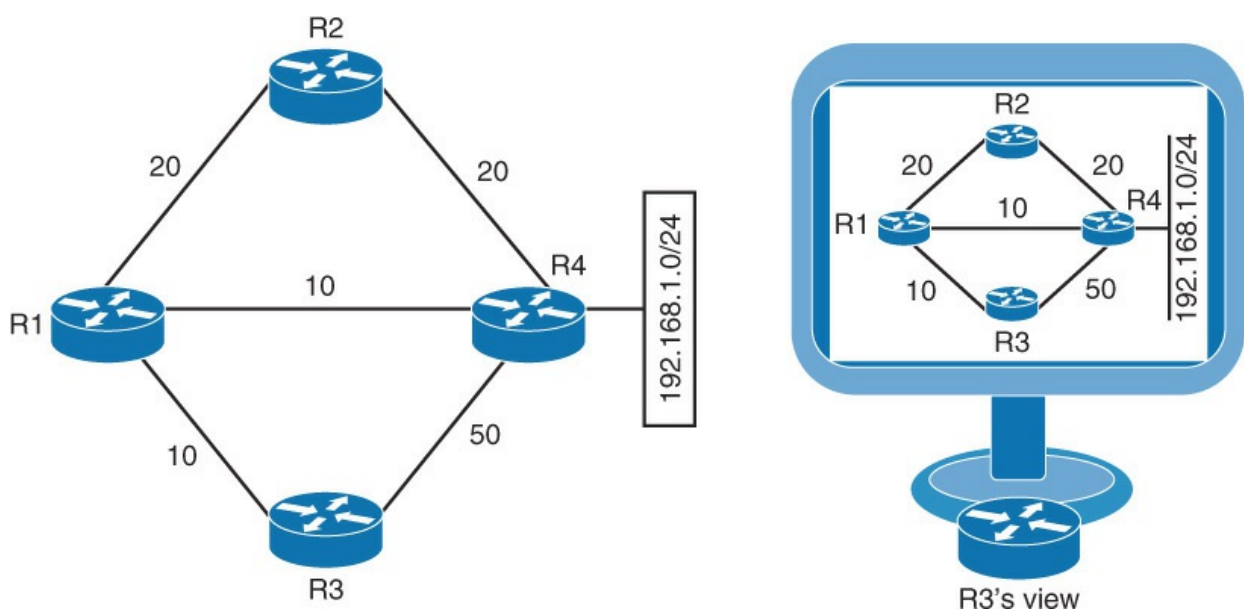
Link-state dynamic IP routing protocols advertise the link state and link metric for each of their connected links and directly connected routers to every router in the network. OSPF and IS-IS are two common link-state routing protocols found in enterprise and service provider networks. OSPF advertisements are called *link-state advertisements (LSAs)*, and IS-IS uses link-state packets (LSPs) for its advertisements.

As a router receives an advertisement from a neighbor, it stores the information in a local database called the *link-state database (LSDB)*, and advertises the link-state information on to each of its neighbor routers exactly as it was received. The link-state information is essentially flooded throughout the network from router to router unchanged, just as the originating router advertised it. This allows all the routers in the network to have a synchronized and identical map of the network.

Using the complete map of the network, every router in the network then runs the Dijkstra shortest path first (SPF) algorithm (developed by Edsger W. Dijkstra) to calculate the best shortest loop-free paths. The link-state algorithm then populates the routing table with this information.

Due to having the complete map of the network, link-state protocols usually require more CPU and memory than distance vector protocols, but they are less prone to routing loops and make better path decisions. In addition, link-state protocols are equipped with extended capabilities such as opaque LSAs for OSPF and TLVs (type/length/value) for IS-IS that allows them to support features commonly used by service providers such as MPLS traffic engineering.

An analogy for link-state protocols is a GPS navigation system. The GPS navigation system has a complete map and can make the best decision as to which way is the shortest and best path to reach the destination. [Figure 3-3](#) illustrates how R3 would view the network to reach the 192.168.1.0/24 subnet.



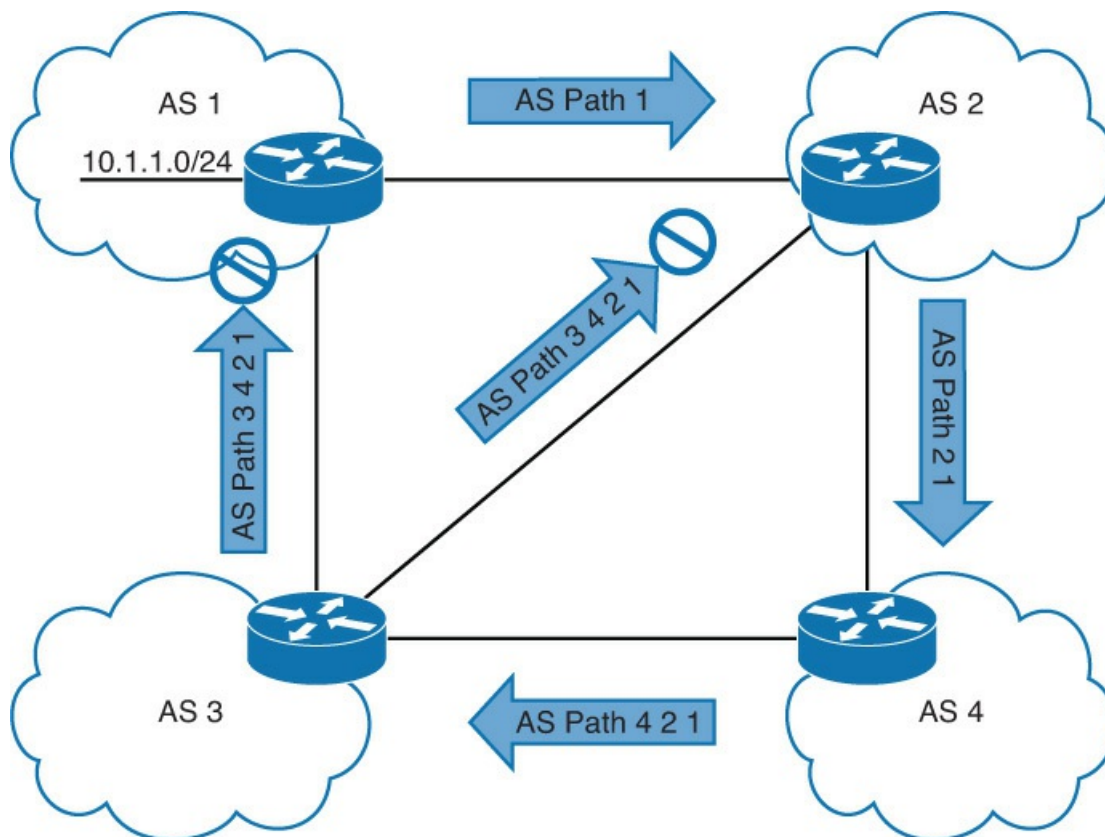
**Figure 3-3** Link-State Protocol View of the Network

## Path Vector Algorithm

A path vector protocol such as BGP is similar to a distance vector protocol; the difference is that instead of looking at the distance to determine the best loop-free path, it looks at various BGP path attributes. BGP path attributes include autonomous system path (AS\_Path), Multi-Exit Discriminator (MED), origin, next hop, local preference, atomic aggregate, and aggregator. BGP path attributes are covered in [Chapter 10, “BGP,”](#) and [Chapter 14, “Advanced BGP.”](#)

A path vector protocol guarantees loop-free paths by keeping a record of each autonomous system that the routing advertisement traverses. Any time a router receives an advertisement in which it is already part of the autonomous system path, the advertisement is rejected because accepting the autonomous system path would effectively result in a routing loop.

[Figure 3-4](#) illustrates this concept where autonomous system 1 advertises the 10.1.1.0/24 network to autonomous system 2. Autonomous system 2 receives this information and adds itself to the autonomous system path and advertises it to autonomous system 4. Autonomous system 4 adds itself to the path and advertises it to autonomous system 3. Autonomous system 3 receives the route advertisement and adds itself to the path as well. However, when autonomous system 3 advertises that it can reach 10.1.1.0/24 to autonomous system 1, autonomous system 1 discards the advertisement because the autonomous system path (path vector) contained in the advertisement includes its autonomous system number (autonomous system 1). When autonomous system 3 attempts to advertise reachability for 10.1.1.0/24 to autonomous system 2, autonomous system 2 also discards it because the advertisement includes autonomous system 2 in the autonomous system path, too.



**Figure 3-4** Path Vector Loop Avoidance

All BGP path attributes and how to manipulate them to influence the best path selection process are covered in [Chapter 15, “BGP Best Path Selection.”](#)



## Routing Table

A router identifies the path a packet should take by evaluating the following components on a router:

- **Prefix length:** The prefix length represents the number of leading binary bits in the subnet mask that are in the on position.
- **Administrative distance:** Administrative distance (AD) is a rating of the trustworthiness of a routing information source. If a router learns about a route to a destination from more than one routing protocol and they all have the same prefix length, AD is compared. The preference is given to the route with the lower AD.
- **Metrics:** A unit of measure used by a routing protocol in the best path calculation.

### Prefix Length

Let's look at a scenario of a router selecting a route when the packet destination is within the network range for multiple routes. Assume that a router has the following routes with various prefix lengths in the routing table:

- 10.0.3.0/28
- 10.0.3.0/26
- 10.0.3.0/24

Because each of these routes, also known as *prefix routes* or simply *prefixes*, has a different prefix length (subnet mask), they are considered to be different destinations, and they will all be installed into the routing table. This is represented in [Table 3-1](#).

Prefix	Subnet Range	Next Hop	Outgoing Interface
10.0.3.0/28	10.0.3.0 – 10.0.3.15	10.1.1.1	Gigabit Ethernet 1/1
10.0.3.0/26	10.0.3.0 – 10.0.3.63	10.2.2.2	Gigabit Ethernet 2/2
10.0.3.0/24	10.0.3.0 – 10.0.3.255	10.3.3.3	Gigabit Ethernet 3/3

**Table 3-1** Representation of Routing Table

If a packet needs to be forwarded, the route chosen depends on the prefix length, where the *longest prefix length* is always preferred. For example, /28 is preferred over /26, and /26 is preferred over /24. The following is an example using [Table 3-1](#) as a reference:

- If a packet needs to be forwarded to 10.0.3.14, it would match all three routes, but it would be sent to next hop 10.1.1.1 and outgoing interface Gigabit Ethernet 1/1 because 10.0.3.0/28 has the longest prefix match.
- If a packet needs to be forwarded to 10.0.3.42, it would match 10.0.3.0/24 and 10.0.3.0/26, so the packet would be sent to 10.2.2.2 and outgoing interface Gigabit Ethernet 2/2 because 10.0.3.0/26 has the longest prefix match.
- If a packet needs to be forwarded to 10.0.3.100, it matches only 10.0.3.0/24, so the packet is sent to 10.3.3.3 and outgoing interface Gigabit Ethernet 3/3.

## Administrative Distance

As each routing protocol receives updates and other routing information, it chooses the best path to any given destination and attempts to install this path into the routing table. Table 3-2 provides the default AD for the routing protocols covered in this book.

Routing Protocol	Default Administrative Distance
Connected	0
Static	1
eBGP	20
EIGRP summary route	5
EIGRP (internal)	90
OSPF	110
IS-IS	115
RIP	120
EIGRP (external)	170
iBGP	200

Table 3-2 Routing Protocol Default Administrative Distances

For example, if OSPF learns of a best path toward 10.0.1.0/24, it first checks to see whether an entry exists in the routing table. If it does not exist, the route is installed into the Routing Information Base (RIB). If the route already exists in the RIB, the router decides whether to install the route presented by OSPF based on the AD of the route in OSPF and the AD of the existing route in the RIB. If this route has the *lowest AD* to the destination (when compared to the other route in the table), it is installed in the routing table. If this route is not the route with the best AD, the route is rejected.

Consider another example on this topic. A router has OSPF, IS-IS, and EIGRP running, and all three protocols have learned of the destination 10.3.3.0/24 network with a different best path and metric.

Each of these three protocols will then attempt to install the route to 10.0.3.0/24 into the routing table. Because the prefix length is the same, the next decision point is the AD, where the routing protocol with the lowest AD installs the route into the routing table.

Because the EIGRP internal route has the best AD, it is the one installed into the routing table:

<b>10.0.3.0/24</b>	<b>EIGRP</b>	<b>90 &lt;&lt;&lt; Lowest AD Installed in Route Table</b>
10.0.3.0/24	OSPF	110
10.0.3.0/24	IS-IS	115

The routing protocol or protocols that failed to install their route into the table (in this example, that would be OSPF and IS-IS) will hang on to this route to use it as a backup route and will tell the routing table process to report to them if the best path fails so that they can then try to reinstall this route.



For example, if the EIGRP route 10.0.3.0/24 installed in the routing table fails for some reason, the routing table process calls OSPF and IS-IS, and requests them to reinstall the route in the routing table. Out of these two protocols, the preferred route is chosen based on AD, which would be OSPF because of its lower AD.

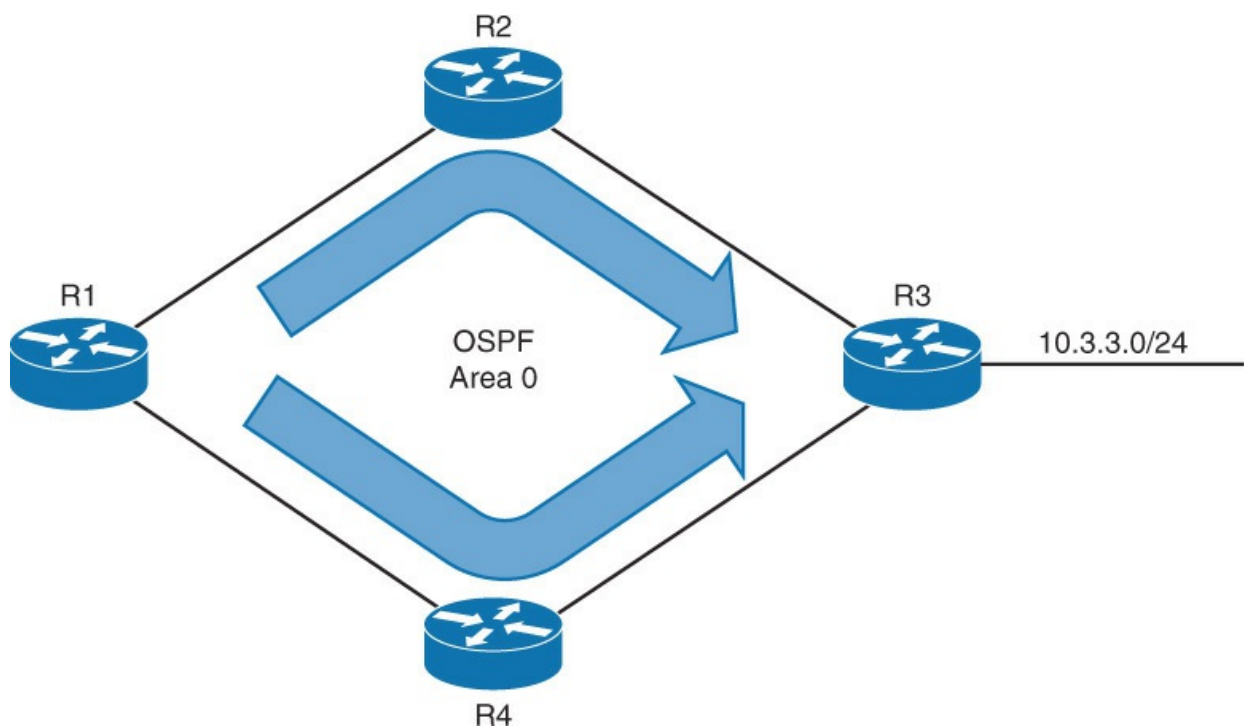
The default AD might not always be suitable for a network; for instance, there might be a requirement to adjust it so that OSPF routes are preferred over EIGRP routes. However, changing the AD on routing protocols can have severe consequences, such as routing loops and other odd behavior in a network. It is recommended that the AD be changed only with extreme caution, and only after what needs to be accomplished has been thoroughly thought out. A good backup plan is recommended in case things do not turn out as planned.

### Metrics

As discussed in the previous section, routes are chosen and installed into the routing table based on the routing protocol's AD. The routes learned from the routing protocol with the lowest AD are the ones installed into the routing table. If there are multiple paths to the same destination from a single routing protocol, these paths would have the same AD; for this case, the best path is selected within the routing protocol. Most protocols use the path with the best metric, but OSPF and IS-IS have additional logic that preempts the lowest metric.

If a routing protocol identifies multiple paths as a *best path*, and supports multiple path entries, the router installs the maximum number of paths allowed per destination. This is known as *equal-cost multipath* (ECMP) and provides load sharing across all links.

For example, [Figure 3-5](#) illustrates a network running OSPF to reach the prefix 10.3.3.0/24. Router 1 (R1) has two equal-cost paths; therefore, it will install both in the routing table.



**Figure 3-5** OSPF ECMP Technology

[Example 3-1](#) confirms that both paths have been installed into the RIB, and because the metrics are identical, this confirms the router is using ECMP.

### Example 3-1 R1's Routing Table Showing the ECMP Paths to 10.3.3.0/24

[Click here to view code image](#)

```
R1#show ip route
! Output omitted for brevity
O      10.3.3.0/24 [110/30] via 10.4.4.4, 00:49:12, GigabitEthernet0/0
                    [110/30] via 10.2.2.2, 00:49:51, GigabitEthernet0/2
```

#### Note

Best path metric calculation and the default and maximum ECMP paths allowed for each routing protocol vary. This is covered in later routing protocol-related chapters.

### Virtual Routing and Forwarding

Virtual Routing and Forwarding (VRF) is a technology that allows multiple independent virtual routing table and forwarding table instances to exist concurrently in a router. This can be leveraged to create segmentation between networks, which allows for overlapping IP addressing to be used even on a single interface (that is, using subinterfaces), and because the traffic paths are isolated, network security is increased and can eliminate the need for encryption and authentication for network traffic.

Service Providers with Multiprotocol Label Switching (MPLS) backbones typically use VRFs to create separate virtual private networks (VPNs) for their customers, and when used in this manner, VRFs are known as *VPN Routing and Forwarding*.

When VRF is not used in conjunction with MPLS, it is known as *VRF-Lite* (also termed *multi-VRF CE*, or *multi-VRF customer-edge device*). Because MPLS is beyond the scope of this book, only VRF-Lite is covered in this section and is referred to it simply as VRF.

The configurations in [Example 3-2](#) should help clarify the VRF concept. [Example 3-2](#) shows how configuring different interfaces with overlapping IP addresses and subnets is not allowed within a routing table, not even if they are both on different interfaces because they would end up in the same routing table and cause a conflict.

### Example 3-2 Overlapping IP Address Problems

[Click here to view code image](#)

```
IOS
R1 (config) #interface GigabitEthernet0/1
R1 (config-if) #ip address 10.0.3.1 255.255.255.0
R1 (config-if) #interface GigabitEthernet0/3
R1 (config-if) #ip address 10.0.3.2 255.255.255.0

% 10.0.3.0 overlaps with GigabitEthernet0/1
```

#### IOS XR

```
RP/0/0/CPU0:XR2 (config) #interface gigabitEthernet 0/0/0/5
RP/0/0/CPU0:XR2 (config-if) #ipv4 address 10.0.3.1/24
RP/0/0/CPU0:XR2 (config-if) #commit
RP/0/0/CPU0:XR2 (config-if) #
RP/0/0/CPU0:XR2 (config) #interface gigabitEthernet 0/0/0/3
RP/0/0/CPU0:XR2 (config-if) #ipv4 address 10.0.3.2/24
RP/0/0/CPU0:XR2 (config-if) #commit

RP/0/0/CPU0:Jan 13 18:55:35.643 : ipv4_arm[189]: %IP-IP_ARM-3-CFLCT_FORCED_DOWN :
The IPv4 address 10.0.3.1/24 on GigabitEthernet0/0/0/5 conflicts with other IPv4
addresses and has been forced down
```

#### Note

In IOS XR, the IP Address Repository Manager (IPARM) enforces the uniqueness of global IP addresses configured in the system. By default, when there is an IP address and subnet mask conflict, the lowest rack/slot/interface (that is, g0/0/0/3 is lower than g0/0/0/5) is the one that gets assigned the IP address. To change the default behavior, use the `ipv4 conflict-policy {static | highest-ip | longest-prefix}` command.

In older IOS releases, only single-protocol IPv4-only VRFs could be created. The command `ip vrf vrf-name` created a single-protocol VRF on the router and was activated on an interface with the command `ip vrf forwarding vrf-name` under the interface configuration mode.

In current IOS releases, a new configuration option allows the creation of multiprotocol VRFs that support both IPv4 and IPv6. Entering the command `vrf definition vrf-name` creates the multiprotocol VRF. Under VRF definition submode, the command `address-family {ipv4 | ipv6}` is required to specify the appropriate address family. The VRF is then associated to the interface with the command `vrf forwarding vrf-name` under the interface configuration submode.

#### Note

The commands `ip vrf vrf-name` and `ip vrf forwarding vrf-name` will be available for a period of time before they are deprecated. To migrate any older IPv4-only VRFs to the new multiprotocol VRF configuration, you can use the `vrf upgrade-cli multi-af-mode {common-policies | non-common-policies} [vrf vrf-name]` command. When creating a new VRF, even if it is just an IPv4-only VRF, Cisco recommends using the multiprotocol VRF `vrf definition` and `vrf forwarding` commands.

In IOS, the following steps are required to create a VRF and assign it to an interface:

### Step 1. Create a multiprotocol VRF.

The multiprotocol VRF routing table is created with the command `vrf definition vrf-name`.

### Step 2. Identify the address family.

Initialize the appropriate address family with the command `address-family {ipv4 | ipv6}`. The address family can be IPv4, IPv6, or both.

### Step 3. Specify the interface to be associated with the VRF.

Enter interface configuration submode and specify the interface to be associated with the VRF with the command **interface** *interface-type interface-number*.

#### Step 4. Associate the VRF to the interface.

The VRF is associated to the interface or subinterface by entering the command **vrf forwarding** *vrf-name* under interface configuration submode.

#### Step 5. Configure an IP address on the interface or subinterface.

The IP address can be IPv4, IPv6, or both. It is configured by entering the following commands:

[Click here to view code image](#)

```
IPv4  
ip address ip-address subnet-mask [secondary]
```

```
IPv6  
ipv6 address {ipv6-address/prefix-length | prefix-name sub-bits/  
prefix-length}
```

##### Note

On IOS nodes, the VRF needs to be associated to the interface first before configuring an IP address. If an IP address is already configured, and the VRF is associated to the interface, IOS will remove the IP address.

IOS XR supports only multiprotocol VRFs. The following steps are required to create a multiprotocol VRF and assign it to an interface on an IOS XR node:

#### Step 1. Create a multiprotocol VRF.

The multiprotocol VRF routing table is created with the command **vrf** *vrf-name*. The VRF name is arbitrary.

#### Step 2. Identify the address family.

Initialize the appropriate address family with the command **address-family** {**ipv4** | **ipv6**} **unicast**. The address family can be IPv4, IPv6, or both.

#### Step 3. Specify the interface to be associated with the VRF.

Enter interface configuration submode and specify the interface to be associated with the VRF with the command **interface** *interface-type interface-number*.

#### Step 4. Associate the VRF with an interface or subinterface.

The VRF is associated with the interface or subinterface by entering the command **vrf** *vrf-name* under interface configuration submode.

#### Step 5. Configure an IP address on the interface or subinterface.

The IP address can be IPv4, IPv6, or both. It is configured by entering the following commands:

[Click here to view code image](#)

IPv4  
**ipv4 address** *ipv4-address subnet-mask*

IPv6  
**ipv6 address** *ipv6-address/prefix-length*

Note

For IOS XR, the VRF needs to be associated to the interface first before configuring an IP address; otherwise, the VRF configuration will not be accepted.

Figure 3-6 illustrates two routers to help visualize the VRF routing table concept. One of the routers has no VRFs configured, and the other one has a management VRF named MGMT. This figure can be used as a reference for the following examples.

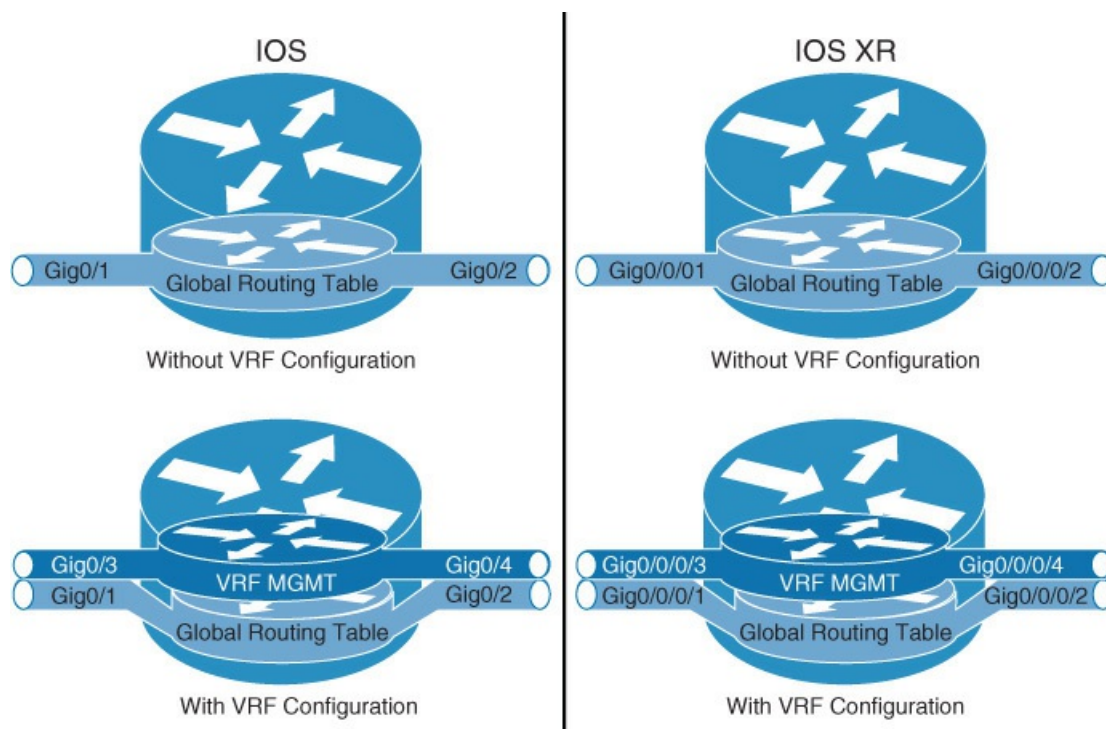


Figure 3-6 Comparison of a Router with no VRFs and a Router with a VRF

Table 3-3 provides a set of interfaces and IP addresses that overlap between the global routing table and the VRF. This information is used in the following examples.

IOS Interface	IOS XR Interface	IP Address	VRF	Global
Gigabit Ethernet 0/1	Gigabit Ethernet 0/0/0/1	10.0.3.1/24		✓
Gigabit Ethernet 0/2	Gigabit Ethernet 0/0/0/2	10.0.4.1/24		✓
Gigabit Ethernet 0/3	Gigabit Ethernet 0/0/0/3	10.0.3.1/24	MGMT	
Gigabit Ethernet 0/4	Gigabit Ethernet 0/0/0/4	10.0.4.1/24	MGMT	

Table 3-3 Sample Interfaces and IP Addresses

[Example 3-3](#) shows how the IP addresses are assigned to the interfaces in the global routing table shown in [Table 3-3](#).

### Example 3-3 IP Address Configuration in Global Routing Table

[Click here to view code image](#)

```
IOS
R1 (config) #interface GigabitEthernet0/1
R1 (config-if) #ip address 10.0.3.1 255.255.255.0
R1 (config) #interface GigabitEthernet0/2
R1 (config-if) #ip address 10.0.4.1 255.255.255.0
```

```
IOS XR
RP/0/0/CPU0:XR1 (config) #interface gigabitEthernet 0/0/0/1
RP/0/0/CPU0:XR1 (config-if) #ipv4 address 10.0.3.1/24
RP/0/0/CPU0:XR1 (config) #interface gigabitEthernet 0/0/0/2
RP/0/0/CPU0:XR1 (config-if) #ipv4 address 10.0.4.1/24
RP/0/0/CPU0:XR1 (config-if) #commit
```

[Example 3-4](#) displays the global routing table with the command **show ip route** for IOS and **show route** for IOS XR to show the IP addresses configured in [Example 3-3](#).

### Example 3-4 Output of Global Routing Table

[Click here to view code image](#)

```
IOS
R1#show ip route
! Output omitted for brevity
 10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C       10.0.3.0/24 is directly connected, GigabitEthernet0/1
L       10.0.3.1/32 is directly connected, GigabitEthernet0/1
C       10.0.4.0/24 is directly connected, GigabitEthernet0/2
L       10.0.4.1/32 is directly connected, GigabitEthernet0/2
```

```
IOS XR
RP/0/0/CPU0:XR1#show route
! Output omitted for brevity

C  10.0.3.0/24 is directly connected, 00:00:25, GigabitEthernet0/0/0/1
L  10.0.3.1/32 is directly connected, 00:00:25, GigabitEthernet0/0/0/1
C  10.0.4.0/24 is directly connected, 00:00:02, GigabitEthernet0/0/0/2
L  10.0.4.1/32 is directly connected, 00:00:02, GigabitEthernet0/0/0/2
```

[Example 3-5](#) shows how the VRF named MGMT is created, two interfaces are associated with it, and the IP addresses in [Table 3-3](#) are configured on the interfaces. These IP addresses overlap with the ones configured in [Example 3-3](#), but there is no conflict because they are in a different routing table.

## Example 3-5 VRF Configuration Example

[Click here to view code image](#)

### IOS

```
R1 (config) #vrf definition MGMT
R1 (config-vrf) # address-family ipv4
R1 (config) #interface GigabitEthernet0/3
R1 (config-if) #vrf forwarding MGMT
R1 (config-if) #ip address 10.0.3.1 255.255.255.0
R1 (config) #interface GigabitEthernet0/4
R1 (config-if) #vrf forwarding MGMT
R1 (config-if) #ip address 10.0.4.1 255.255.255.0
```

### IOS XR

```
RP/0/0/CPU0:XR1 (config) #vrf MGMT address-family ipv4 unicast
RP/0/0/CPU0:XR1 (config-vrf-af) #root
RP/0/0/CPU0:XR1 (config) #interface gigabitEthernet 0/0/0/3
RP/0/0/CPU0:XR1 (config-if) #vrf MGMT
RP/0/0/CPU0:XR1 (config-if) #ipv4 address 10.0.3.1/24
RP/0/0/CPU0:XR1 (config) #interface gigabitEthernet 0/0/0/4
RP/0/0/CPU0:XR1 (config-if) #vrf MGMT
RP/0/0/CPU0:XR1 (config-if) #ipv4 address 10.0.4.1/24
RP/0/0/CPU0:XR1 (config-if) #commit
```

[Example 3-6](#) shows how the VRF IP addresses configured in [Example 3-5](#) cannot be seen in the output of the **show ip route** command for IOS and the **show route** command for IOS XR; these commands display only the contents of the global routing table. To see a VRF routing table, the commands **show ip route vrf vrf-name** for IOS and **show route vrf {all | vrf-name}** for IOS XR should be used.

## Example 3-6 Output of Global Routing Table and VRF Routing Table

[Click here to view code image](#)

```
R1#show ip route
! Output omitted for brevity
 10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C       10.0.3.0/24 is directly connected, GigabitEthernet0/1
L       10.0.3.1/32 is directly connected, GigabitEthernet0/1
C       10.0.4.0/24 is directly connected, GigabitEthernet0/2
L       10.0.4.1/32 is directly connected, GigabitEthernet0/2

R1#show ip route vrf MGMT
! Output omitted for brevity
 10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C       10.0.3.0/24 is directly connected, GigabitEthernet0/3
L       10.0.3.1/32 is directly connected, GigabitEthernet0/3
C       10.0.4.0/24 is directly connected, GigabitEthernet0/4
L       10.0.4.1/32 is directly connected, GigabitEthernet0/4
```

```
RP/0/0/CPU0:XR1#show route
! Output omitted for brevity
C 10.0.3.0/24 is directly connected, 00:12:44, GigabitEthernet0/0/0/1
L 10.0.3.1/32 is directly connected, 00:12:44, GigabitEthernet0/0/0/1
C 10.0.4.0/24 is directly connected, 00:12:21, GigabitEthernet0/0/0/2
L 10.0.4.1/32 is directly connected, 00:12:21, GigabitEthernet0/0/0/2

RP/0/0/CPU0:XR1#show route vrf MGMT
! Output omitted for brevity
C 10.0.3.0/24 is directly connected, 00:09:15, GigabitEthernet0/0/0/3
L 10.0.3.1/32 is directly connected, 00:09:15, GigabitEthernet0/0/0/3
C 10.0.4.0/24 is directly connected, 00:00:10, GigabitEthernet0/0/0/4
L 10.0.4.1/32 is directly connected, 00:00:10, GigabitEthernet0/0/0/4
```

In IOS, to display a quick summary of the usability status for each IP interface, in addition to all the IP addresses configured in the global routing table and all VRFs, the command **show ip interface brief** should be used. In IOS XR, the command **show ipv4 interface brief** only shows the IP addresses in the global routing table. To see the IP addresses in the global routing table and all VRFs, use the command **show ipv4 vrf all interface brief**. [Example 3-7](#) provides sample output of these **show** commands.

### Example 3-7 Verification of Interfaces Status and IP Addresses

[Click here to view code image](#)

```
R1#show ip interface brief
```

Interface	IP-Address	OK?	Method	Status	Protocol
GigabitEthernet0/1	10.0.3.1	YES	NVRAM	up	up
GigabitEthernet0/2	10.0.4.1	YES	NVRAM	up	up
GigabitEthernet0/3	10.0.3.1	YES	NVRAM	up	up
GigabitEthernet0/4	10.0.4.1	YES	NVRAM	up	up

```
RP/0/0/CPU0:XR2#show ipv4 interface brief
```

Interface	IP-Address	Status	Protocol
GigabitEthernet0/0/0/0	unassigned	Shutdown	Down
GigabitEthernet0/0/0/1	10.0.3.1	Up	Up
GigabitEthernet0/0/0/2	10.0.4.1	Up	Up

```
RP/0/0/CPU0:XR2#show ipv4 vrf all interface brief
```

Interface	IP-Address	Status	Protocol	Vrf-Name
GigabitEthernet0/0/0/0	unassigned	Shutdown	Down	default
GigabitEthernet0/0/0/1	10.0.3.1	Up	Up	default
GigabitEthernet0/0/0/2	10.0.4.1	Up	Up	default
GigabitEthernet0/0/0/3	10.0.3.1	Up	Up	MGMT
GigabitEthernet0/0/0/4	10.0.4.1	Up	Up	MGMT

VRF-Lite can provide similar functionality to that of virtual local-area networks (VLANs); however, instead of relying on Layer 2 technologies such as spanning tree, Layer 3 dynamic



routing protocols can be used. Using routing protocols over Layer 2 technologies has some advantages such as improved network convergence times, dynamic traffic load sharing, and troubleshooting tools such as ping and traceroute.

## IP PACKET SWITCHING

Chapter 2, “IP Addressing,” explained that devices on the same subnet could communicate directly with each other without the need of a router. The second layer of the OSI model, the data link layer, handles addressing beneath the IP protocol stack so that communication is directed between hosts. Network packets include the Layer 2 addressing with unique source and destination addresses for that segment. Ethernet commonly uses MAC addresses, and other data link layer protocols such as Frame Relay use an entirely different method of Layer 2 addressing.

The first routers would receive a packet, remove the Layer 2 information, and verify that the route exists for the destination IP address. If a matching route could not be found, the packet was dropped. If a matching route was found, the router would identify it and add new Layer 2 information to the packet. The Layer 2 source address would be the router’s outbound interface, and the destination information would be next hop’s Layer 2 address.

Figure 3-7 illustrates the concept where PC A is sending a packet to PC B via Ethernet connection to R1. PC A sends the packet to R1’s MAC address of 00:C1:5C: 00:00:02. R1 receives the packet, removes the Layer 2 information, and looks for a route to the 192.168.2.2 address. R1 identifies that connectivity to the 192.168.2.2 IP address is through Gigabit Ethernet 0/1. R1 adds the Layer 2 source address using its Gigabit Ethernet 0/1’s MAC address 00:C1:5C:00:00:03 and a destination address for PC B of 00:00:00:00:00:04.

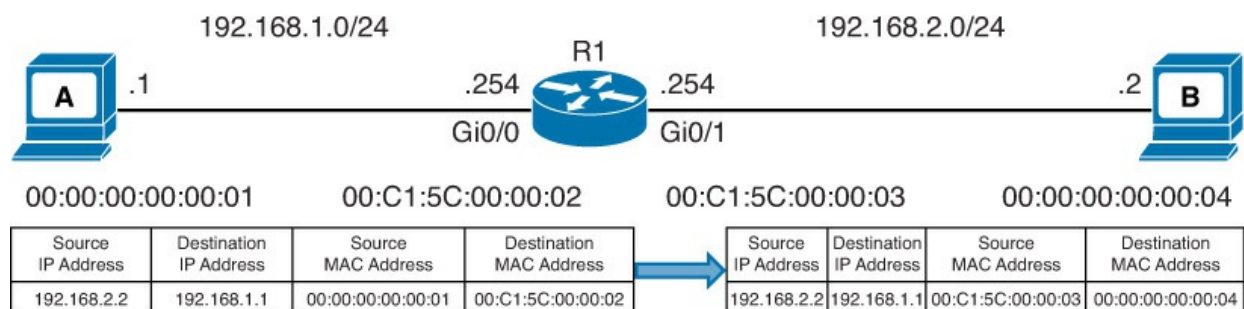


Figure 3-7 Layer 2 Addressing

Advancement in technologies has streamlined the process so that routers do not remove and add the Layer 2 addressing but simply rewrites them. IP packet switching or IP packet forwarding is the faster process of receiving an IP packet on an input interface and making a decision of whether to forward the packet to an output interface or drop it. This process is simple and streamlined for a router to be able to forward large amounts of packets.

When the first Cisco routers were developed, they used a mechanism called *process switching* to switch the packets through the routers. As network devices evolved, Cisco created Fast Switching and Cisco Express Forwarding (CEF) to optimize the switching process for the routers to be able to handle larger packet volumes. Fast Switching is deprecated in newer IOS releases and is not covered in this book.

## Process Switching

Process switching, also referred to as *software switching* or *slow path*, is the switching mechanism in which the general-purpose CPU on a router is in charge of packet switching. In IOS, the `ip_input` process runs on the general-purpose CPU for processing incoming IP packets. Process switching is the fallback for CEF because it is dedicated for processing punted IP packets when they cannot be switched by CEF.

In IOS XR, the Network Input/Output (NetIO) process is the equivalent to the IOS `ip_input` process and is responsible for forwarding packets in software.

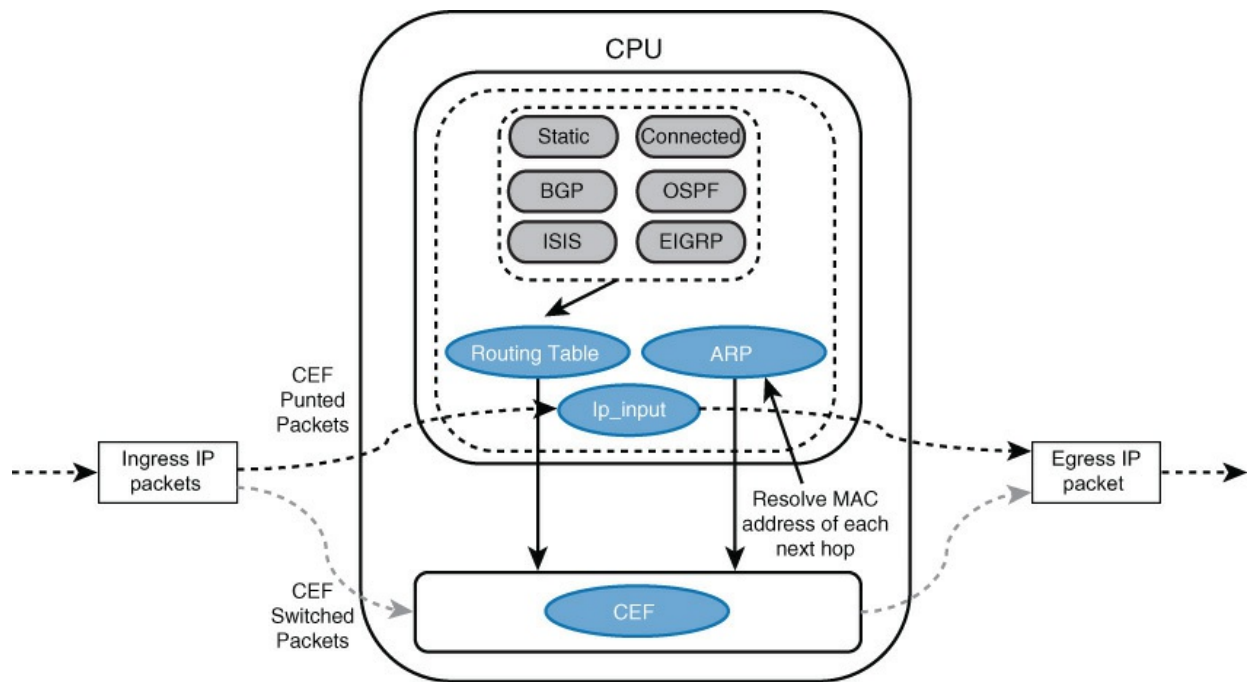
The type of packets that require software handling for both IOS and IOS XR include the following:

- Packets sourced or destined to the router (that is, control traffic, routing protocols)
- Packets that are too complex for the hardware to handle (that is, IP packets with IP options)
- Packets that require extra information that is not currently known (that is, Address Resolution Protocol [ARP] resolution, and so on)

### Note

Software switching is significantly slower than switching done in hardware. NetIO is designed to handle a very small percentage of traffic handled by the system. Packets are hardware switched whenever possible.

**Figure 3-8** illustrates how a packet that cannot be CEF switched is punted to the CPU for processing. The `ip_input` process consults the routing table and ARP table to obtain the next-hop router's IP address, outgoing interface, and MAC address. It then overwrites the destination MAC address of the packet with the next-hop router's MAC address, overwrites the source MAC address with the MAC address of the outgoing Layer 3 interface, decrements the IP Time-To-Live (TTL) field, recomputes the IP header checksum, and finally delivers the packet to the next-hop router.



**Figure 3-8** Process Switching

The routing table, also known as the *Routing Information Base* (RIB), is built from information obtained from dynamic routing protocols, directly connected and static routes. The ARP table is built from information obtained from the ARP protocol. The ARP protocol is used by IP hosts to dynamically learn the MAC address of other IP hosts on the same subnet. For example, an IP host that needs to perform address resolution for another IP host connected by Ethernet can send an ARP request using a LAN broadcast address, and it then waits for an ARP reply from the IP host. The ARP reply includes the required Layer 2 physical MAC address information.

### Cisco Express Forwarding

Cisco Express Forwarding (CEF) is a Cisco proprietary switching mechanism developed to keep up with the demands of evolving network infrastructures. It has been the default switching mechanism on most Cisco platforms that do all their packet switching using the general-purpose CPU (software based routers) since the 1990s, and it is the default switching mechanism used by all Cisco platforms that use specialized application specific integrated circuits (ASICs) and network processing units (NPUs) for high packet throughput (hardware-based routers).

The general-purpose CPU on the software-based and hardware-based routers is similar and perform all the same functions, the difference being that on software based routers the general-purpose CPU is in charge of all operations, including CEF switching (software CEF), and the hardware-based routers do CEF switching using *forwarding engines* that are implemented in specialized ASICs, TCAMs, and NPUs (hardware CEF). Forwarding engines provide the packet switching, forwarding, and route lookup capability to routers.

Given the low cost of the general-purpose CPUs, the price point of software-based routers will be much more affordable, but at the expense of total packet throughput.

When a route processor (RP) engine is equipped with a forwarding engine so that it can make all the packet switching decisions, this is known as a *centralized forwarding architecture*. If the line cards are equipped with forwarding engines so that they can make packet switching decision without intervention of the RP, this is known as a *distributed forwarding architecture*.

For a centralized forwarding architecture, when a packet is received on the ingress line card, it is transmitted to the forwarding engine on the RP. The forwarding engine examines the packet's headers and determines that the packet will be sent out a port on the egress line card, and forwards the packet to the egress line card to be forwarded.

For a distributed forwarding architecture, when a packet is received on the ingress line card, it is transmitted to the local forwarding engine. The forwarding engine performs a packet lookup, and if it determines that the outbound interface is local, it forwards the packet out a local interface. If the outbound interface is located on a different line card, the packet is sent across the switch fabric, also known as the *backplane*, directly to the egress line card, bypassing the RP.

Figure 3-9 illustrates a packet flowing across a centralized and a distributed forwarding architecture.

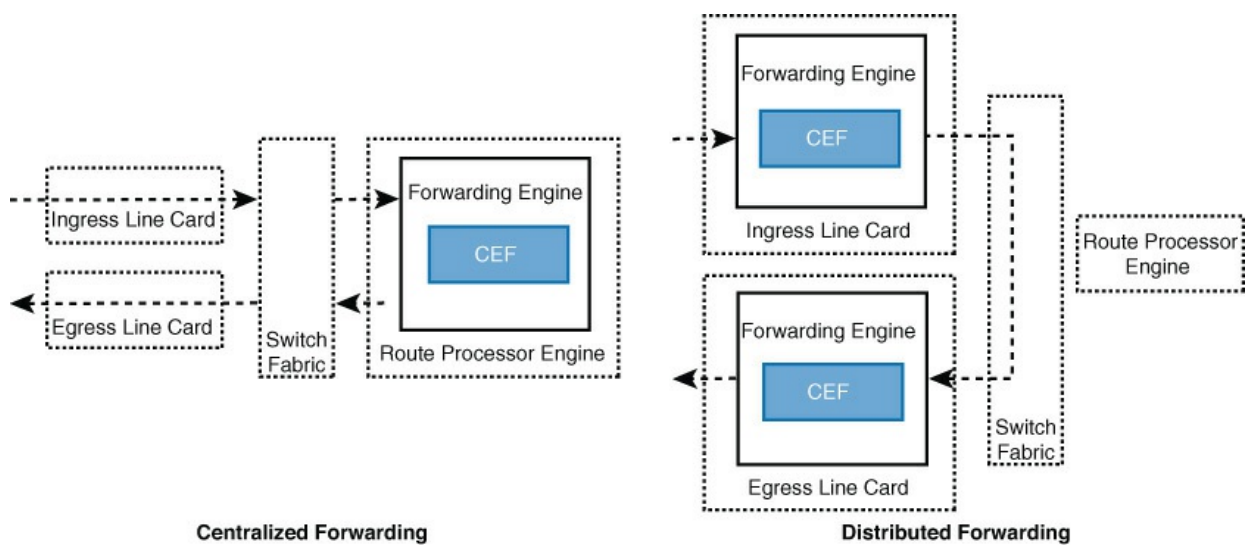


Figure 3-9 Centralized Versus Distributed Forwarding Architectures

### Software CEF

Software CEF, also known as the software *Forwarding Information Base* (FIB), consists of the following components:

- **Forwarding Information Base:** The FIB is built directly from the routing table and contains the next-hop IP address for each destination IP in the network. It keeps a mirror image of the forwarding information contained in the IP routing table. When a routing or topology change occurs in the network, the IP routing table is updated, and these changes are reflected in the FIB. CEF uses the FIB to make IP destination prefix-based switching decisions

- **Adjacency table:** The adjacency table is also known as the *Adjacency Information Base* (AIB). It contains the MAC addresses and egress interfaces of all directly connected next hops, and it is populated with data from the ARP table and other Layer 2 protocol tables (that is, Frame Relay map tables).

Figure 3-10 illustrates how the CEF table is built from the routing table and the ARP table and how a packet is CEF switched through the router. When an IP packet is received, if there is a valid FIB and adjacency table entry for it, the router overwrites the destination MAC address of the packet with the next hop router's MAC address, overwrites the source MAC address with the

MAC address of the outgoing Layer 3 interface, decrements IP TTL field, recomputes the IP header checksum, and finally delivers the packet to the next-hop router.

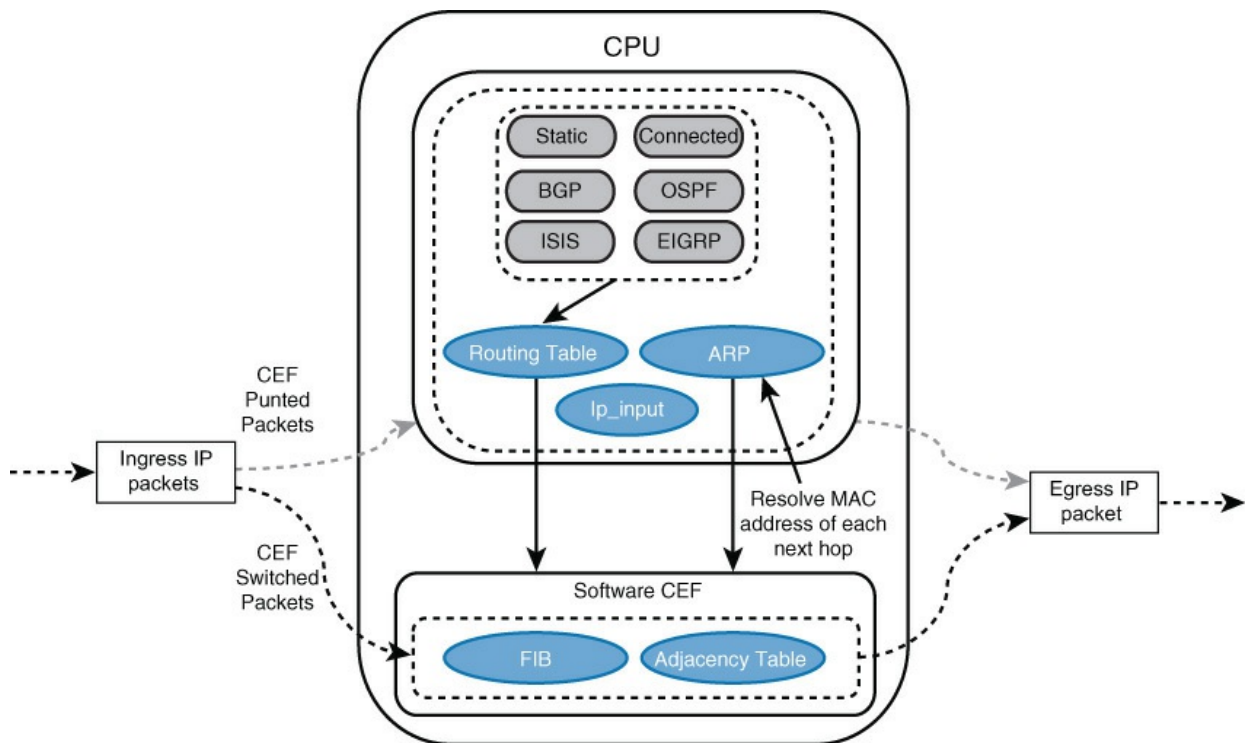


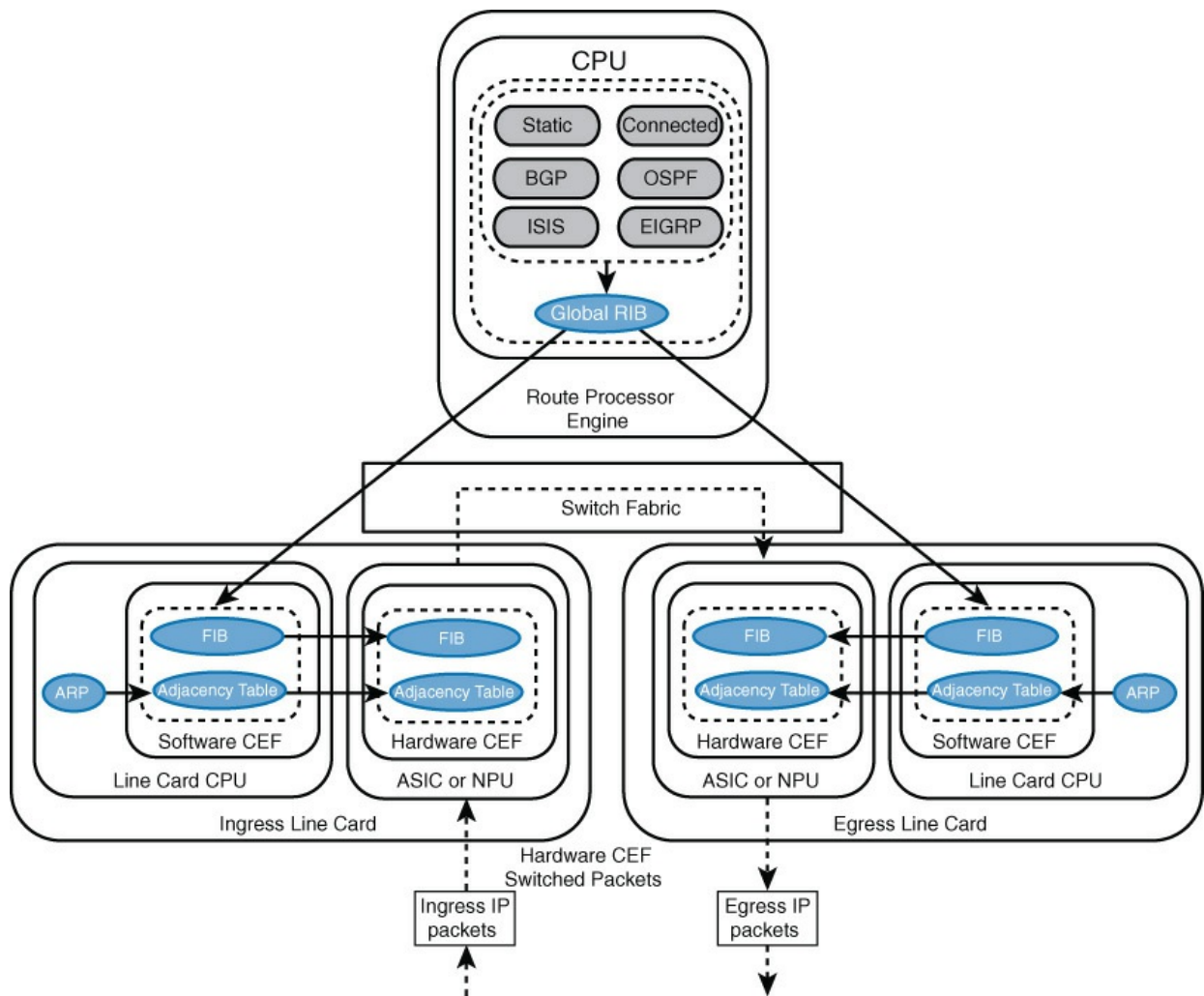
Figure 3-10 CEF Switching

### Hardware CEF

The ASICs in hardware-based routers have a very high cost to design, produce, and troubleshoot. ASICs allow for very high packet rates, but the trade-off is that they are limited in their functionality because they are hardwired to perform specific tasks. There are routers equipped with NPUs that are designed to overcome the inflexibility of ASICs. Unlike ASICs, NPUs are programmable, and their firmware can be changed with relative ease.

The main advantage of the distributed forwarding architectures is that the packet throughput performance is greatly improved by offloading the packet switching responsibilities to the line cards. Packet switching in distributed architecture platforms is done via distributed CEF (dCEF), which is a mechanism in which the CEF data structures are downloaded to forwarding ASICs and the CPUs of all line cards so that they can participate in packet switching; this allows for the switching to be done at the distributed level, thus increasing the packet throughput of the router.

Software CEF in hardware-based platforms is not used to do packet switching as in software-based platforms; instead, it is used to program the hardware CEF, as shown in Figure 3-11.



**Figure 3-11** *dCEF Hardware Switching*

Figure 3-11 also illustrates how the RIB process interacts with the RIBs of the routing protocols. The RIB process is in charge of the calculation of best paths, alternative paths, and the redistribution from different protocols and all these details merge into the global RIB (gRIB), where the best path for a destination network is installed. This is further distributed into the software CEF tables of different line cards, which is further mirrored into hardware CEF. The Switch Fabric is the backplane for all modules in the system. It creates a dedicated connection between all line cards and the route processors and provides fast data switching transmission between them.

In most distributed architecture platforms, if the incoming packet is control plane traffic or management traffic it is punted to the RP's CPU. The following list includes some examples of packets that are typically punted for processing by the RP's CPU or line card's CPU:

- Control traffic, such as BGP, OSPF, IS-IS, PIM, IGMP, and so on
- Management traffic, such as Telnet, SSH, SNMP, and so on
- Layer 2 mechanisms, such as CDP, ARP, LACP PDU, BFD, and so on
- Fragmentation, DF bit set, IP options set

- TTL expired

- ICMP echo request

## PLANES OF OPERATION

A router is typically segmented into three planes of operation, each with a specific and clearly defined objective:

- **The control plane:** The control plane is the brain of the router. It consists of dynamic IP routing protocols (that is OSPF, IS-IS, BGP, and so on), the RIB, routing updates, in addition to other protocols such as PIM, IGMP, ICMP, ARP, BFD, LACP, and so on. In short, the control plane is responsible for maintaining sessions and exchanging protocol information with other router or network devices.

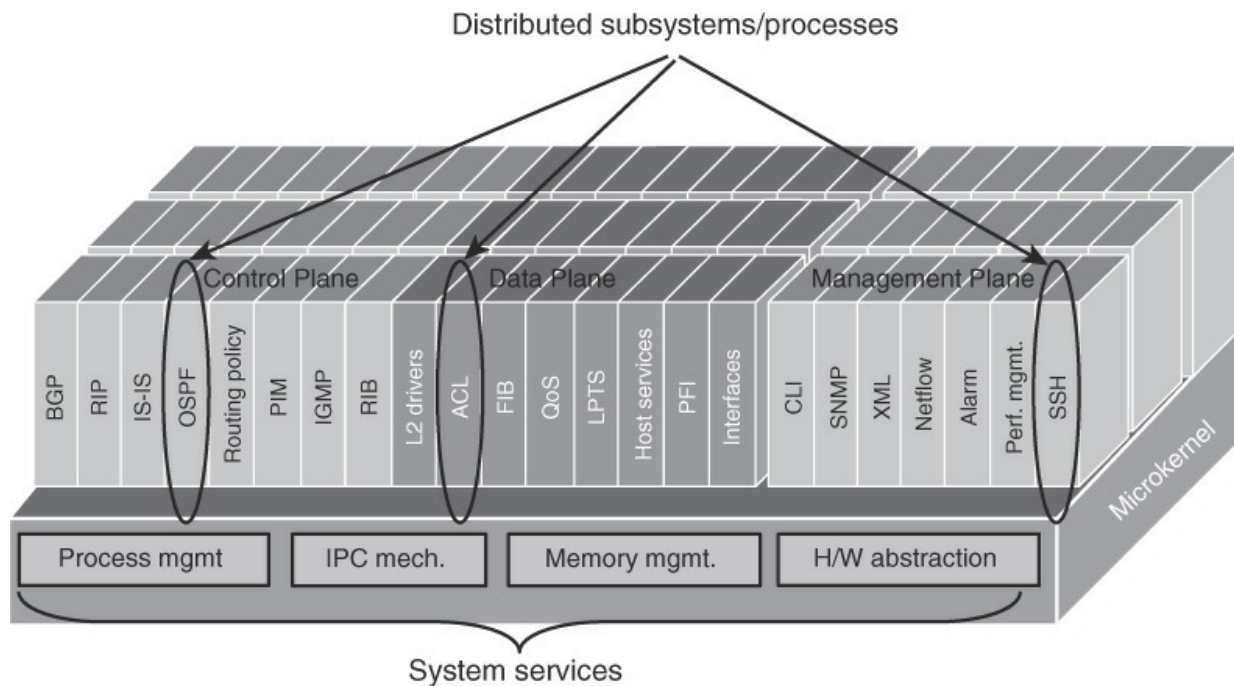
In centralized architecture platforms, the general-purpose CPU manages all control plane protocols. In distributed architecture platforms, routing protocols, and most other protocols, always run on the core CPU in the RPs or Supervisor engines, but there are other control plane protocols such as ARP, BFD, and ICMP that in some distributed architecture platforms have now been offloaded to the line card CPU.

- **The data plane:** The data plane is the forwarding plane, which is responsible for the switching of packets through the router (that is, process switching and CEF switching). In the data plane, there could be features that could affect packet forwarding such as quality of service (QoS) and access control lists (ACLs).

- **The management plane:** The management plane is used to manage a device through its connection to the network. Examples of protocols processed in the management plane include Simple Network Management Protocol (SNMP), Telnet, File Transfer Protocol (FTP), Secure FTP, and Secure Shell (SSH). These management protocols are used for monitoring and for command-line interface (CLI) access.

Figure 3-12 shows how the three planes of operation and how the processes are isolated from each other. In IOS XR, a process failure within one plane does not affect other processes or applications within that plane. This layered architecture creates a more reliable model than one with a monolithic architecture such as IOS, where failure of a single process may cause a failure of the whole system.





**Figure 3-12** Separation of Control, Data, and Management Planes

## SUMMARY

This chapter provided an overview of the fundamentals of IP routing and IP switching and the control planes of operation. In summary, it showed how a router makes a forwarding decision, which consists of three basic components:

- The routing protocols, which are used to build the routing table (RIB)
- The routing table, which is used to program the switching mechanisms (that is, CEF)
- The switching mechanisms used to perform the actual packet forwarding

## REFERENCES IN THIS CHAPTER

Bollapragada, Vijay, Russ White, and Curtis Murphy. *Inside Cisco IOS Software Architecture*. (ISBN-13: 978-1587058165).

Stringfield, Nakiya, Russ White, and Stacia McKee. *Cisco Express Forwarding*. (ISBN-13: 978-0-13-343334-0).

Tahir, Mobeen, Mark Ghattas, Dawit Birhanu, and Syed Natif Nawaz. *Cisco IOS XR Fundamentals*. (ISBN-13: 978-1-58705-271-2).

Doyle, Jeff and Jennifer Carroll. *Routing TCP/IP, Volume 1*, 2nd Ed. (ISBN-13: 978-1-58705-202-6).



## **Part II: Routing Protocols**

## Chapter 4. Static Routing

This chapter covers the following topics:

- Connected networks
- Static route fundamentals
- Floating static routes
- Multihop static routes
- Equal-cost multipath routing
- Null interfaces
- VRFs

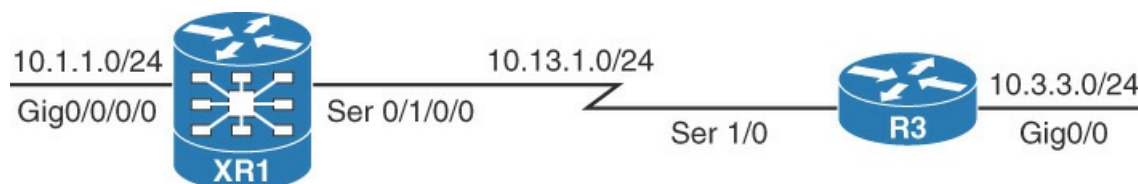
A router natively provides connectivity for networks between its interfaces. The number of interfaces and IP addresses on a router limits this basic form of routing. Providing connectivity to networks on another router requires static or dynamic routing.

This chapter covers the nondynamic aspect of routing, such as connected and static routes. Static routing provides precise control over the routing entry through manual configuration. This chapter discusses the benefits and drawbacks of using static routes along with the techniques to implement them.

### CONNECTED NETWORKS

Assigning an IP address to an interface in the *up state* will inject the associated network into the router's routing table (Routing Information Base [RIB]). The locally attached network is referred to as a *connected network*. Connected networks or routes have an administrative distance (AD) of zero. It is not possible for any other routing protocol to preempt a connected route in the RIB.

Figure 4-1 demonstrates a simple topology between router XR1 and router R3. XR1's connected networks are 10.1.1.0/24 and 10.13.1.0/24. R3's connected networks are 10.13.1.0/24 and 10.3.3.0/24.



**Figure 4-1** Sample IOS and IOS XR Topology

Example 4-1 displays XR1's and R3's routing table. Notice that the connected routes do not report an AD or metric.

**Example 4-1** Routing Table for Sample Topology

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
```

```
Codes: C - connected, S - static, R - RIP, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
U - per-user static route, o - ODR, L - local, G - DAGR
A - access/subscriber, (!) - FRR Backup path
```

```
Gateway of last resort is not set
```

```
C 10.1.1.0/24 is directly connected, 00:44:39, GigabitEthernet0/0/0/0
L 10.1.1.1/32 is directly connected, 00:44:39, GigabitEthernet0/0/0/0
C 10.13.1.0/24 is directly connected, 00:44:39, Serial 0/1/0/0
L 10.13.1.1/32 is directly connected, 00:44:39, Serial 0/1/0/0
```

```
R3#show ip route
```

```
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C 10.3.3.0/24 is directly connected, GigabitEthernet0/0
L 10.3.3.3/32 is directly connected, GigabitEthernet0/0
C 10.13.1.0/24 is directly connected, GigabitEthernet0/0
L 10.13.1.3/32 is directly connected, GigabitEthernet0/0
```

Notice the other routes with the *L* beside them to indicate that they are *local routes*. Local routes are explicit /32 (IPv4) and /128 (IPv6) routes referencing the IP addresses configured for that interface. Local routes are necessary for programming the Cisco Express Forwarding (CEF) as *receive* entries.

The AD and metric for a route is verifiable by examining the specific network entry in the routing table. The command **show ip route network [subnet-mask]** shows the specific network entry on IOS routers, and IOS XR uses the equivalent command **show route network [subnet-mask | /prefix-length]**.

If the subnet mask or prefix length is omitted, IOS and IOS XR will identify a shorter match (more vague) if the network does not exist in the routing table. If the subnet mask or prefix length is included, IOS and IOS XR nodes will output “%Network not in table” if the network does not exist in the routing table.

Example 4-2 provides output of a specific network entry for a connected network.

### Example 4-2 Detailed IP Routing Detail Information for a Specific Network

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route 10.1.1.0
Routing entry for 10.1.1.0/24
  Known via "connected", distance 0, metric 0 (connected)
  Routing Descriptor Blocks
    directly connected, via GigabitEthernet0/0/0/0
      Route metric is 0
  No advertising protos.
```

```
R3#show ip route 10.3.3.0
Routing entry for 10.3.3.0/24
  Known via "connected", distance 0, metric 0 (connected, via interface)
  Routing Descriptor Blocks:
    * directly connected, via GigabitEthernet0/0
      Route metric is 0, traffic share count is 1
```

#### Note

In Figure 4-1, XR1 does not have a method to connect to the 10.3.3.0/24 network, and R3 does not have a method to connect to the 10.1.1.0/24 network. A static route on each router is required for bidirectional connectivity.

## SECONDARY CONNECTED NETWORKS

Secondary IP addresses on an interface operate identically to the primary IP address in programming the RIB. *Secondary connected networks* are the networks associated with the secondary IP addresses.

Figure 4-2 expands on the previous topology by adding secondary IP addresses to router XR1 and R3's Ethernet interfaces.

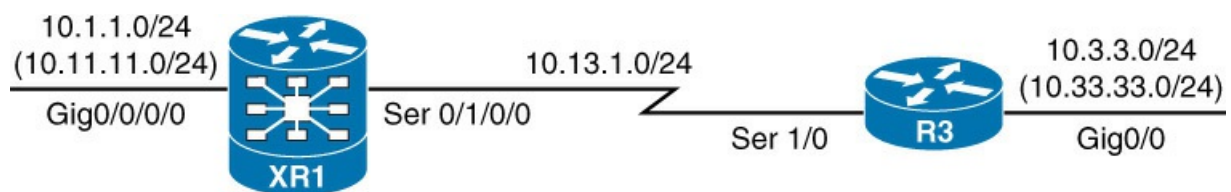


Figure 4-2 Sample IOS and IOS XR Topology with Secondary IP Addresses

Example 4-3 confirms that connected network routes align with the configured secondary IP addresses.

### Example 4-3 Routing Table for XR1 and R3

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
! Output omitted for brevity
```

Gateway of last resort is not set

```
C 10.1.1.0/24 is directly connected, 01:30:07, GigabitEthernet0/0/0/0
C 10.11.11.0/24 is directly connected, 00:02:49, GigabitEthernet0/0/0/0
C 10.13.1.0/24 is directly connected, 01:30:07, Serial 0/1/0/0
```

```
R3#show ip route connected
! Output omitted for brevity
```

Gateway of last resort is not set

```
10.0.0.0/8 is variably subnetted, 5 subnets, 3 masks
C 10.3.3.0/24 is directly connected, GigabitEthernet0/0
C 10.13.1.0/24 is directly connected, GigabitEthernet 0/0
C 10.33.33.0/24 is directly connected, GigabitEthernet0/0
```

## STATIC ROUTING FUNDAMENTALS

Static routes provide precise control over routing but may be an administrative burden as the number of routers and network segments grow. Using static routing requires zero network bandwidth because implementing manual route entries does not require communicating with other routers.

Unfortunately, because the routers are not communicating, there is zero network intelligence. If a link goes down, other routers will not be aware that the network path is no longer valid. Static routes are useful when

- Dynamic routing protocols cannot be used on a router because of limited router CPU or memory.
- Routes learned from dynamic routing protocols need to be superseded.
- The default AD for a static route is 1.

Table 4-1 outlines the benefits as well as the drawbacks of implementing static routing.

Static Route Benefits	Static Route Drawbacks
Easy to configure	Scalability
Predictable	Administrative burden
Zero network bandwidth overhead	Lack of network intelligence that could introduce suboptimal routing or black-holing of traffic

Table 4-1 Static Route Benefits and Drawbacks

Within the RIB, a static route identifies the outbound path or interface that will be used when the router forwards packets to the destination network. Static routes are classified as either

- Directly attached static routes

- Recursive static route
- Fully specified static route

Figure 4-3 demonstrates the concept of a static route on R1 for the 10.23.1.0/24 network. R1 knows that it must send a packet to the IP address 10.12.1.2 out of the Gi0/0 interface. R1 does not know how many hops it will take to get to the 10.23.1.0/24 network, but it does know the outbound interface to reach the 10.23.1.0/24 network.

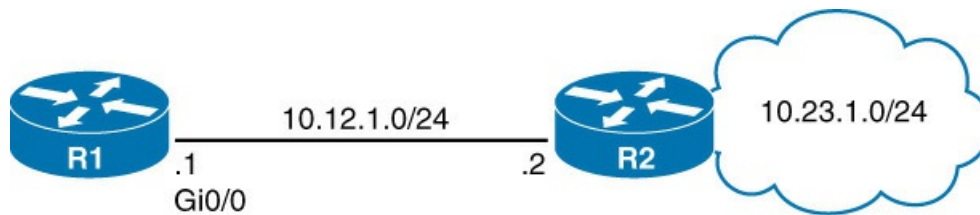


Figure 4-3 Static Route Concept

### Point-to-Point Interfaces

Point-to-point (P2P) interfaces do not have to worry about maintaining an adjacency table and do not use Address Resolution Protocol (ARP), so static routes can directly reference the outbound interface of the router. A static route that uses only the outbound next-hop interface is known as a *directly attached static route* and requires that the outbound interface be in an *up* state for the route to install into the RIB.

IOS nodes use the command **ip route network subnet-mask interface-type interface-number** for configuring a static route pointed directly out an interface.

IOS XR's configuration is hierarchical and a static route configuration occurs as follows:

#### Step 1. Initialize the static routing process.

The static routing process is initialized with the command **router static**.

#### Step 2. Identify the address family.

Initialize the appropriate address family with the command **address-family {ipv4 | ipv6} unicast address-family**. IPv4 static routes will use the IPv4 unicast option.

#### Step 3. Configure the static route.

Directly attached static routes use the configuration syntax **network {subnet-mask | /prefix-length} interface-type interface-number** for point-to-point interfaces.

#### Note

Configuring a directly attached static route to an interface that uses ARP (that is, Ethernet) will cause problems and is not recommended. The router must repeat the ARP process for every destination that matches the static route, which consumes CPU, and memory. Depending on the size of the prefix of the static route and number of lookups, the configuration can cause system instability.

Figure 4-4 illustrates XR1 connecting to R3 via a serial connection. XR1 will use a directly

attached static route to the 10.3.3.0/24 network, and R3 will use a directly attached static route to the 10.1.1.0/24 to allow connectivity between PC A and PC B. Static routes are required on both routers so that return traffic will have a path back.

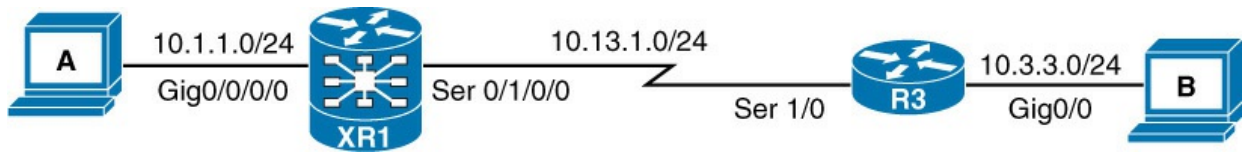


Figure 4-4 XR1 and R3 Connecting via Serial Connection

Example 4-4 displays the configuration of XR1 and R3 using static routes with P2P interfaces. XR1 defines the 10.3.3.0/24 network is reachable via the S0/1/0/0 interface, and R3 defines the 10.1.1.0/24 network is reachable via the S1/0 interface.

#### Example 4-4 Directly Attached Static Route Configuration

[Click here to view code image](#)

```
XR1
router static
address-family ipv4 unicast
  10.3.3.0/24 Serial 0/1/0/0
```

```
R3
ip route 10.1.1.0 255.255.255.0 Serial 1/0
```

Example 4-5 displays the routing table with the static route configured. A static route specifying only the outbound interface does not display the [AD/Metric] when looking at the routing table. Notice that the static route displays *directly connected* with the outbound interface. XR1 views the 10.3.3.0/24 network as directly connected via the Serial 0/1/0/0 interface, and R3 views the 10.1.1.0/24 network as directly connected to the Serial 1/0 interface.

#### Example 4-5 XR1 and R3 Routing Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
! Output omitted for brevity

Gateway of last resort is not set

C   10.1.1.0/24 is directly connected, 00:24:57, GigabitEthernet0/0/0/0
S   10.3.3.0/24 is directly connected, 00:24:22, Serial 0/1/0/0
C   10.13.1.0/24 is directly connected, 00:24:22, Serial 0/1/0/0
```

```
R3#show ip route
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
```

```
S 10.1.1.0/24 is directly connected, Serial 1/0
```

```
C 10.3.3.0/24 is directly connected, GigabitEthernet0/0
```

```
C 10.13.1.0/24 is directly connected, Serial1/0
```

**Example 4-6** demonstrates that bidirectional connectivity exists between the routers. Using the **ping** command with the source interface of the nonshared network simulates a ping from one of the PCs. This provides verification that both routers have connectivity to the 10.1.1.0/24 and 10.3.3.0/24 networks. If only one router was configured, the ping would fail because packets would not have a return route.

### **Example 4-6** Verification of Bidirectional Connectivity Between XR1 and R3

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#ping 10.3.3.3 source 10.1.1.1
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 10.3.3.3, timeout is 2 seconds:
```

```
!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
```

```
R3#ping 10.1.1.1 source GigabitEthernet0/0
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
```

```
Packet sent with a source address of 10.3.3.3
```

```
!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/12 ms
```

### **Broadcast Interfaces**

IPv4 Ethernet interfaces use ARP to determine the forwarding address on the shared medium, so static routes using these interface types need to specify an IP address for the next hop. Static routes of this type are known as *recursive static routes* and require the route's next-hop address to exist in the routing table to install the static route into the RIB. The static route will forward packets toward the next-hop IP address. Recursive static routes will work on P2P interfaces, too.

For IOS routers, the configuration is in the global configuration with the command **ip route network subnet-mask next-hop-ip**. Configuration on IOS-XR routers is under the appropriate address family. The command structure is *network {subnet-mask | /prefix-length} next-hop-ip*.

**Figure 4-5** shows a topology with XR1 and R3 connect via an Ethernet connection. XR1 will use a recursive static route to the 10.3.3.0/24 network, and R3 will use a recursive static route to the 10.1.1.0/24 to allow connectivity between PC A and PC B.



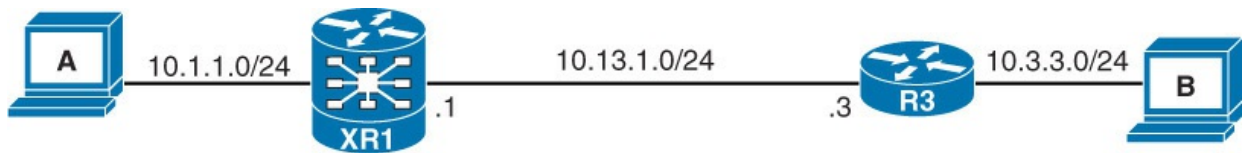


Figure 4-5 XR1 and R3 Connected by Ethernet

In Example 4-7, XR1's configuration states the 10.3.3.0/24 network is reachable via the 10.13.1.3 IP address, and R3's configuration states the 10.1.1.0/24 network is reachable via the 10.13.1.1 IP address.

### Example 4-7 Recursive Static Route Configuration

[Click here to view code image](#)

```
XR1
router static
address-family ipv4 unicast
10.3.3.0/24 10.13.1.3
```

```
R3
ip route 10.1.1.0 255.255.255.0 10.13.1.1
```

Example 4-8 shows the routing tables with the static routes configuration. Notice that the [AD/Metric] is present in the output and that the next-hop IP address is included.

### Example 4-8 IP Routing Table for XR1 and R3

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
! Output omitted for brevity

Gateway of last resort is not set

C    10.1.1.0/24 is directly connected, 00:24:57, GigabitEthernet0/0/0/0
S    10.3.3.0/24 [1/0] via 10.13.2.3, 00:01:01
C    10.13.1.0/24 is directly connected, 00:21:22, GigabitEthernet0/0/0/1
```

```
R3#show ip route
! Output omitted for brevity

Gateway of last resort is not set

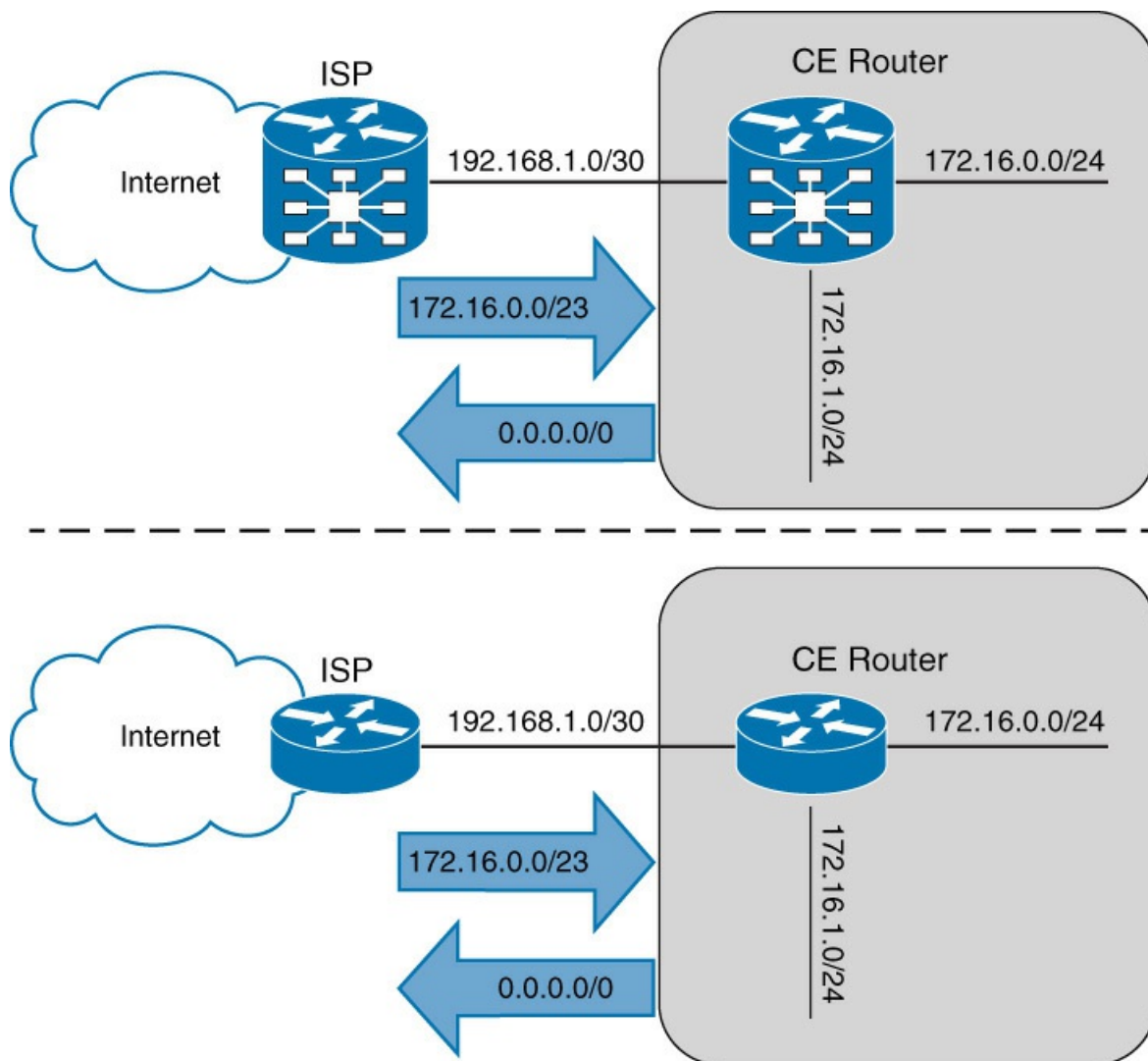
    10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
S    10.1.1.0/24 [1/0] via 10.13.2.1
C    10.3.3.0/24 is directly connected, GigabitEthernet0/0
C    10.13.1.0/24 is directly connected, GigabitEthernet0/2
```

## Default Route

A static route can be set for destination networks with any size prefix length, 0–32 bits (IPv4) and 0–128 bits (IPv6). A default route is the route of last resort. A router uses a default route to forward packets when a more specific route entry does not exist in the route table. A default route's destination network will be 0.0.0.0 with a subnet mask of 0.0.0.0 or a prefix length of /0.

Default routes are common method for providing basic connectivity when a router has only one exit path (normally a WAN circuit). Instead of programming multiple static routes that use the same outbound interface, a default route can replace the need to configure multiple static routes to the same destination if there is only one exit point. A default route reduces the configuration and memory consumed by the routing table.

In [Figure 4-6](#), the service provider assigns the 172.16.0.0/23 network range to a customer. The service provider will create a static route for the 172.16.0.0/23 network directed at the customer edge (CE) device. The CE router needs only a default route to provide Internet connectivity because there is only one external connection.



**Figure 4-6** ISP to CE Router Connectivity

[Example 4-9](#) demonstrates a default route configuration for the CE router using IOS or IOS XR.

### **Example 4-9** Default Route Configuration

[Click here to view code image](#)

```
IOS XR
router static
address-family ipv4 unicast
0.0.0.0/0 192.168.1.2
```

```
IOS
ip route 0.0.0.0 0.0.0.0 192.168.1.2
```

**Example 4-10** displays the routing tables for IOS or IOS XR platforms. Routing between the 172.16.0.0/24 and 172.16.1.0/24 networks occurs because they are connected networks to the router.

#### **Example 4-10** IP Routing Table for the CE Router Using IOS or IOS XR

[Click here to view code image](#)

```
RP/0/0/CPU0:IOS-XR#show route
! Output omitted for brevity

Gateway of last resort is 192.168.1.2 to network 0.0.0.0

S* 0.0.0.0/0 [1/0] via 192.168.1.2, 00:00:05
C 172.16.0.0/24 is directly connected, 00:04:54, GigabitEthernet0/0/0/1
C 172.16.1.0/24 is directly connected, 00:04:54, GigabitEthernet0/0/0/2
C 192.168.1.0/30 is directly connected, 00:03:44, GigabitEthernet0/0/0/0
```

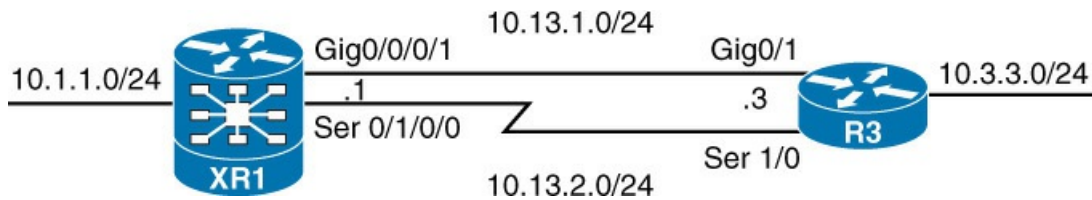
```
IOS#show ip route
! Output omitted for brevity

Gateway of last resort is 192.168.1.2 to network 0.0.0.0

S* 0.0.0.0/0 [1/0] via 192.168.1.2
10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
C 172.16.0.0/24 is directly connected, GigabitEthernet0/1
C 172.16.1.0/24 is directly connected, GigabitEthernet0/2
C 192.168.1.0/30 is directly connected, GigabitEthernet0/0
```

A default route can provide backup connectivity because more specific routes would be selected first by the router.

**Figure 4-7** shows XR1 and R3 with connectivity to each other with an Ethernet (10.13.1.0/24) and serial (10.13.2.0/24) link. XR1 and R3 should use the Ethernet link for forwarding all traffic between the 10.1.1.0/24 and 10.3.3.0/24 networks. The serial connection should be used only in the event that the Ethernet link fails.



**Figure 4-7** Backup Connectivity with a Default Static Route

**Example 4-11** provides the configuration where XR1 contains a recursive static route for the 10.3.3.0/24 network via the Ethernet link (10.13.1.3) and a directly connected static default route out the serial interface. R3 contains a static route for the 10.1.1.0/24 network via the Ethernet Link (10.13.1.1) and a static default route out the serial interface.

### **Example 4-11** XR1 and R3 Static Default Route Configuration

[Click here to view code image](#)

```
XR1
router static
address-family ipv4 unicast
  0.0.0/0 10.13.2.3
  10.3.3.0/24 10.13.1.3
```

```
R3
ip route 0.0.0.0 0.0.0.0 10.13.2.1
ip route 10.1.1.0 255.255.255.0 10.13.1.1
```

**Example 4-12** displays the routing tables for XR1 and R3. Notice that the gateway of last resort shows the next-hop address of the default route (remote endpoint of serial link). Traffic between 10.1.1.0/24 and 10.3.3.0/24 will use the Ethernet connection because those routes are a longer match in the RIB.

If the Ethernet link were to fail, the specific /24 route entries for the 10.1.1.0/24 and 10.3.3.0/24 prefixes would be removed from XR1 and R3's RIB, and traffic would flow across the serial interfaces using the default route.

### **Example 4-12** IP Routing Table for XR1 and R3 After Static Default Routes

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
! Output omitted for brevity

Gateway of last resort is 10.13.2.3 to network 0.0.0.0

S* 0.0.0.0/0 [1/0] via 10.13.2.3, 00:00:05
C 10.1.1.0/24 is directly connected, 00:04:54, GigabitEthernet0/0/0/0
S 10.3.3.0/24 [1/0] via 10.13.1.3, 00:03:44
C 10.13.2.0/24 is directly connected, 00:04:54, Serial10/1/0/0
C 10.13.1.0/24 is directly connected, 00:03:44, GigabitEthernet0/0/0/1
```

```

R3#show ip route
! Output omitted for brevity

Gateway of last resort is 10.13.2.1 to network 0.0.0.0

S* 0.0.0.0/0 [1/0] via 10.13.2.1
    10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
S   10.1.1.0/24 [1/0] via 10.13.1.1
C   10.3.3.0/24 is directly connected, GigabitEthernet0/0
C   10.13.2.0/24 is directly connected, Serial1/0
C   10.13.1.0/24 is directly connected, GigabitEthernet0/2

```

## Floating Static Routing

The default AD can be preempted with an AD value of 1–255 for a specific route. The AD is set on a static route by appending the AD for that route.

A *floating static route* is a common technique for providing a backup connectivity for prefixes learned via dynamic routing protocols. A floating static route is configured with an AD higher than the primary route. Because the AD is higher than the primary route, it will be installed only in the RIB when the primary route is withdrawn.

Previously shown in [Figure 4-7](#), XR1 and R3 have connectivity to each other with an Ethernet (10.13.1.0/24) and serial (10.13.2.0/24) link. XR1 and R3 should use the Ethernet link for traffic and only use the serial link in the event that the Ethernet link fails. XR1 and R3 should use a floating static route.

[Example 4-13](#) demonstrates the configuration of XR1 and R3. The static route using the Ethernet link (10.13.1.0/24) has an AD of 10, and the serial link (10.13.2.0/24) has an AD set to 210.

### Example 4-13 Floating Static Route Configuration for XR1 and R3

[Click here to view code image](#)

```

XR1
router static
address-family ipv4 unicast
  10.3.3.0/24 10.13.1.3 10
  10.3.3.0/24 Serial 0/1/0/0 210

```

```

R3
ip route 10.1.1.0 255.255.255.0 10.13.1.1 10
ip route 10.1.1.0 255.255.255.0 Serial 1/0 210

```

[Example 4-14](#) displays the routing tables of XR1 and R3. Notice that the static route across the serial link does not install into the RIB. Only the static route for the Ethernet link (10.13.1.0/24) with an AD of 10 installs into the RIB.

### Example 4-14 Routing Table of XR1 and R3 with a Floating Static Route

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
! Output omitted for brevity
```

Gateway of last resort is not set

```
C 10.1.1.0/24 is directly connected, 00:24:57, GigabitEthernet0/0/0/0
S 10.3.3.0/24 [10/0] via 10.13.1.3, 00:01:01
C 10.13.2.0/24 is directly connected, 00:24:57, Serial0/1/0/0
C 10.13.1.0/24 is directly connected, 00:21:22, GigabitEthernet0/0/0/1
```

```
R3#show ip route
```

```
! Output omitted for brevity
```

Gateway of last resort is not set

```
10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
S 10.1.1.0/24 [10/0] via 10.13.1.1
C 10.3.3.0/24 is directly connected, GigabitEthernet0/0
C 10.13.2.0/24 is directly connected, Serial1/0
C 10.13.1.0/24 is directly connected, GigabitEthernet0/2
```

**Example 4-15** displays the routing table for router XR1 and R3 after the Ethernet link has been shut down to simulate a link failure. The 10.13.1.0/24 network (XR1's Gigabit Ethernet 0/0/0/1 and R3's Gi0/3) is removed from the RIB. The floating static route via the 10.13.2.0/24 network (XR1's So/1/0/0 and R3's S1/0) is now the best path and is installed into the RIB. Notice that the AD is not shown for that static route.

### **Example 4-15** Routing Table After Ethernet Link Failure

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
! Output omitted for brevity
```

Gateway of last resort is not set

```
C 10.1.1.0/24 is directly connected, 00:24:57, GigabitEthernet0/0/0/0
S 10.3.3.0/24 is directly connected, 00:24:22, Serial 0/1/0/0
C 10.13.1.0/24 is directly connected, 00:24:22, Serial 0/1/0/0
```

```
R3#show ip route
```

```
! Output omitted for brevity
```

Gateway of last resort is not set

```
10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
S 10.1.1.0/24 is directly connected, Serial 1/0
C 10.3.3.0/24 is directly connected, GigabitEthernet0/0
C 10.13.2.0/24 is directly connected, Serial1/0
```

Even though the AD is not shown, it is still programmed with the static route. [Example 4-16](#) shows the explicit network entry. The output confirms that the floating static route with an AD of 210 is currently active in the routing table.

### Example 4-16 Verification of AD for Floating Static Route

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route 10.3.3.0
Routing entry for 10.3.3.0/24
  Known via "static", distance 210, metric 0 (connected)
  Routing Descriptor Blocks
    directly connected, via Serial 0/1/0/0
      Route metric is 0
  No advertising protos.
```

```
R3#show ip route 10.1.1.0
Routing entry for 10.1.1.0/24
  Known via "static", distance 210, metric 0 (connected)
  Routing Descriptor Blocks:
    * directly connected, via Serial 1/0
      Route metric is 0, traffic share count is 1
```

### Recursive Lookup

The forwarding engine on Cisco devices needs to know which interface an outbound packet should use. A recursive lookup refers to the process in which additional checks on the router are required to identify the outbound interface for a route in the routing table. Directly attached static routes do not need to perform a recursive lookup because the outbound interface is already specified. A recursive static route will use a different method for identifying the outbound interface.

[Figure 4-8](#) provides a topology to explain how Cisco devices find the outbound interface for a static route. XR1 and R3 have two network segments each and are connected with an Ethernet and serial link. Each router will have two static routes to the remote network to demonstrate the underlying mechanics within the forwarding engine.

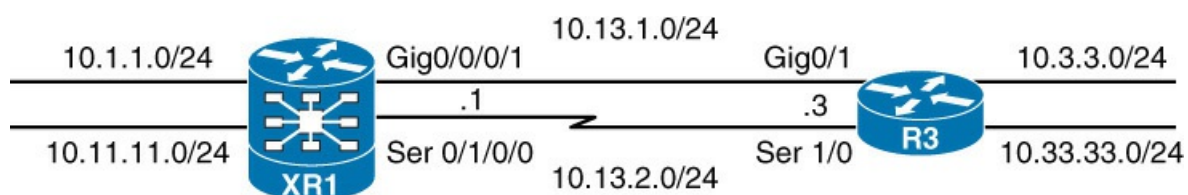


Figure 4-8 Recursive Lookup Topology

[Example 4-17](#) provides the relevant configurations for XR1 and R3. XR1 and R3 are connected with an Ethernet link (10.13.1.0/24) and a serial link (10.13.2.0/24). XR1 will configure a recursive static route to the 10.3.3.0/24 using the next-hop IP address 10.13.1.3 and use a second directly attached static route to the 10.33.33.0/24 using the Serial 0/1/0/0 interface. R3 will configure a recursive static route to the 10.1.1.0/24 network using the next-hop IP address 10.13.1.2 and configure a second directly attached static route to the 10.11.11.0/24 network using the Serial 1/0 interface.

## Example 4-17 Configuration for XR and R3

[Click here to view code image](#)

### XR1

```
router static
address-family ipv4 unicast
 10.3.3.0/24 10.13.1.3
 10.33.33.0/24 Serial0/1/0/0
```

### R3

```
ip route 10.1.1.0 255.255.255.0 10.13.1.1
ip route 10.11.11.0 255.255.255.0 Serial1/0
```

Example 4-18 provides the explicit route entries for XR1 and R3. The 10.1.1.0 and 10.3.3.0 networks list the IP address for the next-hop. The router will need to identify the physical interface connected to the next-hop IP address. The *recursive lookup* is the process of identifying the outbound interface using the next-hop IP address and the adjacency table.

The 10.11.11.0 and 10.33.33.0 network entries include the outbound interface, and a recursive lookup is not required.

## Example 4-18 Explicit RIB Entries for Remote Networks

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route 10.3.3.0
! Output omitted for brevity
Routing entry for 10.3.3.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks
    10.13.1.3
      Route metric is 0

RP/0/0/CPU0:XR1#show route 10.33.33.0
! Output omitted for brevity
Routing entry for 10.33.33.0/24
  Known via "static", distance 1, metric 0 (connected)
  Routing Descriptor Blocks
    directly connected, via Serial 0/1/0/0
      Route metric is 0
```



```

R3#show ip route 10.1.1.0
Routing entry for 10.1.1.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
    * 10.13.1.1
      Route metric is 0, traffic share count is 1

R3#show ip route 10.11.11.0
Routing entry for 10.11.11.0/24
  Known via "static", distance 1, metric 0 (connected)
  Routing Descriptor Blocks:
    * directly connected, via Serial1/0
      Route metric is 0, traffic share count is 1

```

The Forwarding Information Base (FIB) will contain the outbound interface used for a route. The CEF table may be viewed on IOS and IOS XR nodes with the command **show ip cef network [/prefix-length | subnet-mask] [detail]**.

**Example 4-19** provides detailed output of the CEF table for XR1 and R3 for the remote networks. Notice that the 10.3.3.0 and 10.1.1.0 networks show as recursive with the outbound interface identified.

#### **Example 4-19** Configurations for Nonrecursive Multihop Static Routes

[Click here to view code image](#)

```

RP/0/0/CPU0:XR1#show cef 10.3.3.0 detail
! Output omitted for brevity
10.3.3.0/24, version 30, internal 0x4000001 (ptr 0x577189f4)
Prefix Len 24, traffic index 0, precedence routine (0), priority 3
via 10.13.1.3, 2 dependencies, recursive [flags 0x0]
  next hop 10.13.1.3 via 10.13.1.3/32

Hash OK Interface Address
0 Y GigabitEthernet0/0/0/0 10.13.1.3

RP/0/0/CPU0:XR1#show cef 10.33.33.0 detail
! Output omitted for brevity
10.33.33.0/24, version 42, attached, internal 0x4000081 (ptr 0x57719174)
Prefix Len 24, traffic index 0, precedence routine (0), priority 2
via Serial0/1/0/0, 4 dependencies, weight 0, class 0 [flags 0x8]

Hash OK Interface Address
0 Y Serial0/1/0/0 remote

```

```

R3#show ip cef 10.1.1.0 detail
10.1.1.0/24, epoch 0
  recursive via 10.13.1.1
    attached to GigabitEthernet0/0

R3#show ip cef 10.11.11.0 detail
10.11.11.0/24, epoch 0, flags attached
  attached to Serial1/0

```

## Multihop Routing

Multihop routing involves network paths that must cross two or more routers to reach the destination. Static routing does not scale very well with multihop routing because all routers along the path need to be aware of the source and destination networks to ensure full IP reachability. Failure to do so will result in packet drops. Topologies with a large number of routers may be extremely complex for network administrators to manage when using static routing.

Figure 4-9 represents a topology where the network 10.34.1.0/24 is two hops away from XR1, and 10.12.1.0/24 is two hops away from R4. Providing bidirectional connectivity between XR1 and R4 will require installing multiple routes. XR1 will need a static route to reach the 10.34.1.0/24 network and R4 will need a static route to reach the 10.12.1.0/24 network. Both routers will create a static route to the 10.23.1.0/24 network to allow full connectivity between the four routers in the topology.

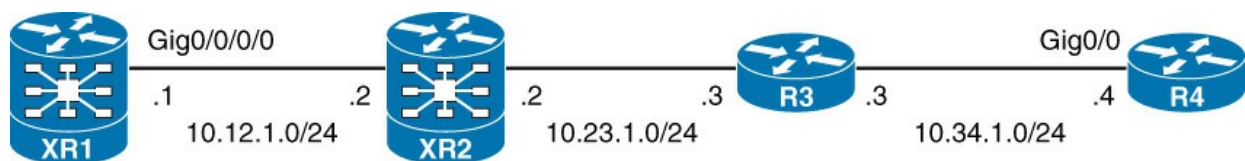


Figure 4-9 Multihop Topology

### Note

XR2 and R3 are connected or one hop away and will require only one static route. This section focuses on XR1 and R4.

## Single Recursive Lookup

The simplest method for multihop static routes is to use the same next-hop address for both network prefixes. When a packet is forwarded on XR1 or R4, the router identifies the next-hop IP address in the routing table; the router performs a single recursive lookup for the outbound interface.

Example 4-20 shows the configuration of XR1 and R3. XR1 and R4 set the next-hop IP address for both routes to the upstream router.

## Example 4-20 Single Recursive Static Routes

[Click here to view code image](#)

```

XR1
router static
address-family ipv4 unicast
  10.23.1.0/24 10.12.1.2
  10.34.1.0/24 10.12.1.2

```

#### R4

```
ip route 10.12.1.0 255.255.255.0 10.34.1.3
ip route 10.23.1.0 255.255.255.0 10.34.1.3
```

Example 4-21 shows the routing table with single recursive lookups. Notice the next-hop address for each network is the same.

#### Example 4-21 Single Recursive Routing Table for XR1 and R4

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1# show route
! Output omitted for brevity

Gateway of last resort is not set

C   10.12.1.0/24 is directly connected, 00:27:03, GigabitEthernet0/0/0/0
S   10.23.1.0/24 [1/0] via 10.12.1.2, 00:27:03
S   10.34.1.0/24 [1/0] via 10.12.1.2, 00:27:03
```

```
R4#show ip route
! Output omitted for brevity

Gateway of last resort is not set

    10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
S       10.12.1.0/24 [1/0] via 10.34.1.3
S       10.23.1.0/24 [1/0] via 10.34.1.3
C       10.34.1.0/24 is directly connected, GigabitEthernet0/0
```

Using single recursive static routes removes the burden of having to know the topology or any upcoming changes in the topology. The downfall to this is that when a new link is added all the static routes need to be changed.

#### Multiple Recursive Lookups

Another method for multihop static routes allows for using a next-hop IP address that is contained within another entry in the routing table. The router will perform multiple recursive lookups to identify the outbound interface.

In Figure 4-9, XR1 needs to establish connectivity to R4 (10.34.1.4). XR1 will install a route to the 10.23.1.0/24 network with XR2 (10.12.1.2) as the next hop. XR1 can configure connectivity to the 10.34.1.0/24 network with R3 (10.23.1.3) as the next hop.

When XR1 sends a packet to R4, it locates the next-hop IP address of 10.23.1.3. XR1 will not see 10.23.1.3 in the adjacency table but will then locate the route to the 10.23.1.0/24 network. The 10.23.1.0/24 network will use the next-hop IP address of 10.12.1.2 that is listed in the adjacency table. XR1 has performed two recursive lookups to locate the outbound interface for a route.

If XR1 removes the 10.23.1.0/24 from the RIB, the recursive lookup will fail. XR1 will remove the 10.34.1.0/24 because the next-hop IP address for the 10.23.1.0/24 route is no longer reachable

via the 10.23.1.0/24 network.

**Example 4-22** shows the router configuration on XR1 and R4 using recursive static routes. XR1 configures the static route to the 10.34.1.0/24 network with the next-hop IP address of 10.23.1.3. R4 configures the static route to the 10.12.1.0/24 network with the next-hop IP address of 10.23.1.2.

### **Example 4-22** Multiple Recursive Lookups Configuration

[Click here to view code image](#)

#### **XR1**

```
router static
  address-family ipv4 unicast
    10.23.1.0/24 10.12.1.2
    10.34.1.0/24 10.23.1.3
```

#### **R4**

```
ip route 10.12.1.0 255.255.255.0 10.23.1.2
ip route 10.23.1.0 255.255.255.0 10.34.1.3
```

**Example 4-23** shows the routing table for XR1 and R4 with recursive static routes. Notice the next-hop IP addresses are not directly connected and reachable via route recursion.

### **Example 4-23** Routing Table with Multiple Recursive Lookups

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
C 10.12.1.0/24 is directly connected, 00:36:44, GigabitEthernet0/0/0/0
S 10.23.1.0/24 [1/0] via 10.12.1.2, 00:36:44
S 10.34.1.0/24 [1/0] via 10.23.1.3, 00:00:32
```

```
R4#show ip route
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
S 10.12.1.0/24 [1/0] via 10.23.1.2
S 10.23.1.0/24 [1/0] via 10.34.1.3
C 10.34.1.0/24 is directly connected, GigabitEthernet0/0
```

**Example 4-24** provides the detailed CEF output for the remote network segment. IOS shows a logical progression of the recursive lookups on a step-by-step basis. IOS XR shows the first route

lookup and provides the final outbound interface and initial next-hop IP address.

### Example 4-24 Routing Table with Recursive Static Routes

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show cef 10.34.1.0 detail
! Output omitted for brevity
10.34.1.0/24, version 17, internal 0x4000001 (ptr 0x576721f4)
Prefix Len 24, traffic index 0, precedence routine (0), priority 3
via 10.23.1.3, 2 dependencies, recursive [flags 0x0]
next hop 10.23.1.3 via 10.23.1.0/24

Hash OK Interface Address
0 Y GigabitEthernet0/0/0/0 10.12.1.2
```

```
R4#show ip cef 10.12.1.0 detail
10.12.1.0/24, epoch 0
recursive via 10.23.1.2
recursive via 10.23.1.0/24
recursive via 10.34.1.3
attached to GigabitEthernet0/0
```

#### Note

A recursive static route may not resolve the next-hop forwarding address using the default route (0.0.0.0) entry. The static route will fail next-hop reachability requirements and will not be inserted into the RIB. An exception is made for IOS when there is a directly attached default route where only the output interface is specified. The destination is assumed to be directly connected, so the reachability test will succeed.

Cisco does not recommend using directly attached default routes on interfaces that use ARP because it can impact the performance or stability on the system.

Figure 4-10 shows an example of where multiple recursive lookups are beneficial. XR1 configures a static route to the 10.34.1.0/24 network with the next-hop IP address of 10.23.1.3, and configures two static routes to the 10.23.1.0/24 network using the next-hop IP address of 10.12.1.2 and 10.12.2.2. If either network (10.12.1.0/24 or 10.12.2.0/24) fails, XR1 still maintains connectivity to the 10.34.1.0/24 network. This would require fewer static routes than using static route entries with single recursive lookups.

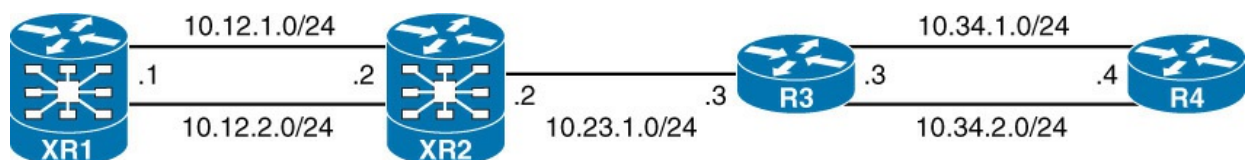


Figure 4-10 Multihop Technology

#### Problems with Static Route Recursion

Static route recursion can simplify topologies if a link fails because it may allow the static route to stay installed while it changes to a different outbound interface in the same direction of the destination. The opposite behavior can cause unintended consequences in a network, too.

To illustrate this concept, Figure 4-11 shows two companies, ABC and XYZ, where two network

links are deployed between their edge routers so that the servers in 192.168.1.0/24 and 192.168.2.0/24 subnets can talk to each other. The Ethernet link is preferred because it has a lower AD. The serial link should be used only as a backup via a floating static route.

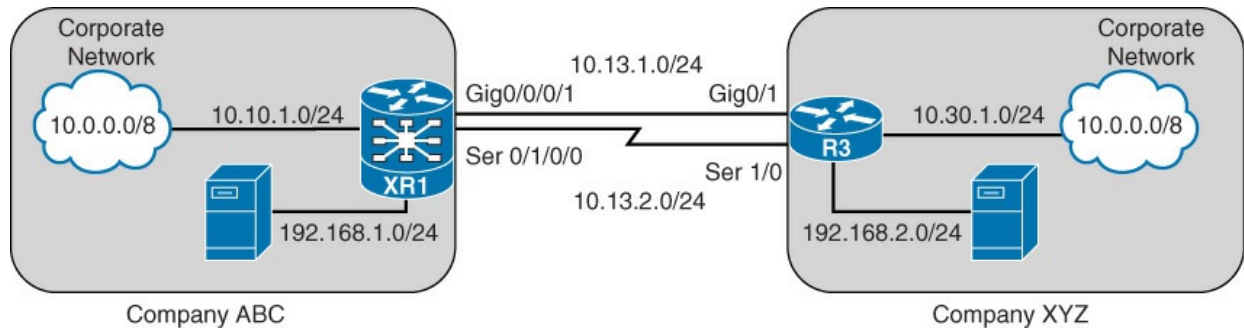


Figure 4-11 Company ABC and XYZ Connection

Example 4-25 shows XR1's and R3's configuration with floating static routes. Notice that the static route using the serial link uses a floating static route with an AD of 50.

### Example 4-25 Configuration with Floating Static Routes for Backup

[Click here to view code image](#)

```
XR1
router static
address-family ipv4 unicast
10.0.0.0/8 10.10.1.2
192.168.2.0/24 10.13.1.3
192.168.2.0/24 10.13.2.3 50
```

```
R3
ip route 10.0.0.0 0.0.0.0 10.30.1.2
ip route 192.168.1.0 255.255.255.0 10.13.1.1
ip route 192.168.1.0 255.255.255.0 10.13.2.1 50
```

Example 4-26 shows the routing table while all links are stable. Notice that the routes to the 192.168.x.x/24 network have a next-hop IP address in the 10.13.1.0/24 network. The static route has an AD of 1, which confirms the Ethernet link is being used.

### Example 4-26 Routing Table for XR1 and R3

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
! Output omitted for brevity
```

Gateway of last resort is 192.168.1.2 to network 0.0.0.0

```
S 10.0.0.0/8 [1/0] via 10.10.1.2, 00:04:43
C 10.10.1.0/24 is directly connected, 00:04:43, GigabitEthernet0/0/0/2
C 10.13.1.0/24 is directly connected, 00:04:43, GigabitEthernet0/0/0/1
C 10.13.2.0/24 is directly connected, 00:04:43, Serial0/1/0/0
C 192.168.1.0/24 is directly connected, 00:08:47, GigabitEthernet0/0/0/0
S 192.168.2.0/24 [1/0] via 10.13.1.3, 00:04:43
```

```
R3#show ip route
```

```
! Output omitted for brevity
```

Gateway of last resort is not set

```
10.0.0.0/8 is variably subnetted, 5 subnets, 3 masks
S 10.0.0.0/8 [1/0] via 10.30.1.1
C 10.13.1.0/24 is directly connected, GigabitEthernet0/1
C 10.13.2.0/24 is directly connected, Serial1/0
C 10.30.1.0/24 is directly connected, GigabitEthernet0/2
S 192.168.1.0/24 [1/0] via 10.13.1.1
192.168.2.0/24 is variably subnetted, 2 subnets, 2 masks
C 192.168.2.0/24 is directly connected, GigabitEthernet0/0
```

Figure 4-12 shows a failure on the Ethernet link between XR1 and R3.

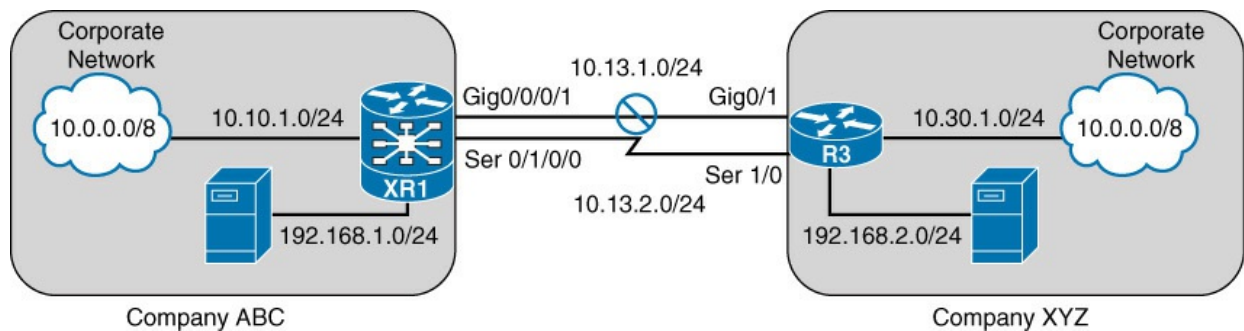


Figure 4-12 Ethernet Link Failure Between XR1 and R3

Example 4-27 shows the routing table of XR1 and R3 with the failed link. Notice that the static route for the 192.168.x.x/24 network still shows the next-hop IP address in the 10.13.1.0/24 network, but the 10.13.1.0/24 network is not in the routing table. The AD still shows a value of 1, so the floating static route using the serial link did not install.

#### Example 4-27 Routing Table After Link Failure

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
```

```
! Output omitted for brevity
```

```
S 10.0.0.0/8 [1/0] via 10.10.1.2, 00:07:34
C 10.10.1.0/24 is directly connected, 00:04:43, GigabitEthernet0/0/0/2
C 10.13.2.0/24 is directly connected, 00:07:34, Serial0/1/0/0
C 192.168.1.0/24 is directly connected, 00:08:47, GigabitEthernet0/0/0/0
S 192.168.2.0/24 [1/0] via 10.13.1.3, 00:07:34
```

```
R3#show ip route
```

```
! Output omitted for brevity
```

```
10.0.0.0/8 is variably subnetted, 5 subnets, 3 masks
S 10.0.0.0/8 [1/0] via 10.30.1.1
C 10.13.2.0/24 is directly connected, Serial1/0
C 10.30.1.0/24 is directly connected, GigabitEthernet0/2
S 192.168.1.0/24 [1/0] via 10.13.1.1
192.168.2.0/24 is variably subnetted, 2 subnets, 2 masks
C 192.168.2.0/24 is directly connected, GigabitEthernet0/0
```

Examining the CEF table in [Example 4-28](#) explains why the static route with the next-hop IP address of 10.13.1.3 (XR1) and 10.13.1.1 (R3)'s did not remove from the RIB. The appropriate next-hop IP address recursively found in the 10.0.0.0/8 static route to the corporate network. ABC traffic destined for servers in the 192.168.2.0/24 network will be sent to ABC's corporate network, and XYZ traffic destined for servers in the 192.168.1.0/24 network will be sent to XYZ's corporate network.

### Example 4-28 CEF Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show cef 192.168.2.0
```

```
192.168.2.0/24, version 37, internal 0x4000001 (ptr 0x574f4cf4) [1], 0x0 (0x0), 0x0 (0x0)
```

```
Prefix Len 24, traffic index 0, precedence routine (0), priority 3
```

```
via 10.13.1.3, 2 dependencies, recursive [flags 0x0]
```

```
path-idx 0 [0x574f4ff4 0x0]
```

```
next hop 10.13.1.3 via 10.0.0.0/8
```

```
R3#show ip cef 192.168.1.0 detail
```

```
192.168.1.0/24, epoch 0
```

```
recursive via 10.13.1.1
```

```
recursive via 10.0.0.0/8
```

```
recursive via 10.30.1.1
```

```
recursive via 10.30.1.0/24
```

```
attached to GigabitEthernet0/2
```

To correct this issue, the static route configuration should use the outbound interface and the next-hop IP address. Static routes with both an interface and next-hop IP address are known as a



*fully specified static route*. If the interface listed is not in an *up* state, the router removes the static route from the RIB. Specifying the next-hop address along with the physical interface removes the recursive lookup and does not include the ARP processing problems found when using only the outbound interface.

**Example 4-29** demonstrates the static route configuration using next-hop IP addressing along with the outbound interface.

### **Example 4-29** *Sample Configuration for XR1 and R3 with Static Route*

[Click here to view code image](#)

```
X2
router static
address-family ipv4 unicast
 10.0.0.0/8 10.10.1.2
 192.168.2.0/24 10.13.2.3 50
 192.168.2.0/24 GigabitEthernet0/0/0/1 10.13.1.3
```

```
R3
ip route 10.0.0.0 255.0.0.0 10.30.1.1
ip route 192.168.1.0 255.255.255.0 GigabitEthernet0/1 10.13.1.1
ip route 192.168.1.0 255.255.255.0 10.13.2.1 50
```

**Example 4-30** confirms that the 10.13.1.0/24 is not present when the Ethernet link is shut down. Notice that the routing table does not have the fully specified static route via the 10.13.1.0/24 network. The next-hop IP address and AD for the static routes to the 192.168.x.x/24 networks has changed to 50. Now the floating static routes install as planned.

### **Example 4-30** *Routing Table with Floating Static Route Working as Intended*

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route
! Output omitted for brevity

S   10.0.0.0/8 [1/0] via 10.10.1.2, 00:16:14
C   10.10.1.0/24 is directly connected, 00:16:14, GigabitEthernet0/0/0/2
C   10.13.2.0/24 is directly connected, 00:16:14, Serial0/1/0/0
C   192.168.1.0/24 is directly connected, 00:08:47, GigabitEthernet0/0/0/0
S   192.168.2.0/24 [50/0] via 10.13.2.3, 00:03:14
```

```
R3#show ip route
```

```
! Output omitted for brevity
```

```
10.0.0.0/8 is variably subnetted, 5 subnets, 3 masks
S    10.0.0.0/8 [1/0] via 10.30.1.1
C    10.13.2.0/24 is directly connected, Serial1/0
C    10.30.1.0/24 is directly connected, GigabitEthernet0/2
S    192.168.1.0/24 [50/0] via 10.13.2.1
192.168.2.0/24 is variably subnetted, 2 subnets, 2 masks
C    192.168.2.0/24 is directly connected, GigabitEthernet0/0
```

**Example 4-31** displays the routing tables for the 192.168.1.0/24 and 192.168.2.0/24 networks after the Ethernet link has recovered. The 10.13.1.0/24 connected network exists in both routers' routing tables, so the RIB reinstalls the fully specified static route for the Ethernet link because it has the better AD (1). Notice that the fully specified static route includes the outbound interface and next-hop IP address.

### **Example 4-31** Routing Table with Outbound Interface and Next-Hop IP Address

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
```

```
! Output omitted for brevity
```

```
S    10.0.0.0/8 [1/0] via 10.10.1.2, 00:27:35
C    10.10.1.0/24 is directly connected, 00:27:35, GigabitEthernet0/0/0/2
C    10.13.1.0/24 is directly connected, 00:00:14, GigabitEthernet0/0/0/1
C    10.13.2.0/24 is directly connected, 00:27:35, Serial0/1/0/0
C    192.168.1.0/24 is directly connected, 00:08:47, GigabitEthernet0/0/0/0
S    192.168.2.0/24 [1/0] via 10.13.1.3, 00:00:14, GigabitEthernet0/0/0/1
```

```
R3#show ip route
```

```
! Output omitted for brevity
```

```
10.0.0.0/8 is variably subnetted, 7 subnets, 3 masks
S    10.0.0.0/8 [1/0] via 10.30.1.1
C    10.13.1.0/24 is directly connected, GigabitEthernet0/1
C    10.13.2.0/24 is directly connected, Serial1/0
C    10.30.1.0/24 is directly connected, GigabitEthernet0/2
S    192.168.1.0/24 [1/0] via 10.13.1.1, GigabitEthernet0/2
192.168.2.0/24 is variably subnetted, 2 subnets, 2 masks
C    192.168.2.0/24 is directly connected, GigabitEthernet0/0
```

#### Note

Cisco recommends that a fully specified static route is used to ensure predictable behavior.

## NULL INTERFACE

The null interface is a virtual interface that is always in an *up* state. Null interfaces do not forward or receive network traffic and drop all traffic destined toward them without adding overhead to a router's CPU.

Configuring a static route to a null interface provides a method to drop network traffic without requiring the configuration of an access list. Creating a static route to the Null0 interface is a common technique to prevent routing loops. The static route to the Null0 interface uses a summarized network range while routes that are more specific point toward the actual destination.

Figure 4-13 shows a common topology where company ABC has acquired the 172.16.0.0/20 network range from their service provider. ABC uses only a portion of the given addresses but keeps the large network block in anticipation of future growth.

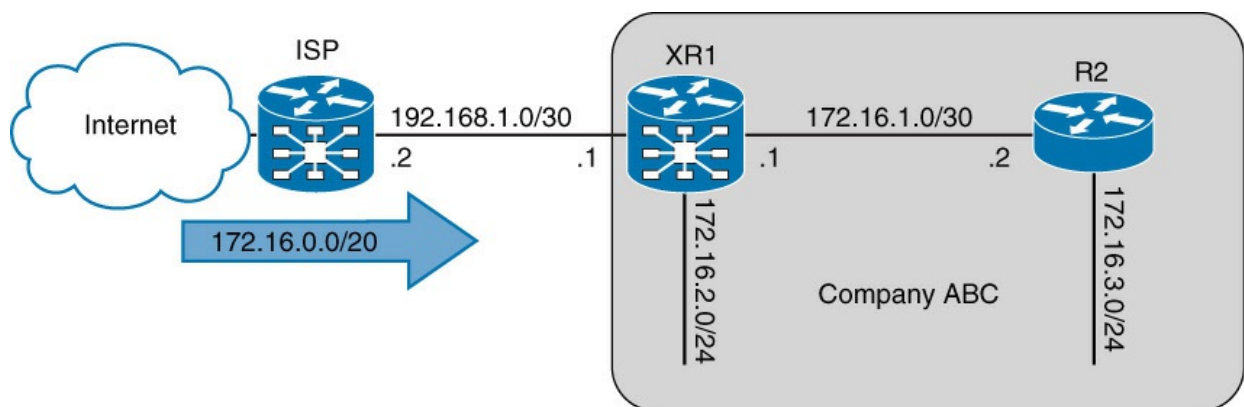


Figure 4-13 Routing Loop Topology

The service provider places a static route for the 172.16.0.0/20 to XR1's interface (192.168.1.1). XR1 uses a static default route pointed toward the service provider (192.168.1.2) and a static route to the 172.16.3.0/24 network via R2 (172.16.1.2). Because R2 accesses all other networks through XR1, a static default route points toward XR1's interface (172.16.1.1).

Example 4-32 shows the configuration for XR1 and R2.

### Example 4-32 R1 and R2 Configuration

[Click here to view code image](#)

```
XR1
router static
address-family ipv4 unicast
0.0.0.0/0 GigabitEthernet0/0/0/0 192.168.1.2
172.16.3.0/24 GigabitEthernet0/0/0/2 172.16.1.2
```

```
R2
ip route 0.0.0.0 0.0.0.0 GigabitEthernet0/0 172.16.1.1
```

Example 4-33 displays the IP routing table for XR1 and R2.

## Example 4-33 IP Routing Table for XR1 and R2

[Click here to view code image](#)

```
XR1#show ip route
! Output omitted for brevity

Gateway of last resort is 192.168.1.2 to network 0.0.0.0

S* 0.0.0.0/0 [1/0] via 192.168.1.2, 00:00:27, GigabitEthernet0/0/0/0
C   172.16.1.0/30 is directly connected, 00:00:27, GigabitEthernet0/0/0/2
C   172.16.2.0/24 is directly connected, 00:00:27, GigabitEthernet0/0/0/1
S   172.16.3.0/24 [1/0] via 172.16.1.2, 00:00:27, GigabitEthernet0/0/0/2
C   192.168.1.0/30 is directly connected, 00:00:27, GigabitEthernet0/0/0/0
```

```
R2#show ip route
! Output omitted for brevity

Gateway of last resort is 172.16.1.1 to network 0.0.0.0

S* 0.0.0.0/0 [1/0] via 172.16.1.1, GigabitEthernet0/0
   10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C   10.2.2.0/24 is directly connected, GigabitEthernet0/1
   172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
C   172.16.1.0/30 is directly connected, GigabitEthernet0/0
```

If packets are sent to any address in the 172.16.0.0/20 range that is not used by company ABC, the packet will get stuck in a loop between XR1 and the Internet service provider (ISP), consuming additional bandwidth until the packet's time-to-live (TTL) expires.

For example, a computer on the Internet sends a packet to 172.16.5.5, and the 172.16.5.0/24 is not allocated on XR1 or R2. The ISP will send the packet to XR1 because of the 172.16.0.0/20 static route. XR1 looks into the RIB (and the longest match for that prefix is the default route back to the ISP) and then sends the packet back to the ISP, thus creating the routing loop.

Example 4-34 shows the routing loop when packets originate from R2 or the ISP router. Notice the IP address in the traceroute alternate between the ISP router (192.168.1.2) and XR1 (192.168.1.1).

## Example 4-34 Packet Traces Demonstrating the Routing Loop

[Click here to view code image](#)

```
R2#trace 172.16.5.5 source GigabitEthernet 0/2
```

```
Type escape sequence to abort.
```

```
Tracing the route to 172.16.5.5
```

```
 1 172.16.1.1 0 msec 0 msec 0 msec
 2 192.168.1.1 0 msec 0 msec 0 msec
 3 192.168.1.2 0 msec 4 msec 0 msec
 4 192.168.1.1 0 msec 0 msec 0 msec
 5 192.168.1.2 0 msec 0 msec 0 msec
```

```
! Output omitted for brevity
```

```
RP/0/0/CPU0:ISP#trace 172.16.5.5
```

```
Type escape sequence to abort.
```

```
Tracing the route to 172.16.5.5
```

```
 1 192.168.1.1 3 msec 1 msec 1 msec
 2 192.168.1.2 1 msec 1 msec 1 msec
 3 192.168.1.1 2 msec 1 msec 2 msec
 4 192.168.1.2 1 msec 2 msec 2 msec
 5 192.168.1.1 2 msec 2 msec 2 msec
```

```
! Output omitted for brevity
```

To prevent the routing loop, a static route is added for 172.16.0.0/20 pointed to the Null0 interface on XR1. Any packets matching the 172.16.0.0/20 network range that do not have a longer match in XR1's RIB are dropped.

To optimize the network further, R2 adds a static route to 172.16.0.0/20 pointed at Null0. R2 will also need to add a static route to the 172.16.2.0/24 network pointed to XR1 (172.16.1.1) because it will need a more explicit route to avoid traffic for that network from being dropped by the Null0 aggregate route.

[Example 4-35](#) provides updated configurations for XR1 and R2.

### **Example 4-35** XR1 and R2 Configurations to 172.16.0.0/20 to Null0

[Click here to view code image](#)

#### **XR1**

```
router static
 address-family ipv4 unicast
 0.0.0.0/0 GigabitEthernet0/0/0/0 192.168.1.2
 172.16.0.0/20 Null0
 172.16.3.0/24 GigabitEthernet0/0/0/2 172.16.1.2
```

#### **R2**

```
ip route 0.0.0.0 0.0.0.0 GigabitEthernet0/0 172.16.1.1
ip route 172.16.0.0 255.255.240.0 Null0
ip route 172.16.2.0 255.255.255.0 GigabitEthernet0/0 172.16.1.1
```

[Example 4-36](#) displays the routing table on XR1 and R2. The summary (aggregate) route 172.16.0.0/20 to the Null0 is used to suppress routing loops and the specific static routes to 172.16.2.0/24 and 172.16.3.0/24 provide internal connectivity between the LAN segments.

### Example 4-36 IP Routing Table for R1 and R3

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
! Output omitted for brevity

Gateway of last resort is 192.168.1.2 to network 0.0.0.0

S* 0.0.0.0/0 [1/0] via 192.168.1.2, 00:23:22, GigabitEthernet0/0/0/0
S 172.16.0.0/20 is directly connected, 00:04:50, Null0
C 172.16.1.0/30 is directly connected, 00:23:22, GigabitEthernet0/0/0/2
C 172.16.2.0/24 is directly connected, 00:23:22, GigabitEthernet0/0/0/1
S 172.16.3.0/24 [1/0] via 172.16.1.2, 00:17:12, GigabitEthernet0/0/0/2
C 192.168.1.0/30 is directly connected, 00:23:22, GigabitEthernet0/0/0/0
```

```
R2#show ip route
! Output omitted for brevity

Gateway of last resort is 172.16.1.1 to network 0.0.0.0

S* 0.0.0.0/0 [1/0] via 172.16.1.1, Ethernet0/0
    172.16.0.0/16 is variably subnetted, 6 subnets, 4 masks
S 172.16.0.0/20 is directly connected, Null0
C 172.16.1.0/30 is directly connected, GigabitEthernet0/0
S 172.16.2.0/24 [1/0] via 172.16.1.1, GigabitEthernet0/0
C 172.16.3.0/24 is directly connected, GigabitEthernet0/1
```

[Example 4-37](#) demonstrates a traceroute from R2 to the address 172.16.5.5. The 172.16.5.5 matches the null route 172.16.0.0/20 on XR1, so the packets are dropped, avoiding a routing loop.

### Example 4-37 IP Routing Table for R1 and R3

[Click here to view code image](#)

```
R2#trace 172.16.5.5 source GigabitEthernet 0/2
Type escape sequence to abort.
Tracing the route to 172.16.5.5

 1 172.16.1.1 * * *
 2 172.16.1.1 * * *
! Output omitted for brevity
```

```
RP/0/0/CPU0:ISP#trace 172.16.5.5
Type escape sequence to abort.
Tracing the route to 172.16.5.5

 1  192.168.1.1 3 * * *
 2  192.168.1.2 1 * * *
! Output omitted for brevity
```

## STATIC VRF ROUTES

All the previous concepts shown earlier are extended to use within a VRF.

IOS nodes use the command **ip route vrf** *vrf-name network subnet-mask {interface-type interface-number [next-hop-IP] | next-hop-IP}* for configuring a static route within a VRF context.

IOS XR's configuration adds a step to the hierarchy by declaring the specific VRF, as shown here:

### Step 1. Initialize the static routing process.

The static routing process is initialized with the command **router static**.

### Step 2. Specify the VRF context.

Specify the vrf context that the static route will apply to with the command **vrf** *vrf-name*.

### Step 3. Identify the address family.

Initialize the appropriate address family with the command **address-family** {IPv4 | IPv6} **unicast**. IPv4 static routes will use the IPv4 parameter.

### Step 4. Configure the static route.

Configure the static route with the command *network* {**subnet mask** | */prefix\_length*} {*interface-type interface-number [next-hop-IP] | next-hop-IP*}.

Service providers and enterprise networks commonly use a separate out-of-band (OOB) dedicated network for device management. Creating a management VRF on the router allows management traffic to/from the router to use the OOB network.

In [Figure 4-14](#), XR1 has a separate dedicated network in the Management VRF but still requires a static route in the global table to the 10.2.2.0/24 network.

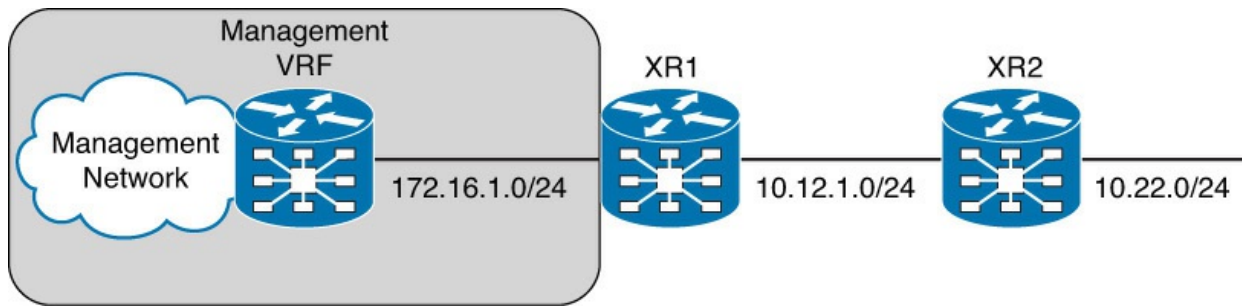


Figure 4-14 IOS XR Management VRF

Example 4-38 shows XR's configuration for the 10.2.2.0/24 network in the global table and a default route to the management router in the Management VRF.

### Example 4-38 XR1 Static Route Configuration

[Click here to view code image](#)

```
XR1
router static
 address-family ipv4 unicast
  10.2.2.0/24 10.12.1.2
 !
 vrf MANAGEMENT
  address-family ipv4 unicast
  0.0.0.0/0 172.16.1.254
 !
 !
 !
```

Example 4-39 shows that XR1 maintains two separate routing tables, one for the global table and another for the Management VRF. Notice that the IOS XR node uses a special interface dedicated for management, which is common for the hardware platforms that IOS XR runs on.

### Example 4-39 IP Routing Table for XR1's Routing Tables

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
! Output omitted for brevity
Gateway of last resort is not set

S   10.2.2.0/24 [1/0] via 10.12.1.2, 00:34:04
C   10.12.1.0/24 is directly connected, 00:34:04, GigabitEthernet0/0/0/0
L   10.12.1.1/32 is directly connected, 00:34:04, GigabitEthernet0/0/0/0
```



```

RP/0/0/CPU0:XR1#show route vrf MANAGEMENT
! Output omitted for brevity
Gateway of last resort is 172.16.1.254 to network 0.0.0.0

S* 0.0.0.0/0 [1/0] via 172.16.1.254, 00:02:43
C 172.16.1.0/24 is directly connected, 00:08:02, MgmtEth0/0/CPU0/0
L 172.16.1.1/32 is directly connected, 00:08:02, MgmtEth0/0/CPU0/0

```

Figure 4-15 provides an identical topology using IOS where R1 has a separate dedicated network in the Management VRF but still requires a static route in the global table to the 10.2.2.0/24 network.

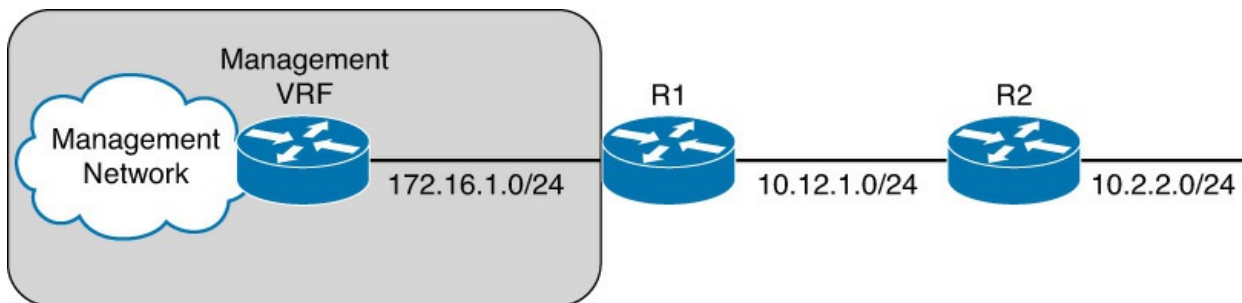


Figure 4-15 IOS Management VRF

Example 4-40 shows R1's configuration for the 10.2.2.0/24 network in the global table and a default route to the management router.

#### Example 4-40 R1 and R2 Configuration

[Click here to view code image](#)

```

R1
ip route 10.2.2.0 255.255.255.0 10.12.1.2
ip route vrf MANAGEMENT 0.0.0.0 0.0.0.0 172.16.1.254

```

Example 4-41 shows that R1 maintains two separate routing tables, one for the global table and another for the Management VRF.

#### Example 4-41 IP Routing Table for R1's Routing Tables

[Click here to view code image](#)

```

R1#show ip route | begin Gateway
Gateway of last resort is not set

10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
S 10.2.2.0/24 [1/0] via 10.12.1.2
C 10.12.1.0/24 is directly connected, GigabitEthernet0/1
L 10.12.1.1/32 is directly connected, GigabitEthernet0/1

```

```
R1#show ip route vrf MANAGEMENT | begin Gateway
Gateway of last resort is 172.16.1.254 to network 0.0.0.0

S*   0.0.0.0/0 [1/0] via 172.16.1.254
      172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
C     172.16.1.0/24 is directly connected, GigabitEthernet0/0
L     172.16.1.1/32 is directly connected, GigabitEthernet0/0
```

#### Note

This is the last topic on routing protocols in a VRF context in this book.

## SUMMARY

A router's default behavior is to provide connectivity between connected networks. Static routes may be used to provide precise control of routing behavior in a network. Multihop routing requires multiple static routes with one or more recursive lookups.

Static routes can be simple, but caveats do exist that have caused network outages for organizations of all sizes. Consider the following points:

- Only P2P interfaces should use a directly connected static route.
- Instead of recursive static routes, use a fully specified static route.
- A default static route reduces resource consumption if a router has one outbound link.
- Floating Static or default routes are useful for providing backup connectivity.
- Using a summary static routes directed at the Null0 interface prevents routing loops.

Large network topologies or topologies that constantly change will find that using only static routing burdens network engineers with tedious tasks. These networks should consider using dynamic routing protocols covered in the following chapters.

## REFERENCES IN THIS CHAPTER

Cisco IOS Software configuration guides. <http://www.cisco.com>

Cisco IOS XR Software configuration guides. <http://www.cisco.com>

## Chapter 5. EIGRP

This chapter covers the following topics:

- EIGRP inter-router communication
- EIGRP configuration
- EIGRP terminology
- EIGRP path selection
- EIGRP failure detection and convergence
- EIGRP summarization
- WAN considerations for EIGRP

Enhanced Interior Gateway Routing Protocol (EIGRP) is an enhanced distance vector routing protocol commonly found in enterprises networks. EIGRP is a derivative of the Interior Gateway Routing Protocol (IGRP) routing protocol but includes variable-length subnet masks (VLSMs) and metrics capable of supporting higher-speed interfaces. Initially, it was a Cisco proprietary protocol, but in 2013, Cisco released EIGRP to the Internet Engineering Task Force (IETF) through an informational RFC.

### EIGRP FUNDAMENTALS

EIGRP overcomes the deficiencies of other distance vector routing protocols like RIP with features such as unequal-cost load balancing, support for networks 255 hops away, and rapid-convergence features. EIGRP uses a diffusing update algorithm (DUAL) to identify network paths and provides for fast convergence using precalculated loop-free backup paths. Most distance vector routing protocols use hop count as the metric for routing decisions. Using hop count for path selection does not take into account link speed and total delay. EIGRP adds logic to the route-selection algorithm that uses factors outside of hop count.

A router can run multiple EIGRP processes. Each process operates under the context of an autonomous system, which represents a common routing domain. Routers within the same domain use the same metric calculation formula and exchange routes only with members of the same autonomous system. An EIGRP autonomous system should not be confused with a Border Gateway Protocol (BGP) autonomous system.

In Figure 5-1, EIGRP autonomous system 100 consists of XR1, R2, R3, R4, and EIGRP; autonomous system 200 consists of R3, R5, and XR7. Each EIGRP process correlates to a specific autonomous system and maintains an independent EIGRP topology table. XR1 does not have knowledge of routes from autonomous system 200 because it is different from its own autonomous system, autonomous system 100.

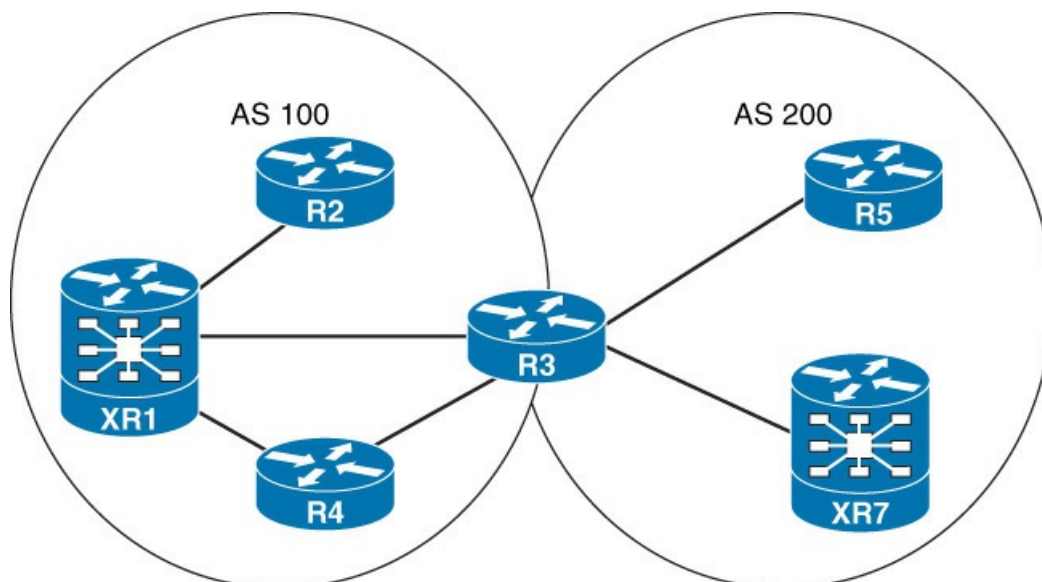


Figure 5-1 EIGRP Autonomous Systems

EIGRP uses protocol-dependent modules (PDMs) to support multiple network protocols such as IPv4, IPv6, AppleTalk, and IPX. EIGRP is written so that the PDM is responsible for the functions to handle the route selection criteria for each communication protocol. In theory, new PDMs can be written as new communication protocols are created.

## EIGRP NEIGHBORS

EIGRP does not rely on periodic advertisement of all the network prefixes in an autonomous system, which is done with routing protocols such as Routing Information Protocol (RIP), Open Shortest Path First (OSPF) Protocol, or Intermediate System-to-Intermediate System (IS-IS). EIGRP neighbors exchange the entire routing table when forming an adjacency and advertise only incremental updates as topology changes occur within a network. The neighbor adjacency table is vital for tracking neighbor status and the updates sent to each neighbor.

### Inter-Router Communication

EIGRP uses five different packet types to communicate with other routers, as shown in Table 5-1. EIGRP uses its own IP protocol number (88) and uses multicast packets where possible and unicast packets when necessary. Communication between routers is done with multicast using the group address of 224.0.0.10 or MAC address of 01:00:5e:00:00:0a when possible.

Type	Packet Name	Function
1	Hello	Used for discovery of EIGRP neighbors and for detection of when a neighbor is no longer available
2	Acknowledgment (ACK)	Packets sent to the originating router for any other EIGRP packet containing a nonzero sequence number
3	Update	Packets sent out searching for another path when an existing path fails
4	Query	Packets sent out searching for another path during convergence
5	Reply	Packets that are sent in response of a query packet

Table 5-1 EIGRP Packet Types

Note

EIGRP uses multicast packets to reduce bandwidth consumed on a link (one packet to reach multiple devices). While broadcast packets accomplish the same concept, all nodes on a network segment process broadcast packets, whereas with multicast, only nodes listening for that multicast group process the multicast packets.

EIGRP uses the Reliable Transport Protocol (RTP) to ensure that packets are delivered in order and that routers receive specific packets. A sequence number is included in all of the EIGRP packets. A sequence value of zero does not require a response from the receiving EIGRP router, and all other values require an ACK packet that includes the original sequence number.

Ensuring that packets are received makes the transport method reliable. All update, query, and reply packets are deemed reliable, whereas hello and ACK packets do not require acknowledgment and could be unreliable.

If the originating router does not receive an ACK packet from the neighbor before the retransmit timeout expires, it notifies the non-acknowledging router to stop processing its multicast packets. The originating router will send all traffic via unicast until the neighbor is fully synchronized. Upon complete synchronization, the originating router notifies the destination router to start processing multicast packets again. All unicast packets require acknowledgment. EIGRP will retry up to 16 times for each packet that requires confirmation and will reset the neighbor relationship when the neighbor reaches the retry limit of 16.

Note

In the context of EIGRP, RTP should not be confused with the Real-time Transport Protocol (RTP) used for carrying audio or video over an IP network. EIGRP's RTP protocol allows for confirmation of packets while supporting multicast. Other protocols that require reliable connection-oriented communication like TCP cannot use multicast addressing.

### Forming EIGRP Neighbors

Unlike other distance vector routing protocols, EIGRP requires a neighbor relationship to form before routes are processed and added to the Routing Information Base (RIB). Upon hearing an EIGRP hello packet, a router attempts to become neighbors. The following parameters must match for the two routers to become neighbors:

- Metric formula K values
- Primary subnet matches
- Autonomous system number (ASN) matches
- Authentication parameters

Figure 5-2 shows the process EIGRP uses for forming neighbor adjacencies.

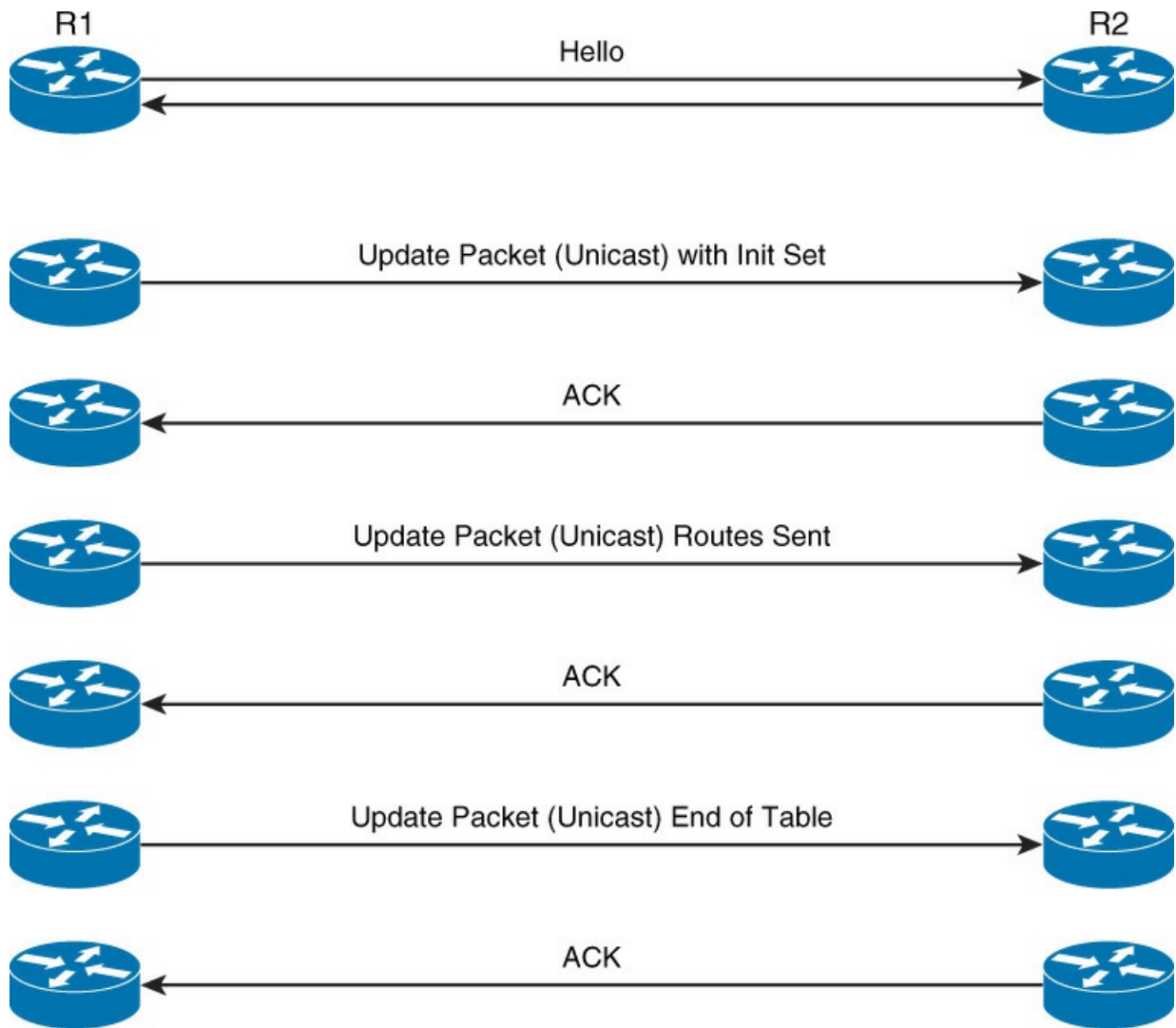


Figure 5-2 EIGRP Neighbor Adjacency Process from R1's Perspective

The following section explains the process of forming an EIGRP neighbor adjacency detail. Figure 5-3 provides a sample topology of two routers, XR1 and R2 that share a common network link, 10.12.1.0/24.

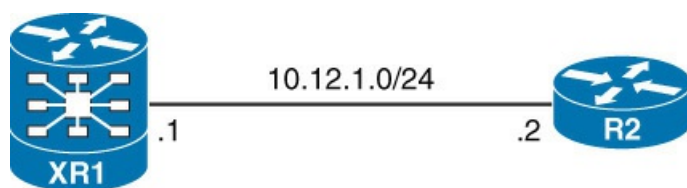


Figure 5-3 Sample Topology

1. R1 and R2 both send out an EIGRP hello packet via multicast, as shown in Figure 5-3.

Notice the highlighted fields for the source IP address for the packet, ASN, and K values used for metric calculation.

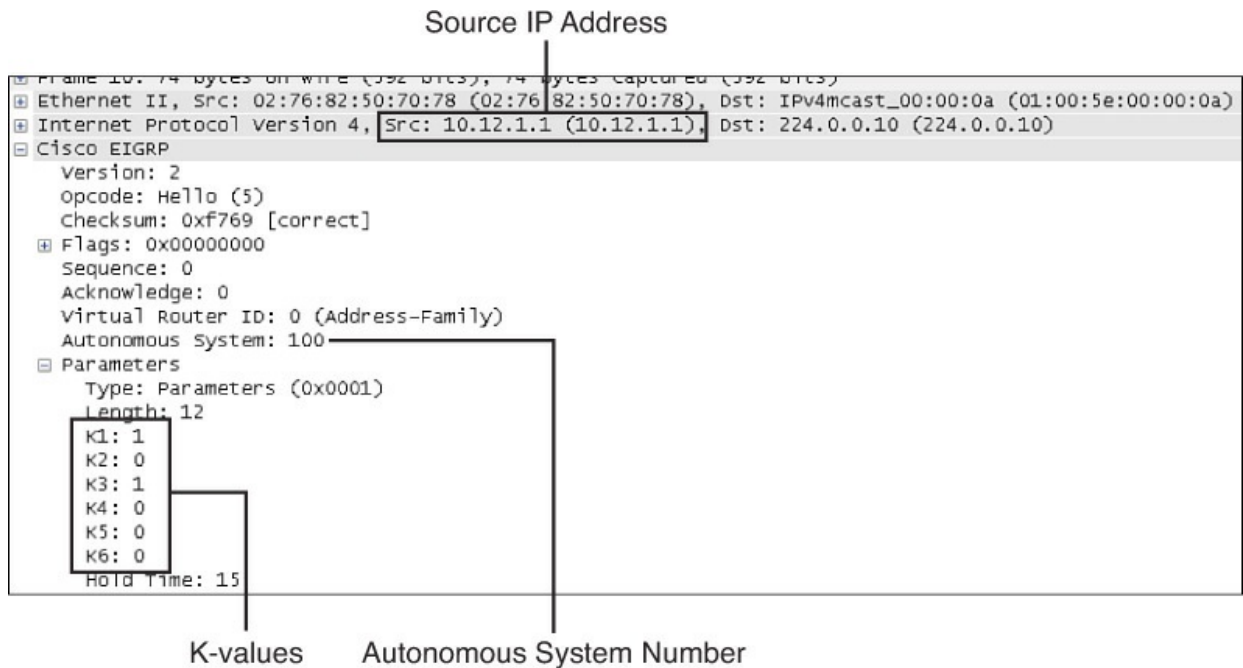


Figure 5-4 Packet Capture of EIGRP Hello

2. R1 receives R2's hello packet and checks to make sure that it passes the following checks identified above. Upon passing these checks, R1 adds R2 to its EIGRP neighbor table in the pending state.

3. R1 sends a unicast update packet to R2 with the INIT flag set. The INIT flag tells the receiving router to send all its routes to it. This is the first stage of a handshake to verify that connectivity between both routers is bidirectional. Figure 5-5 displays the unicast update packet with the INIT flag set. Notice that the sequence number is 4.

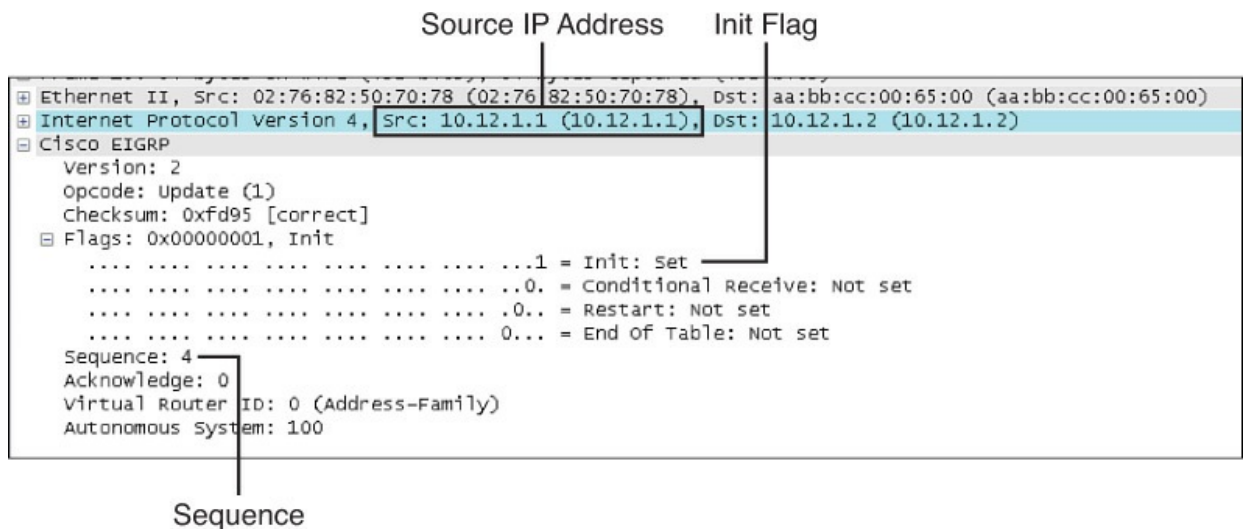


Figure 5-5 Packet Capture of EIGRP Update with INIT Flag Set

4. R2 responds to R1 via an update packet with the INIT flag set to receive all the routes from R1. This confirms that traffic is bidirectional between R1 and R2, and R1 moves R2 to an up state in the EIGRP neighbor table.

Figure 5-6 shows R2's response to R1. Notice the INIT flag is set and that the Acknowledge field is set to 4.

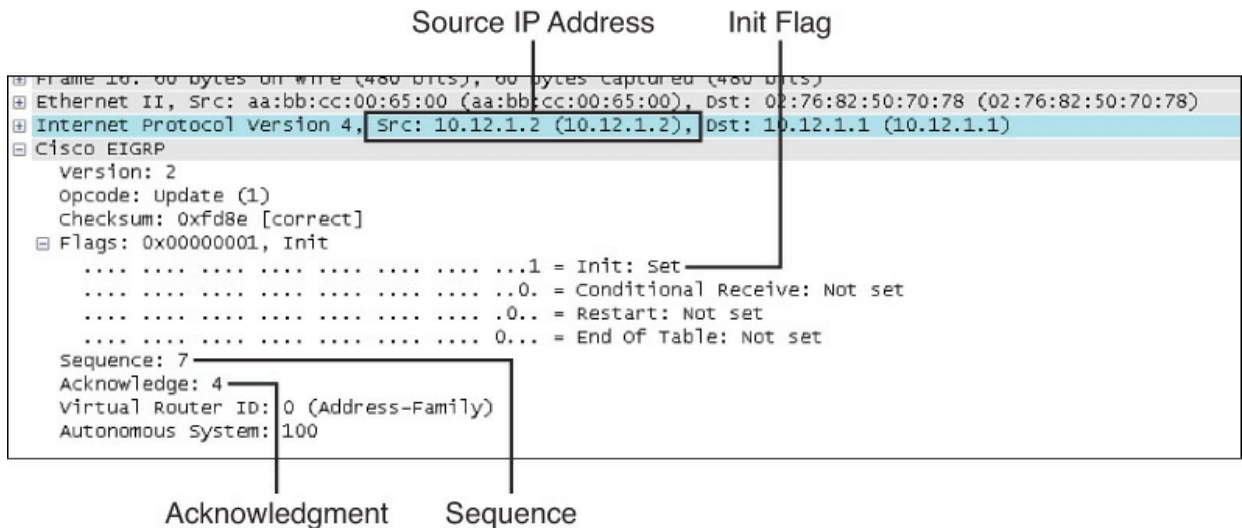


Figure 5-6 Packet of EIGRP Update with INIT Flag and Acknowledgment

5. R1 sends a unicast update packet with all the routes except those that apply to the split-horizon rule. Depending on the amount of routes, this process might take one or more packets to complete. Each packet will require a confirmation from the receiving router. The last packet will have the End of Table flag set.

Figure 5-7 displays an EIGRP update packet. Notice that this packet is the last of the updates and that R1 is advertising the 192.168.1.1/32 prefix to R2 with sequence number 5.

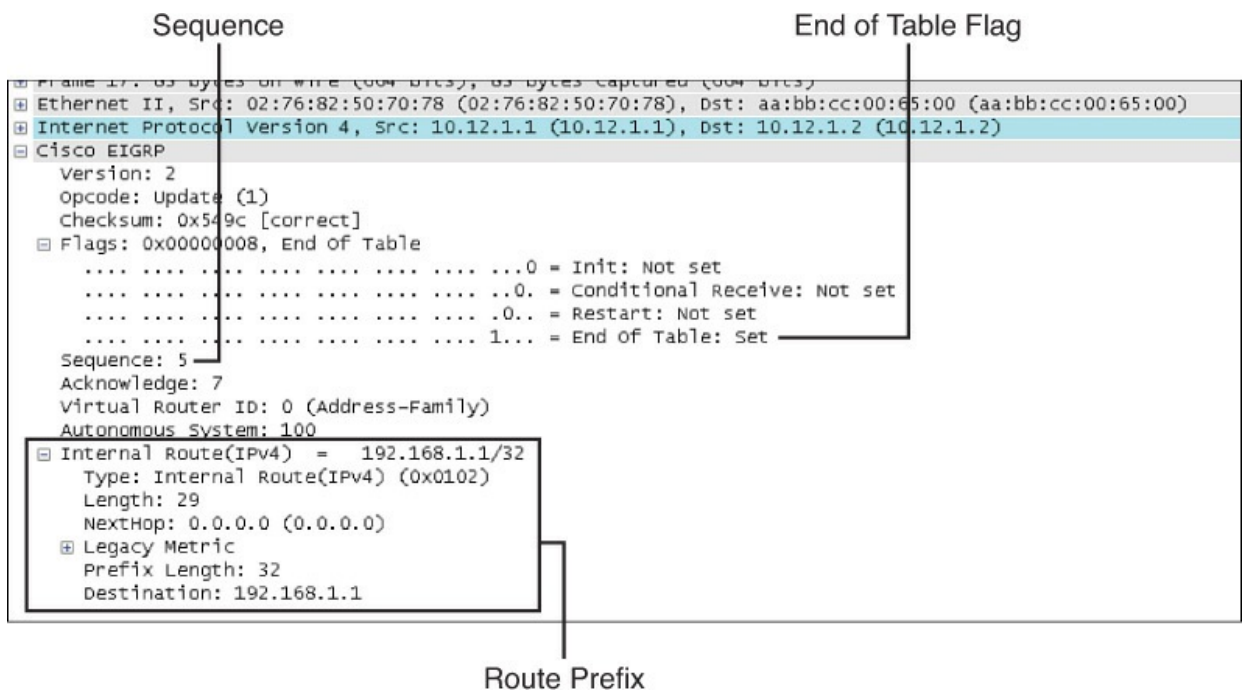


Figure 5-7 Packet Capture of EIGRP Update with Route Prefixes

6. R2 sends an ACK for every update packet, as shown in Figure 5-8. Notice that the acknowledgment is for sequence number 5.



## Source and Destination IP Address

```
Frame 157: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: aa:bb:cc:00:65:00 (aa:bb:cc:00:65:00), Dst: 02:76:82:50:70:78 (02:76:82:50:70:78)
Internet Protocol Version 4, Src: 10.12.1.2 (10.12.1.2), Dst: 10.12.1.1 (10.12.1.1)
Cisco EIGRP
  Version: 2
  Opcode: Hello (5)
  Checksum: 0xfd91 [correct]
  Flags: 0x00000000
    ...0 = Init: Not set
    ...0. = Conditional Receive: Not set
    ...0.. = Restart: Not set
    ...0... = End Of Table: Not set
  Sequence: 0
  Acknowledge: 5
  Virtual Router ID: 0 (Address-Family)
  Autonomous System: 100
```

Acknowledgment

Figure 5-8 Packet Capture of EIGRP Update with Advertised Networks

### Note

The same process occurs from R2's perspective at the same time.

## CLASSIC EIGRP AUTONOMOUS SYSTEM CONFIGURATION

The Classic EIGRP autonomous system configuration process on IOS and IOS XR routers is different. On IOS routers, most of the configuration is under the EIGRP process, but some interface parameters are configured directly on the interface configuration submode. IOS XR's classic EIGRP autonomous system configuration is contained completely within the EIGRP process. IOS and IOS XR use the command **router eigrp as-number** to identify the ASN and initialize the EIGRP process.

### Note

IOS XR version 5.1.0 and later supports multiple EIGRP instances.

### IOS network Statement

Within IOS, the **network** statement identifies the interfaces that EIGRP will use. The **network** statement uses a wildcard mask, which allows the configuration to be as specific or ambiguous as necessary.

The syntax for the **network** statement, which exists under the EIGRP process, is as follows:

[Click here to view code image](#)

```
network ip-address wildcard-mask
```

A common misconception is that the **network** statement adds the networks to the EIGRP topology table. In reality, the **network** statement identifies the interface to enable EIGRP on and adds the interface's connected network to the EIGRP topology table. EIGRP then advertises the topology table to other routers in the EIGRP autonomous system.

EIGRP does not add an interface's secondary connected network into the topology table. To install secondary connected networks into the EIGRP routing table, they need to be redistributed into the EIGRP process. [Chapter 13, "Route Redistribution,"](#) provides additional coverage of

router redistribution.

Note

The wildcard mask can be omitted, which then enables interfaces that fall within the classful boundaries for that **network** statement.

To help illustrate the concept of the wildcard mask, [Table 5-2](#) provides a set of IP addresses and interfaces for a router. The following examples will provide configurations to match a specific scenario.

IOS Interface	IOS XR Interface	IP Address
Gigabit Ethernet 0/0	Gigabit Ethernet 0/0/0/0	10.0.0.10/24
Gigabit Ethernet 0/1	Gigabit Ethernet 0/0/0/1	10.0.10.10/24
Gigabit Ethernet 0/2	Gigabit Ethernet 0/0/0/2	192.0.0.10/24
Gigabit Ethernet 0/3	Gigabit Ethernet 0/0/0/3	192.10.0.10/24

**Table 5-2** Table of Sample Interface and IP Addresses

The configuration in [Example 5-1](#) enables EIGRP only on interfaces that explicitly match the IP addresses in [Table 5-2](#).

### Example 5-1 EIGRP Configuration with Explicit IP Addresses

[Click here to view code image](#)

```
router eigrp 1
  network 10.0.0.10 0.0.0.0
  network 10.0.10.10 0.0.0.0
  network 192.0.0.10 0.0.0.0
  network 192.10.0.10 0.0.0.0
```

[Example 5-2](#) shows the EIGRP configuration using network statements that match the subnets used in [Table 5-2](#). Setting the last octet of the IP address to 0 and changing the wildcard mask to 255, the **network** statements match all IP addresses within the /24 network.

### Example 5-2 EIGRP Configuration with Explicit Subnet

[Click here to view code image](#)

```
router eigrp 1
  network 10.0.0.0 0.0.0.255
  network 10.0.10.0 0.0.0.255
  network 192.0.0.0 0.0.0.255
  network 192.10.0.0 0.0.0.255
```

[Example 5-3](#) shows the EIGRP configuration using **network** statements for interfaces that are within the 10.0.0.0/8 or 192.0.0.0/8 network ranges.

### Example 5-3 EIGRP Configuration with Large Subnet Ranges

[Click here to view code image](#)

```
router eigrp 1
  network 10.0.0.0 0.255.255.255
  network 192.0.0.0 0.255.255.255
```

Example 5-4 shows the EIGRP configuration to enable all interfaces with EIGRP.

### Example 5-4 EIGRP Configuration for All Interfaces

[Click here to view code image](#)

```
router eigrp 1
  network 0.0.0.0 255.255.255.255
```

#### Note

A key point with wildcard **network** statements is that large ranges simplify configuration but could enable EIGRP on unintended interfaces. [Chapter 2, "IPv4 Addressing,"](#) explains identifying wildcard masks in more detail.

### IOS XR

IOS XR's EIGRP configuration is much simpler. After defining the EIGRP process and ASN, the appropriate address family is initialized. IPv4 networks use the **address-family ipv4** command. Enabling EIGRP on an interface involves listing the interfaces underneath the address family with the command **interface interface-type interface-number**.

Example 5-5 shows the IOS XR EIGRP configuration using interfaces from [Table 5-2](#).

### Example 5-5 Sample EIGRP Configuration for IOS XR

[Click here to view code image](#)

```
Router eigrp 1
  address-family ipv4 unicast
    interface GigabitEthernet 0/0/0/0
    interface GigabitEthernet 0/0/0/1
    interface GigabitEthernet 0/0/0/2
    interface GigabitEthernet 0/0/0/3
```

#### Note

Unlike IOS, IOS XR adds secondary connected networks to the EIGRP topology table without the need of route redistribution.

### Passive Interfaces

Some network topologies require advertising a network segment into EIGRP but need to prevent neighbors from forming adjacencies with other routers on that segment. Example scenarios involve advertising access layer networks in a campus topology.

The solution is to place the EIGRP interface into a passive state. Passive EIGRP interfaces do not

send out or process EIGRP hellos, which prevents EIGRP from forming an adjacency on that interface.

Configuring an EIGRP interface as passive uses the command **passive-interface** *interface-type interface-number* under the EIGRP process for IOS nodes. IOS-based nodes allow for all interfaces to be set as passive by default with the command **passive-interface default**. The command **no passive-interface** *interface-type interface-number* allows an interface to process EIGRP packets, preempting the **global passive-interface default** configuration.

IOS XR nodes can set an EIGRP interface as passive with the command **passive-interface** under the interface in the EIGRP process. IOS XR does not support a passive interface default method.

### Sample Topology and Configuration

Figure 5-9 provides a sample topology for demonstrating EIGRP configurations on IOS and IOS-XR nodes.

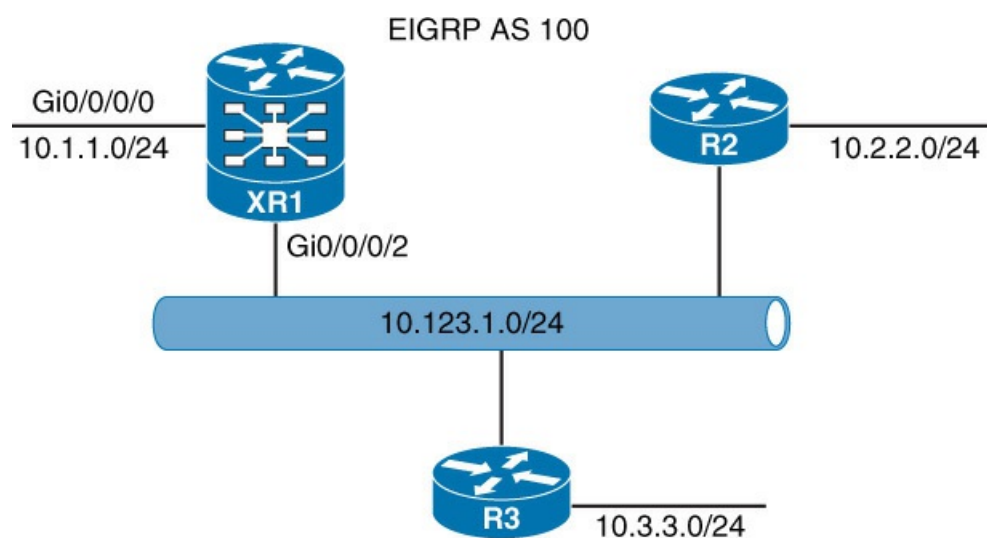


Figure 5-9 EIGRP Sample Topology

XR1, R2, and R3 will enable EIGRP on all their interfaces. R2 will configure EIGRP using specific network interface addresses, and R3 will enable EIGRP on all network interfaces. All interfaces except for those connected to the 10.123.1.0/24 network will be set to passive. R2 will explicitly state the passive interfaces, and R3 will use **passive-interface default** as part of its configuration. [Example 5-6](#) provides the EIGRP configuration for all three routers.

### Example 5-6 Sample EIGRP Configuration

[Click here to view code image](#)

**XR1**

```
interface Loopback0
  ipv4 address 192.168.1.1 255.255.255.255
!
interface GigabitEthernet0/0/0/0
  ipv4 address 10.1.1.1 255.255.255.0
!
interface GigabitEthernet0/0/0/2
  ipv4 address 10.123.1.1 255.255.255.0
!
router eigrp 100
  address-family ipv4
    interface Loopback0
      passive-interface
    !
    interface GigabitEthernet0/0/0/0
      passive-interface
    !
    interface GigabitEthernet0/0/0/2
```

**R2**

```
interface Loopback0
  ip address 192.168.1.1 255.255.255.255
!
interface GigabitEthernet0/0
  ip address 10.123.1.2 255.255.255.0
!
interface GigabitEthernet0/1
  ip address 10.2.2.2 255.255.255.0
!
router eigrp 100
  network 10.2.2.2 0.0.0.0
  network 10.123.1.2 0.0.0.0
  network 192.168.2.2 0.0.0.0
  passive-interface Loopback0
  passive-interface GigabitEthernet0/1
```

**R3**

```
interface Loopback0
  ip address 192.168.3.3 255.255.255.255
!
interface GigabitEthernet0/0
  ip address 10.123.1.3 255.255.255.0
!
interface GigabitEthernet0/1
  ip address 10.3.3.3 255.255.255.0
!
router eigrp 100
  network 0.0.0.0 255.255.255.255
  passive-interface default
  no passive-interface GigabitEthernet0/0
```

## Confirmation of Interfaces

Upon configuring EIGRP, it is a good practice to verify that only the intended interfaces are running EIGRP. The command **show ip eigrp interface** [interface-type interface-number] [detail] shows active EIGRP interfaces on IOS nodes. The command **show eigrp interface** [interface-type interface-number] [detail] shows active and passive interfaces on IOS XR nodes. Both commands allow for the identification of a specific interface or appending the optional **detail** keyword provides additional information such as authentication, EIGRP timers, split horizon, and various packet counts.

[Example 5-7](#) shows XR1 and R2's nondetailed EIGRP interface verification.

### Example 5-7 Verification of EIGRP Interfaces

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show eigrp interfaces

IPv4-EIGRP interfaces for AS(100)

Interface      Peers    Xmit Queue  Mean   Pacing Time  Multicast  Pending
              Un/Reliable SRTT      Un/Reliable  Flow Timer  Routes
Gi0/0/0/0      0         0/0         0      640/640      0          0
Gi0/0/0/2      2         0/0         106    0/10         532        0
Lo0            0         0/0         0      640/640      0          0
```

```
R2#show ip eigrp interfaces
EIGRP-IPv4 Interfaces for AS(100)

Interface      Peers    Xmit Queue  PeerQ   Mean   Pacing Time  Multicast  Pending
              Un/Reliable Un/Reliable SRTT      Un/Reliable  Flow Timer  Routes
Gi0/0          2         0/0         0/0     7      0/0          50         0
```

[Table 5-3](#) provides a brief explanation to the key fields shown with the EIGRP interfaces.

Field	Description
Interface	Interfaces running EIGRP.
Peers	Number of peers detected on that interface.
Xmit Queue Un/Reliable	Number of unreliable/reliable packets remaining in the transmit queue. A value of 0 indicates a stable network.
Mean SRTT	Average time for a packet to be sent to a neighbor and a reply from that neighbor received in milliseconds.
Multicast Flow Timer	Maximum time (seconds) that the router sent multicast packets.
Pending Routes	Number of routes in the transmit queue that need to be sent.

**Table 5-3** EIGRP Interface Fields

Notice that in [Example 5-7](#), the IOS nodes do not display the passive interfaces. Passive interfaces appear under the output from the command **show ip protocols** on IOS nodes or with the command **show protocols ipv4** on IOS XR nodes. [Example 5-8](#) provides sample output, which does show the passive interfaces.

## Example 5-8 IP Protocols Output

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show protocols ipv4

Routing Protocol: EIGRP, instance 100
  Default context AS: 100, Router ID: 192.168.1.1
  Address Family: IPv4
  Default networks not flagged in outgoing updates
  Default networks not accepted from incoming updates
  Distance: internal 90, external 170
  Maximum paths: 4
  EIGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  EIGRP maximum hopcount 100
  EIGRP maximum metric variance 1
  EIGRP NSF: enabled
  NSF-aware route hold timer is 480s
  NSF signal timer is 20s
  NSF converge timer is 300s
  Time since last restart is 18:23:34
  SIA Active timer is 180s
  Interfaces:
    GigabitEthernet0/0/0/2
    Loopback0, passive
    GigabitEthernet0/0/0/0, passive
```

```

R2#show ip protocols
*** IP Routing is NSF aware ***

Routing Protocol is "eigrp 100"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP-IPv4 Protocol for AS(100)
    Metric weight K1=1, K2=0, K3=1, K4=0, K5=0
    NSF-aware route hold timer is 240
    Router-ID: 192.168.2.2
    Topology : 0 (base)
      Active Timer: 3 min
      Distance: internal 90 external 170
      Maximum path: 4
      Maximum hopcount 100
      Maximum metric variance 1

  Automatic Summarization: disabled
  Maximum path: 4
  Routing for Networks:
    10.2.2.2/32
    10.123.1.2/32
    192.168.2.2/32
  Passive Interface(s):
    GigabitEthernet0/1
    Loopback0
  Routing Information Sources:
    Gateway         Distance      Last Update
    10.123.1.1       90           18:15:52
    10.123.1.3       90           18:16:35
  Distance: internal 90 external 170

```

#### Note

The commands **show ip protocols** and **show protocols ipv4** provides information on IP routing protocols configured on a router. IS-IS is excluded because it is an OSI protocol and not a TCP/IP protocol.

### Verification of EIGRP Neighbor Adjacencies

Each EIGRP process maintains a table of neighbors to ensure that they are alive and processing updates accordingly. Without keeping track of a neighbor state, an autonomous system could contain incorrect data and potentially route traffic improperly. EIGRP must form a neighbor relationship before a router advertises update packets containing network prefixes.

To view the EIGRP neighbor table, the command **show ip eigrp neighbor** is used for IOS nodes, and the command **show eigrp neighbors** is used for IOS XR-based nodes. [Example 5-9](#) provides sample output for these commands.

### Example 5-9 EIGRP Neighbor Confirmation

[Click here to view code image](#)



```
RP/0/0/CPU0:XR1#show eigrp neighbors
```

```
IPv4-EIGRP neighbors for AS(100) vrf default
```

H	Address	Interface	Hold Uptime (sec)	SRTT (ms)	RTO	Q Cnt	Seq Num
1	10.123.1.2	Gi0/0/0/2	12 00:00:26	1275	5000	0	6
0	10.123.1.3	Gi0/0/0/2	12 00:00:26	2	200	0	6

```
R2#show ip eigrp interfaces
```

```
EIGRP-IPv4 Neighbors for AS(100)
```

H	Address	Interface	Hold Uptime (sec)	SRTT (ms)	RTO	Q Cnt	Seq Num
1	10.123.1.1	Gi0/0	12 02:09:39	4	100	0	7
0	10.123.1.2	Gi0/0	12 02:10:26	1	100	0	8

Table 5-4 provides a brief explanation to the key fields shown in Example 5-9.

Field	Description
Address	IP address of the EIGRP neighbor
Interface	Interface the neighbor was detected on
Holdtime	Time left to receive a packet from this neighbor to ensure it is still alive
SRTT	Time for a packet to be sent to a neighbor and a reply from that neighbor received in milliseconds
RTO	Timeout for retransmission (waiting for ACK)
Q Cnt	Number of packets (update/query/reply) in queue for sending
Seq Num	Sequence number that was last received from this router

Table 5-4 EIGRP Neighbor Columns

### Display of Installed EIGRP Routes

EIGRP routes that are installed into the RIB can be seen with the command **show ip route eigrp** on IOS-based nodes and **show route eigrp** on IOS XR-based nodes.

EIGRP routes originating within the autonomous system have an administrative distance (AD) of 90 and are displayed in the route table as a *D* for the protocol. Routes that originate from outside of the autonomous system are external EIGRP routes. External EIGRP routes have an AD of 170 and are displayed as *DEX*. Placing external EIGRP routes into the RIB with a higher AD acts as a loop-prevention mechanism.

Example 5-10 displays the EIGRP routes from the sample topology in Figure 5-7. The metric for the selected route is the second number in brackets.

### Example 5-10 EIGRP Routes for XR1, R2, and R3

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route eigrp
```

```
D 10.2.2.0/24 [90/3072] via 10.123.1.2, 01:30:12, GigabitEthernet0/0/0/2
D 10.3.3.0/24 [90/3072] via 10.123.1.3, 01:30:12, GigabitEthernet0/0/0/2
D 192.168.2.2/32 [90/130816] via 10.123.1.2, 18:22:51, GigabitEthernet0/0/0/2
D 192.168.3.3/32 [90/130816] via 10.123.1.3, 18:23:34, GigabitEthernet0/0/0/2
```

```
R2#show ip route eigrp
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
D 10.1.1.0/24 [90/3072] via 10.123.1.1, 00:13:15, GigabitEthernet0/0
D 10.3.3.0/24 [90/3072] via 10.123.1.3, 01:32:43, GigabitEthernet0/0
192.168.1.0/32 is subnetted, 1 subnets
D 192.168.1.1 [90/130816] via 10.123.1.1, 18:25:51, GigabitEthernet0/0
192.168.3.0/32 is subnetted, 1 subnets
D 192.168.3.3 [90/130816] via 10.123.1.3, 18:24:23, GigabitEthernet0/0
```

```
R3#show ip route eigrp
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
D 10.1.1.0/24 [90/3072] via 10.123.1.1, 00:13:59, GigabitEthernet0/0
D 10.2.2.0/24 [90/3072] via 10.123.1.2, 01:33:27, GigabitEthernet0/0
192.168.1.0/32 is subnetted, 1 subnets
D 192.168.1.1 [90/130816] via 10.123.1.1, 18:24:40, GigabitEthernet0/0
192.168.2.0/32 is subnetted, 1 subnets
D 192.168.2.2 [90/130816] via 10.123.1.2, 18:23:56, GigabitEthernet0/0
```

## ROUTER ID

The router ID (RID) is a 32-bit number that uniquely identifies an EIGRP router. EIGRP uses the RID as a loop-prevention mechanism for external routes. The RID can be set manually or dynamically but should be unique for each EIGRP process. Dynamic allocation logic varies from IOS to IOS XR:

- **IOS:** EIGRP uses the highest IPv4 address of any *up* loopback interfaces. If there is not any up loopback interfaces, the highest IPv4 address of any active up physical interfaces becomes the RID when the EIGRP process initializes.

- **IOS XR:** EIGRP uses the IPv4 address of the lowest *up* loopback interface. If there is not any up loopback interfaces, the lowest interface IPv4 address of any active up physical interface becomes the RID when the EIGRP process initializes.

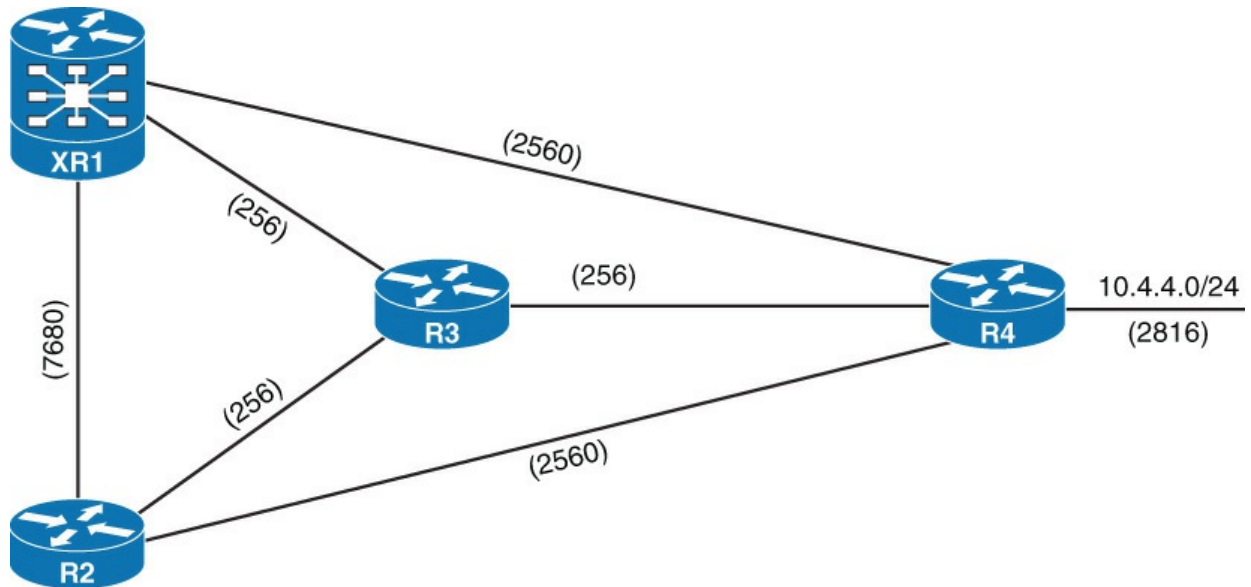
IPv4 addresses are commonly used for the RID because they are 32 bits and are maintained in dotted-decimal format. The command **eigrp router-id router-id** is used on IOS nodes and the command **router-id router-id** on IOS XR nodes under the address family within the EIGRP process.

Note

RIDs normally match a specific loopback address on the router. It is common for the loopback addresses to come from a block of network addresses.

## EIGRP TERMINOLOGY

This section explains some of the core concepts within EIGRP. [Figure 5-10](#) will be used as a reference topology for XR1 calculating the best path to the 10.4.4.0/24 network.



**Figure 5-10** EIGRP Reference Topology

[Table 5-5](#) contains key terms, definitions, and their correlation to [Figure 5-10](#).

Term	Definition
Successor Route	The route with the lowest path metric to reach a destination. The successor route for XR1 to reach 10.4.4.0/24 on R4 is XR1-R3-R4.
Successor	The first next-hop router for the successor route. The successor for 10.4.4.0/24 is R3.
Feasible Distance (FD)	The metric value for the lowest-metric path to reach a destination. The feasible distance is calculated locally using the formula shown in the “Path Metric Calculation” section later in this chapter. The FD calculated by XR1 for the 10.4.4.0/24 network is 3328. (256 + 256 + 2816)
Reported Distance (RD)	Distance reported by a router to reach a prefix. The reported distance value is the FD for the advertising router. R3 advertises the 10.4.4.0/24 prefix with an RD of 3072. R4 advertises the 10.4.4.0/24 to XR1 and R2 with an RD of 2816.
Feasibility Condition	For a route to be considered a backup route, the RD received for that route must be less than the FD calculated locally. This logic guarantees a loop-free path.
Feasible Successor	A route with that satisfies the feasibility condition is maintained as a backup route. The feasibility condition ensures that the backup route is loop free. The route XR1-R4 is the feasible successor because the RD of 2816 is lower than the FD 3328 for the XR1-R3-R4 path.

Table 5-5 EIGRP Terminology

## TOPOLOGY TABLE

EIGRP contains a topology table that makes it different from a “true” distance vector routing protocol. EIGRP’s topology table is a vital component to DUAL and contains information to identify loop-free backup routes. The topology table contains all the network prefixes advertised within an EIGRP autonomous system. Each entry in the table contains the following

- Network prefix
- Nearby EIGRP neighbors that have advertised that prefix
- Metrics from each neighbor (reported distance, hop count)
- Values used for calculating the metric (load, reliability, total delay, minimum bandwidth)

The command **show ip eigrp topology [all-links]** provides the topology table on IOS nodes, and the command **show eigrp topology [all-links]** provides equivalent output for IOS XR nodes. By default only successor and feasible successor routes are displayed, but the optional **all-links** keyword will show the paths that did not pass the feasibility condition.

Figure 5-11 displays the topology table for XR1 and R2 from Figure 5-10. The following section focuses on the 10.4.4.0/24 network when explaining the topology table.

```
RP/0/0/CPU0:XR1#show eigrp topology
! Output omitted for brevity
IPv4-EIGRP Topology Table for AS(100)/ID(192.168.1.1)
```

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,  
r - reply Status, s - sia Status

```
P 10.14.1.0/24, 1 successors,FD is 5120
  via Connected, GigabitEthernet0/0/0/4
P 10.13.1.0/24, 1 successors,FD is 2816
  via Connected, GigabitEthernet0/0/0/3
P 10.12.1.0/24, 1 successors,FD is 2816
  via Connected, GigabitEthernet0/0/0/2
P10.4.4.0/24, 1successors, FD is 3328
  via 10.13.1.3 (3328/3072), GigabitEthernet0/0/0/3
  via 10.14.1.4 (5376/2816), GigabitEthernet0/0/0/4
```

```
R2#show ip eigrp topology
! Output omitted for brevity
EIGRP-IPv4 Topology Table for AS(100)/ID(192.168.2.2)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
r - reply Status, s - sia Status
```

```
P 10.34.1.0/24, 1 successors, FD is 3072
  via 10.23.1.3 (3072/2816), GigabitEthernet0/1
  via 10.24.1.4 (5376/2816), GigabitEthernet0/2
P 10.12.1.0/24, 1 successors, FD is 2816
  via Connected, GigabitEthernet0/0
P 10.24.1.0/24, 1 successors, FD is 5120
  via Connected, GigabitEthernet0/2
P 10.3.3.0/24, 1 successors, FD is 3072
  via 10.23.1.3 (3072/2816), GigabitEthernet0/1
P 10.4.4.0/24, 1 successors, FD is 3328
  via 10.23.1.3 (3328/3072), GigabitEthernet0/1
  via 10.24.1.4 (5376/2816), GigabitEthernet0/2
```

Figure 5-11 EIGRP Topology Output

Upon examining the network 10.4.4.0/24, notice that XR1 and R2 calculate a FD of 3,328 for the successor route. The successor (upstream router) advertises the successor route with an RD of 3072. The second path entry has a metric of 5376 and has an RD of 2816. Because 2816 is less than 3072, the second entry passes the feasibility condition and classifies the second entry as the feasible successor for prefix.

The 10.4.4.0/24 route is passive (P), which means that the topology is stable. During a topology change, routes will go into an active (A) state when computing a new path.

## PATH METRIC CALCULATION

Metric calculation is a critical component for any routing protocol. EIGRP uses multiple factors to calculate the metric for a path. Metric calculation uses *bandwidth* and *delay* by default but can include interface load and reliability, too.

The formula shown in Figure 5-12 illustrates the EIGRP classic metric formula.

$$\text{Metric} = \left[ \left( K_1 * \text{BW} + \frac{K_2 * \text{BW}}{256 - \text{Load}} + K_3 * \text{Delay} \right) * \frac{K_5}{K_4 + \text{Reliability}} \right]$$

Figure 5-12 EIGRP Classic Metric Formula

EIGRP uses *K values* to define which factors that the formula uses and the associated impact with that factor when calculating the metric. A common misconception is that the K values directly apply to bandwidth, load, delay, or reliability; this is not accurate. For example,  $K_1$  and  $K_2$  both reference bandwidth (BW).

BW represents the slowest link in the path scaled to a 10 gigabit per second link ( $10^7$ ). Link speed is collected from the configured interface bandwidth on an interface. Delay is the total measure of delay in the path measured in tenths of microseconds ( $\mu\text{s}$ ).

The EIGRP formula is based on the IGRP metric formula, except the output is multiplied by 256 to change the metric from 24 bits to 32 bits. Taking these definitions into consideration, the formula for EIGRP is as shown in Figure 5-13.

$$\text{Metric} = 256 * \left[ \left( K_1 * \frac{10^7}{\text{Min. Bandwidth}} + \frac{K_2 * \text{Min. Bandwidth}}{256 - \text{Load}} + \frac{K_3 * \text{Total Delay}}{10} \right) * \frac{K_5}{K_4 + \text{Reliability}} \right]$$

Figure 5-13 EIGRP Classic Metric Formula with Definitions

By default  $K_1$  and  $K_3$  have a value of 1, and  $K_2$ ,  $K_4$ , and  $K_5$  are set to 0. Figure 5-14 places default K values into the formula and then shows a streamlined version of the formula.

$$\text{Metric} = 256 * \left[ \left( 1 * \frac{10^7}{\text{Min. Bandwidth}} + \frac{0 * \text{Min. Bandwidth}}{256 - \text{Load}} + \frac{1 * \text{Total Delay}}{10} \right) * \frac{0}{0 + \text{Reliability}} \right]$$



$$\text{Metric} = 256 * \left( \frac{10^7}{\text{Min. Bandwidth}} + \frac{\text{Total Delay}}{10} \right)$$

Figure 5-14 EIGRP Classic Metric Formula with Default K Values

The EIGRP update packet includes path attributes associated with each prefix. The EIGRP path attributes can include hop count, cumulative delay, minimum bandwidth link speed, and RD. The attributes are updated each hop along the way allowing each router to independently identify the shortest path.

Figure 5-15 shows EIGRP updates for the 10.1.1.0/24 prefix propagating through the autonomous system. Notice that the hop count increments, minimum bandwidth decreases, total delay increases, and the RD changes with each EIGRP update.

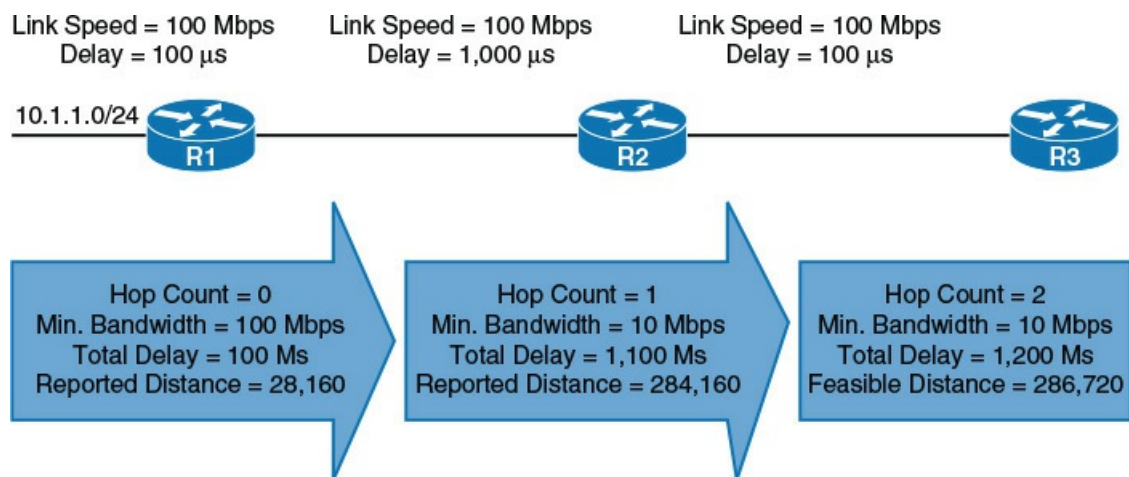


Figure 5-15 EIGRP Attribute Propagation

Table 5-6 shows some of the common network types, link speeds, delay, and EIGRP metric using the streamlined formula from Figure 5-14.

Interface Type	Link Speed (Kbps)	Classic Delay	Metric
Serial	64	20,000 μs	40,512,000
T1	1544	20,000 μs	2,170,031
Ethernet	10,000	1,000 μs	281,600
Fast Ethernet	100,000	100 μs	28,160
Gigabit Ethernet	1,000,000	10 μs	2816
10-Gigabit Ethernet	10,000,000	10 μs	512

Table 5-6 Default EIGRP Interface Metrics for Classic Metrics

Using the topology from Figure 5-10, the metric from XR1 and R2 for the 10.4.4.0/24 network can be calculated using the formula in Figure 5-16. The link speed for both routers is 1 Gbps, and the total delay is 30 μs (10 μs for the 10.4.4.0/24 link, 10 μs for the 10.34.1.0/24 link, and 10 μs for the link to R3).

$$\text{Metric} = 256 * \left( \frac{10^7}{1,000,000} + \frac{30}{10} \right) = 3,328$$

Figure 5-16 Lowest Link Speed and Cumulative Delay

If you are unsure of the EIGRP metrics, the parameters for the formula can be queried directly from EIGRP's topology table. On IOS nodes, the command **show ip eigrp topology network/prefix-length** is used with the equivalent command **show eigrp topology network/prefix-length** for IOS XR nodes.

Example 5-11 shows XR1 and R2's topology table output for the 10.4.4.0/24 network. Notice that the output includes the successor route, any feasible successor paths, and the EIGRP state for the prefix. Each path contains the EIGRP attributes minimum bandwidth, total delay, interface reliability, load, and hop count.

### Example 5-11 EIGRP Topology for a Specific Prefix

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show eigrp topology 10.4.4.0/24
```

```
IPv4-EIGRP autonomous system(100): Topology entry for 10.4.4.0/24
```

```
State is Passive, Query origin flag is 1, 1 Successor(s), FD is 3328
```

```
Routing Descriptor Blocks:
```

```
10.13.1.3 (GigabitEthernet0/0/0/3), from 10.13.1.3, Send flag is 0x0
```

```
Composite metric is (3328/3072), Route is Internal
```

```
Vector metric:
```

```
Minimum bandwidth is 1000000 Kbit
```

```
Total delay is 30 microseconds
```

```
Reliability is 255/255
```

```
Load is 255/255
```

```
Minimum MTU is 1500
```

```
Hop count is 2
```

```
10.14.1.4 (GigabitEthernet0/0/0/4), from 10.14.1.4, Send flag is 0x0
```

```
Composite metric is (5376/2816), Route is Internal, not in RIB
```

```
Vector metric:
```

```
Minimum bandwidth is 1000000 Kbit
```

```
Total delay is 110 microseconds
```

```
Reliability is 255/255
```

```
Load is 255/255
```

```
Minimum MTU is 1500
```

```
Hop count is 1
```

```
R2#show ip eigrp topology 10.4.4.0/24
```

```
EIGRP-IPv4 Topology Entry for AS(100)/ID(192.168.2.2) for 10.4.4.0/24
```

```
State is Passive, Query origin flag is 1, 1 Successor(s), FD is 3328
```

```
Descriptor Blocks:
```

```
10.23.1.3 (GigabitEthernet0/1), from 10.23.1.3, Send flag is 0x0
```

```
Composite metric is (3328/3072), route is Internal
```

```
Vector metric:
```

```
Minimum bandwidth is 1000000 Kbit
```

```
Total delay is 30 microseconds
```

```
Reliability is 255/255
```

```
Load is 255/255
```

```
Minimum MTU is 1500
```

```
Hop count is 2
```

```
Originating router is 192.168.4.4
```

```
10.24.1.4 (GigabitEthernet0/2), from 10.24.1.4, Send flag is 0x0
```

```
Composite metric is (5376/2816), route is Internal
```

```
Vector metric:
```

```
Minimum bandwidth is 1000000 Kbit
```

```
Total delay is 110 microseconds
```

```
Reliability is 255/255
```

```
Load is 255/255
```

```
Minimum MTU is 1500
```

```
Hop count is 1
```

```
Originating router is 192.168.4.4
```

## Custom K Values

If the default metric calculations are insufficient, they can be changed to modify the path metric formula. K values for the path metric formula are set with the command **metric weights TOS K1 K2 K3 K4 K5** under the EIGRP process. The TOS value always uses a value of 0.



#### Note

To ensure consistent routing logic within an EGRP autonomous system, the K values must match between EGRP neighbors to form an adjacency and exchange routes. The K values are included as part of the EGRP hello packet.

The K value settings are shown by looking at the IP protocols configuration on the router. [Example 5-12](#) provides verification of K values for our sample topology. Notice that both routers are using the default K values.

### Example 5-12 Verification of EGRP K Values

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show protocols ipv4 | include EGRP
```

```
Routing Protocol: EGRP, instance 100
  EGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  EGRP maximum hopcount 100
  EGRP maximum metric variance 1
  EGRP NSF: enabled
```

```
R2#show ip protocols | include EGRP|weight
```

```
EGRP-IPv4 Protocol for AS(100)
  Metric weight K1=1, K2=0, K3=1, K4=0, K5=0
```

#### Note

If load and reliability are used, the initial values are set during peer establishment and do not change dynamically with the router.

### Interface Delay Settings

If you do not remember the delay values from [Table 5-6](#), the values can be dynamically queried. The command **show interface interface-type interface-number** provides the EGRP interface delay in microseconds after the DLY field on IOS nodes.

The command **show eigrp interface interface-type interface-number detail** provides the EGRP interface delay in tens of microseconds on IOS XR nodes. Multiply the value by 10 to find the exact delay value.

[Example 5-13](#) provides sample output of the commands. Both interfaces have a delay of 10, but notice the output from XR1 provides a value of 1, which needs to be multiplied by 10.

### Example 5-13 Verification of EGRP Interface Delay

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show eigrp interfaces gi0/0/0/3 detail
```

```
IPv4-EIGRP interfaces for AS(100)
```

Interface	Peers	Xmit Queue Un/Reliable	Mean SRTT	Pacing Time Un/Reliable	Multicast Flow Timer	Pending Routes
Gi0/0/0/3	1	0/0	1276	0/10	6384	0

```
! Output omitted for brevity
```

```
Effective Metric:
```

```
Bandwidth: 1000000, Delay: 1, Reliability: 255, Load: 1, MTU: 1500
```

```
R2#show interface GigabitEthernet 0/1
```

```
GigabitEthernet0/1 is up, line protocol is up
```

```
Hardware is AmdP2, address is aabb.cc00.6510 (bia aabb.cc00.6510)
```

```
Internet address is 10.23.1.2/24
```

```
MTU 1500 bytes, BW 1000000 Kbit/sec, DLY 10 usec,
```

```
! Output omitted for brevity
```

EIGRP delay is set on an interface-by-interface basis allowing for manipulation of traffic patterns flowing through a specific interface on a router. Delay is configured with the interface parameter command **delay** *tens-of-microseconds* under the interface on IOS nodes. IOS XR nodes use the command **metric delay** *tens-of-microseconds* within the EIGRP process under the specific interface.

[Example 5-14](#) provides a sample configuration with XR1 and R2 increasing the delay to 1000  $\mu$ s on the link between XR1 and R2 in [Figure 5-10](#).

### Example 5-14 Interface Delay Configuration

[Click here to view code image](#)

```
XR1
```

```
router eigrp 100
address-family ipv4
interface GigabitEthernet0/0/0/3
metric delay 100
```

```
R3
```

```
interface GigabitEthernet0/1
ip address 10.23.1.2 255.255.255.0
delay 100
```

[Example 5-15](#) confirms that the new delay settings have taken effect. Notice that IOS XR adds the field Configured Metric.

### Example 5-15 Verification of EIGRP Delay Settings

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show eigrp interfaces gi0/0/0/3 detail
```

```
IPv4-EIGRP interfaces for AS(100)
```

Interface	Peers	Xmit Queue Un/Reliable	Mean SRTT	Pacing Time Un/Reliable	Multicast Flow Timer	Pending Routes
Gi0/0/0/3	1	0/0	417	0/10	2084	0

```
! Output omitted for brevity
```

```
Effective Metric:
```

```
Bandwidth: 1000000, Delay: 100, Reliability: 255, Load: 1, MTU: 1500
```

```
Configured Metric:
```

```
Delay: 100
```

```
R2#show interfaces GigabitEthernet 0/1 | include line|DLY
```

```
GigabitEthernet0/1 is up, line protocol is up
```

```
MTU 1500 bytes, BW 1000000 Kbit/sec, DLY 1000 usec,
```

#### Note

Bandwidth modification with the interface parameter command **bandwidth** *bandwidth* can have a similar effect on the metric calculation formula but can impact other routing protocols like OSPF at the same time. Modifying interface delay impacts only the EIGRP protocol.

## Load Balancing

EIGRP allows multiple successor routes (same metric) to be installed into the RIB. Installing multiple paths into the RIB for the same prefix is called *equal-cost multipath* (ECMP) routing. At the time of this writing, the default maximum ECMP paths value for IOS nodes is 4 routes and 16 routes for IOS XR-based nodes. The default ECMP setting can be changed with the command **maximum-paths** *maximum-paths* under the EIGRP process to increase the default value to 16 for IOS and 32 for IOS XR nodes.

#### Note

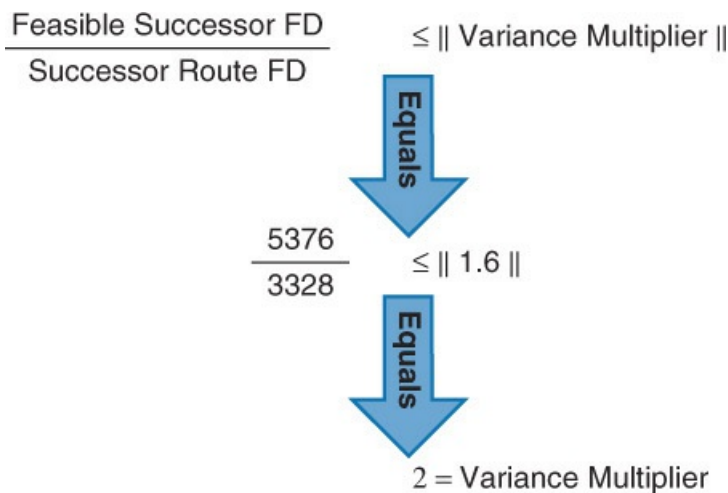
Some IOS platforms support 32 ECMP per prefix.

EIGRP supports unequal-cost load balancing, which allows installation of both successor routes and feasible successors into the EIGRP RIB. EIGRP supports unequal load balancing by changing EIGRP's *variance multiplier*. The *EIGRP variance value* is the feasible distance (FD) for a route multiplied by the EIGRP variance multiplier. Any feasible successor's FD with a metric below the EIGRP variance value is installed into the RIB. EIGRP will install multiple routes where the FD for the routes is less than the EIGRP variance value up to the maximum number of ECMP routes, as discussed earlier.

Dividing the feasible successor metric by the successor route metric provides the variance multiplier. The variance multiplier is a whole number, so any remainders should always round up.

Using [Figure 5-9](#) as the example topology and output from the EIGRP topology table in [Example 5-13](#), the minimum EIGRP variance multiplier can be calculated so that the direct path to R4 can be installed into the RIB. The FD for the successor route is 3328, and the FD for the feasible successor is 5376. The formula provides a value of about 1.6, and is always rounded up to the

nearest whole number to provide an EIGRP variance multiplier of 2. [Figure 5-17](#) shows the calculation.



**Figure 5-17** EIGRP Variance Multiplier Formula

The command **variance multiplier** configures the variance multiplier under the EIGRP process. [Example 5-16](#) provides sample configuration of this configuration.

### **Example 5-16** EIGRP Variance Configuration

[Click here to view code image](#)

```
R1
router eigrp 100
 address-family ipv4
  variance 2
  interface GigabitEthernet0/0/0/2
  !
  interface GigabitEthernet0/0/0/3
  !
  interface GigabitEthernet0/0/0/4
```

```
R2
router eigrp 100
 variance 2
 network 10.0.0.0
```

[Example 5-17](#) provides verification that both paths have been installed into the RIB. Notice that the metric for each path is different. One path metric is 3328 and the other path metric is 5376.

### **Example 5-17** Verification of Unequal Load Balancing

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route 10.4.4.0/24
```

```
Routing entry for 10.4.4.0/24
```

```
Known via "eigrp 100", distance 90, metric 3328, type internal
```

```
Installed Nov 18 15:29:41.318 for 04:30:18
```

```
Routing Descriptor Blocks
```

```
10.13.1.3, from 10.13.1.3, via GigabitEthernet0/0/0/3
```

```
Route metric is 3328
```

```
10.14.1.4, from 10.14.1.4, via GigabitEthernet0/0/0/4
```

```
Route metric is 5376
```

```
No advertising protos.
```

```
R2#show ip route 10.4.4.0
```

```
Routing entry for 10.4.4.0/24
```

```
Known via "eigrp 100", distance 90, metric 3328, type internal
```

```
Redistributing via eigrp 100
```

```
Last update from 10.24.1.4 on GigabitEthernet0/2, 04:30:47 ago
```

```
Routing Descriptor Blocks:
```

```
* 10.24.1.4, from 10.24.1.4, 04:30:47 ago, via GigabitEthernet0/2
```

```
Route metric is 5376, traffic share count is 149
```

```
Total delay is 110 microseconds, minimum bandwidth is 1000000 Kbit
```

```
Reliability 255/255, minimum MTU 1500 bytes
```

```
Loading 255/255, Hops 1
```

```
10.23.1.3, from 10.23.1.3, 04:30:47 ago, via GigabitEthernet0/1
```

```
Route metric is 3328, traffic share count is 240
```

```
Total delay is 30 microseconds, minimum bandwidth is 1000000 Kbit
```

```
Reliability 255/255, minimum MTU 1500 bytes
```

```
Loading 255/255, Hops 2
```

Note

IOS-based platforms will show the unequal load distribution under the *traffic share count*, as indicated in [Example 5-17](#).

## EIGRP Wide Metrics

The original EIGRP specifications measured delay in 10 microsecond ( $\mu$ s) units and bandwidth in kilobytes per second, which did not scale well with higher-speed interfaces. Earlier in [Table 5-6](#), notice that the delay is the same for the Gigabit Ethernet and 10-Gigabit Ethernet interfaces. EIGRP includes support for a second set of metrics known as *wide metrics* that addresses the issue of scalability with higher-capacity interfaces.

Note

The original formula provided earlier in the chapter is known as *EIGRP classic metrics*.

EIGRP wide metric support is supported and the default IOS XR Version 4.3.0 and on various IOS 15 releases. IOS nodes support EIGRP wide metrics only in named configuration mode, which is explained at the end of this chapter.

[Figure 5-18](#) shows the explicit EIGRP wide metric formula. Notice that an additional K value ( $K_6$ ) is included that adds an extended attribute to measure jitter, energy, or other future attributes.

$$\text{Wide Metric} = \left[ (K_1 * \text{BW} + \frac{K_2 * \text{BW}}{256 - \text{Load}} + K_3 * \text{Latency} + K_6 * \text{Extended}) * \frac{K_5}{K_4 + \text{Reliability}} \right]$$

Figure 5-18 EIGRP Wide Metric Formula

Just as EIGRP scaled by 256 to accommodate IGRP, EIGRP wide metrics scale by 65,535 to accommodate higher-speed links. This provides support for interface speeds up to 655 terabits per second ( $65,535 * 10^7$ ) without encountering any scalability issues. Latency is the total interface delay measured in picoseconds ( $10^{-12}$ ) instead of measuring in microseconds ( $10^{-6}$ ).

Figure 5-19 displays the updated formula that takes into account the conversions in latency and scalability.

$$\text{Wide Metric} = 65,535 * \left[ \left( \frac{K_1 * 10^7}{\text{Min. Bandwidth}} + \frac{K_2 * 10^7}{256 - \text{Load}} + \frac{K_3 * \text{Latency}}{10^{-6}} + K_6 * \text{Extended} \right) * \frac{K_5}{K_4 + \text{Reliability}} \right]$$

Figure 5-19 EIGRP Wide Metric Formula with Definitions

The interface delay can vary router to router depending upon the following logic:

- If the interface's delay has been specifically set, the value is converted to picoseconds. Interface delay is always configured in tens of microseconds and is multiplied by  $10^7$  for picosecond conversion.
- If the interface's bandwidth has been specifically set, the interface delay is configured using the classic default delay converted to picoseconds. The configured bandwidth is not considered when determining the interface delay. If delay has been configured, this step is ignored.
- If the interface supports speeds of 1 Gbps or less and does not contain **bandwidth** or **delay** configuration, the delay is the classic default delay converted to picoseconds.
- If the interface supports speeds more than 1 Gbps and does not contain **bandwidth** or **delay** configuration, the interface delay is calculated by  $10^{13}/\text{interface bandwidth}$ .

EIGRP wide metrics were designed with backward compatibility in mind. EIGRP wide metrics set  $K_1$  and  $K_3$  to a value of 1, and  $K_2$ ,  $K_4$ ,  $K_5$ , and  $K_6$  are set to 0, which allows backward compatibility because the K value metrics match with classic metrics. As long as  $K_1$  through  $K_5$  are the same and  $K_6$  is not set, the two metric styles will allow an adjacency between routers.

#### Note

The metric style used by a router can be identified with the command **show ip protocols** on IOS nodes or with the command **show protocols ipv4** on IOS XR nodes. If a  $K_6$  metric is present, the router is using wide-style metrics.

## FAILURE DETECTION AND TIMERS

A secondary function to the EIGRP hello packets is to ensure that EIGRP neighbors are still healthy and available. EIGRP hello packets are sent out in intervals referred to as the *hello timer*. The default EIGRP hello timer is 5 seconds, but it uses 60 seconds on slow-speed interfaces ( $T_1$  or lower).

EIGRP uses a second timer called the *hold time*, which is the amount of time EIGRP deems the

router reachable and functioning. The hold time value defaults to 3 times the hello interval. The default value is 15 seconds and 180 seconds for slow-speed interfaces. The hold time will decrement, and upon receipt of a hello packet, the hold time will reset and restart the countdown. If the hold time reaches 0, EIGRP declares the neighbor unreachable and notifies the DUAL algorithm of a topology change.

The hello timer is modified on IOS nodes with the command **ip hello-interval eigrp as-number seconds** under the interface and can be set with the command **hello-interval seconds** under the interface within the EIGRP process on IOS XR nodes.

The hold timer is modified on IOS nodes with the command **ip hold-time eigrp as-number seconds** under the interface and can be set with the command **hold-time seconds** under the interface within the EIGRP process on IOS XR nodes.

The detailed EIGRP interface command provides the hello and hold time intervals. [Example 5-18](#) shows the detailed EIGRP interface output. Notice that the hello interval is 5 seconds and that the hold time is 15 seconds for the Gigabit Ethernet interfaces.

### Example 5-18 EIGRP Hello and Hold Timer Value Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show eigrp interfaces detail

IPv4-EIGRP interfaces for AS(100)

Interface          Peers  Xmit Queue  Mean   Pacing Time  Multicast  Pending
                  Un/Reliable SRTT    Un/Reliable  Flow Timer  Routes
Gi0/0/0/2          1      0/0         3      0/10         50         0
Hello interval is 5 sec, hold time is 15 sec

R2#show ip eigrp interfaces detail
EIGRP-IPv4 Interfaces for AS(100)

Interface          Peers  Xmit Queue  PeerQ    Mean   Pacing Time  Multicast  Pen
                  Un/Reliable Un/Reliable SRTT    Un/Reliable  Flow Timer  Routes
Gi0/0              1      0/0         0/0      13     0/0         50         0
Hello-interval is 5, Hold-time is 15
! Output omitted for brevity
```

**Note**  
EIGRP neighbors will still form an adjacency if the timers do not match, as long as hellos are received within the hold time. Other routing protocols require the timers to match.

### Convergence

When a link fails and the interface protocol moves to a down state, any neighbor attached to that interface moves to a down state, too. When an EIGRP neighbor moves to a down state, path recomputation must occur for any prefix where that EIGRP neighbor was a successor (upstream router).

When EIGRP detects that it has lost its successor for a path, the feasible successor instantly becomes the successor route providing a backup route. The router sends out an update packet for that path because of the new EIGRP path metrics. Downstream routers will run their own DUAL algorithm for any impacted prefixes to account for the new EIGRP metrics. It is possible that a change of the successor route or feasible successor to occur upon receipt of new EIGRP metrics from a successor router for a prefix.

Figure 5-20 demonstrates such a scenario when the link between R1 and R3 fails.

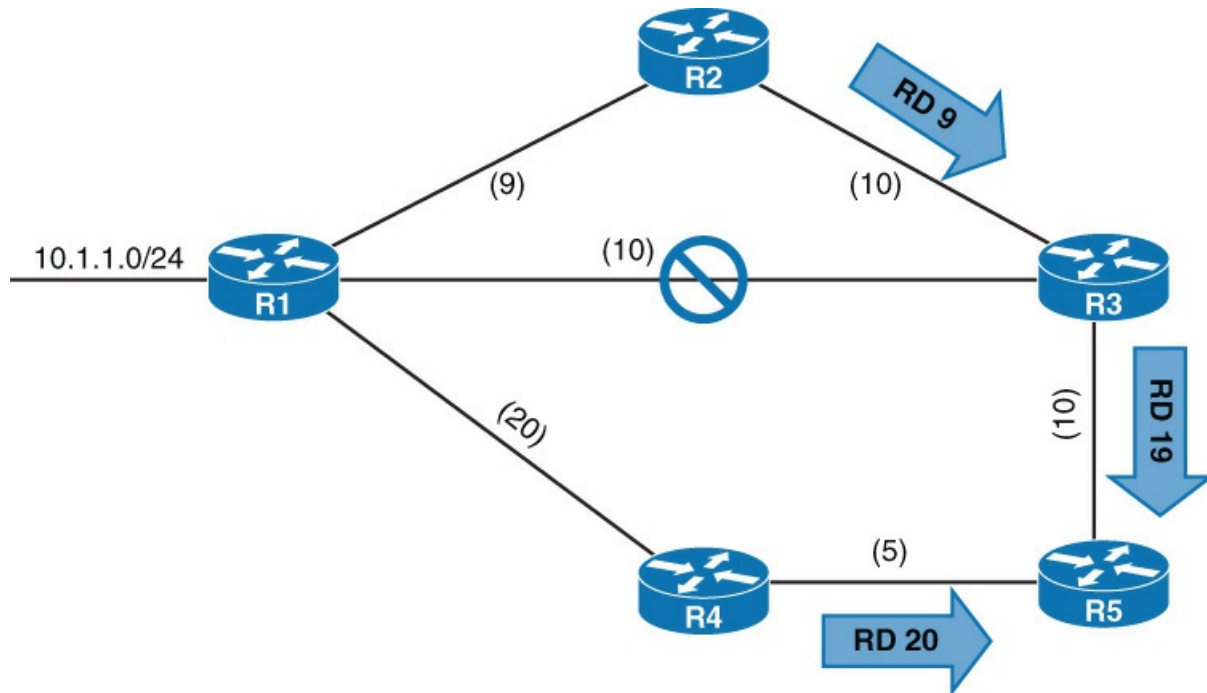


Figure 5-20 EIGRP Topology with Link Failure

R3 installs the feasible successor path advertised from R2 as the successor route. R3 sends an update packet with a new RD of 19 for the 10.1.1.0/24 prefix. R5 receives the update packet from R3 and calculates an FD of 29 for the R1-R2-R3 path to 10.1.1.0/24. R5 compares that path to the one received from R4, which has a path metric of 25. R5 chooses the path via R4 as the successor route.

Example 5-19 provides simulated output of the R5's EIGRP topology for the 10.1.1.0/24 prefix after the R1-R3 link fails.

**Example 5-19** EIGRP Topology for the 10.1.1.0/24 Network

[Click here to view code image](#)



```

R5#show ip eigrp topology 10.1.1.0/24
EIGRP-IPv4 Topology Entry for AS(100)/ID(192.168.5.5) for 10.4.4.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 25
  Descriptor Blocks:
  *10.45.1.4 (GigabitEthernet0/2), from 10.45.1.4, Send flag is 0x0
    Composite metric is (25/20), route is Internal
    Vector metric:
      Hop count is 2
      Originating router is 192.168.1.1
  10.35.1.3 (GigabitEthernet0/1), from 10.35.1.3, Send flag is 0x0
    Composite metric is (29/19), route is Internal
    Vector metric:
      Hop count is 3
      Originating router is 192.168.1.1

```

If a feasible successor is not available for the prefix, DUAL must perform a new route calculation. The route state changes from passive (P) to active (A) in the EIGRP topology table.

The router detecting the topology change sends out query packets to EIGRP neighbors for the route. The query packet includes the network prefix with the delay set to infinity so that other routers are aware that it has gone active. When the router sends the EIGRP query packets, it sets the reply status flag set for each neighbor on a prefix basis.

Upon receipt of a query packet, an EIGRP router will do one of the following:

- Reply to the query that the router does not have a route to the prefix.
- If the query did not come from the successor for that route, it detects the delay set for infinity but ignores it because it did not come from the successor. The receiving router will reply with the EIGRP attributes for that route.
- If the query came from the successor for the route, the receiving router detects the delay set for infinity, sets the prefix as active in the EIGRP topology, and sends out a query packet to all downstream EIGRP neighbors for that route.

The query process continues from router to router until a router establishes the query boundary. A query boundary is established when a router does not mark the prefix as active, meaning that it responds to a query with

- Not having a route to the prefix
- Replying with EIGRP attributes because the query did not come from the successor

When a router receives a reply for all its downstream queries, it completes the DUAL algorithm, changes the route to passive, and sends a reply packet to any upstream routers that sent a query packet to it. Upon receiving the reply packet for a prefix, the reply packet is notated for that neighbor and prefix. The reply process continues upstream for the queries until the first router's queries are received.

Figure 5-21 represents a topology where the link between R1 and R2 has failed.

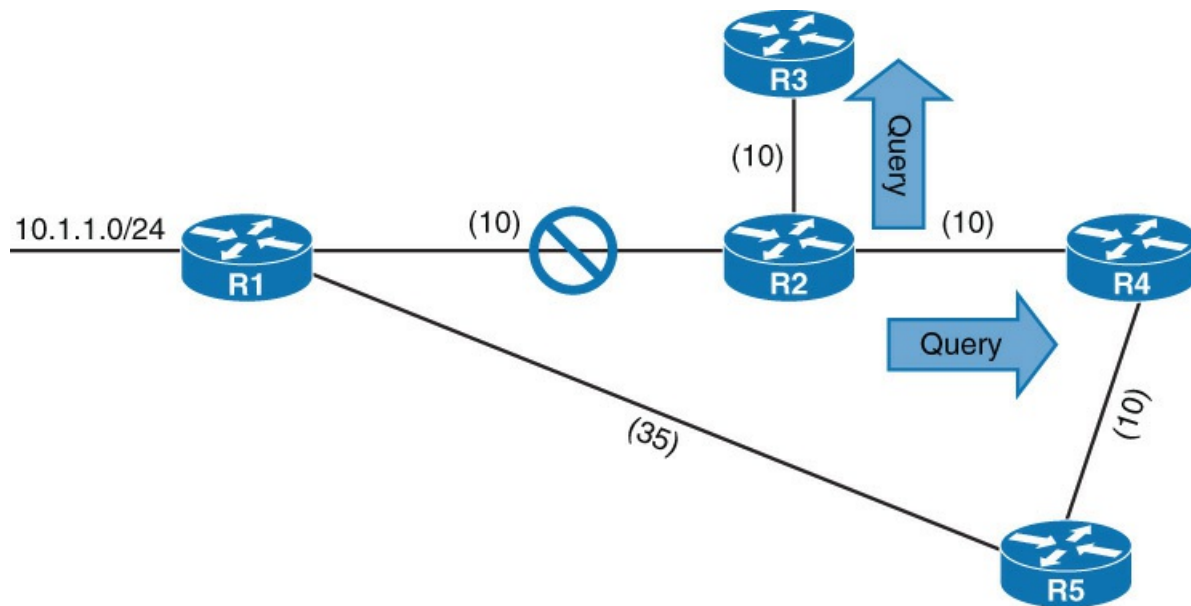


Figure 5-21 EIGRP Convergence Topology

The following steps are processed in order from the perspective of R2 calculating a new route to the 10.1.1.0/24 network:

**Step 1.** R2 detects the link failure. R2 did not have a feasible successor for the route, set the 10.1.1.0/24 prefix as active, and sends queries to R3 and R4.

**Step 2.** R3 receives the query from R2, and processes the Delay field that is set to infinity. R3 does not have any other EIGRP neighbors and sends a reply to R2 that a route does not exist.

R4 receives the query from R2 and processes the Delay field that is set to infinity. Because the query was received by the successor and a feasible successor for the prefix does not exist, R4 marks the route as active and sends a query to R5.

**Step 3.** R5 receives the query from R4 and detects the Delay field is set to infinity. Because the query was received by a nonsuccessor and a successor exists on a different interface, a reply for the 10.4.4.0/24 network is sent back to R4 with the appropriate EIGRP attributes.

**Step 4.** R4 receives R5's reply, acknowledges the packet, and computes a new path. Because this is the last outstanding query packet on R4, R4 sets the prefix as passive. With all queries satisfied, R4 responds to R2's query with the new EIGRP metrics.

**Step 5.** R2 receives R4's reply, acknowledges the packet, and computes a new path. Because this is the last outstanding query packet on R2, R2 sets the prefix as passive.

#### Stuck in Active

DUAL is very efficient at finding loop-free paths quickly and normally finds backup path in seconds. Occasionally, an EIGRP query is delayed because of packet loss, slow neighbors, or a large hop count. EIGRP will wait one-half of the active timer (90 seconds by default) for a reply. If the router does not receive a response within 90 seconds, the originating router will send a stuck-in-active (SIA) query to EIGRP neighbors that have not responded.

Upon receipt of a SIA query, the router should respond within 90 seconds with an SIA reply. An SIA reply contains the route information or provides information on the query process itself. If a router fails to respond to a SIA query by the time the active timer expires, EIGRP deems the router as SIA. If the SIA state is declared for a neighbor, DUAL will delete all routes from that neighbor treating the situation as if the neighbor responded with unreachable message for all routes.

Note

Previous behavior in earlier versions of IOS would terminate EIGRP neighbor sessions with routers that never replied to the SIA query.

Troubleshooting active EIGRP prefixes must occur when a router is waiting for a reply. Active queries are shown with the command **show ip eigrp topology active** on IOS nodes, and with the command **show eigrp topology active** on IOS XR nodes.

Figure 5-22 shows a topology where the link between R1 and R2 has failed. R2 sends out queries to R4 and R3. R4 sends a reply back to R2, and R3 sends a query on to R5.

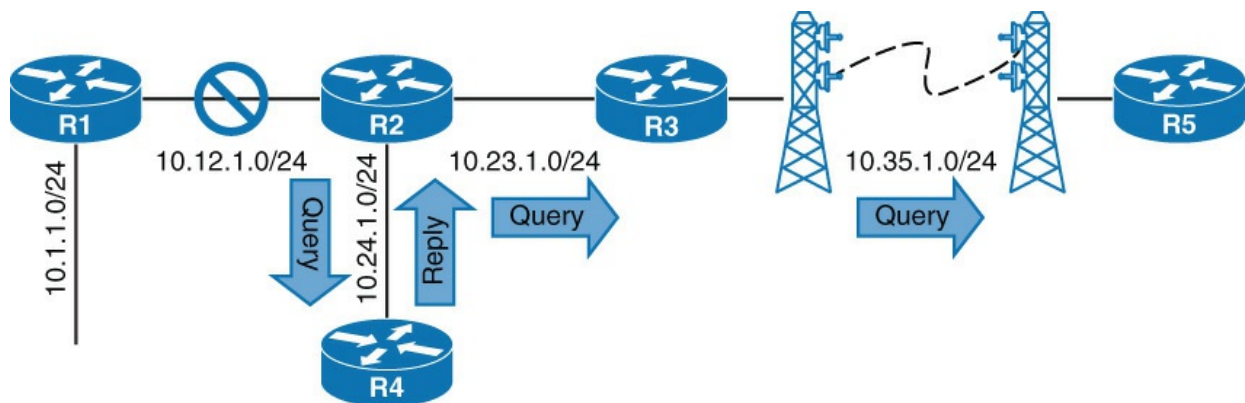


Figure 5-22 EIGRP SIA Topology

A network engineer sees the syslog message and immediately runs the **show ip eigrp topology active** command on R2 and sees output the output from Example 5-20. The *r* next to the peer's IP address (10.23.1.3) indicates that R2 is still waiting on the reply from R3 and that R4 has responded. The command can then be executed on R3, which will indicate it is waiting on a response from R5. Executing the command on R5 does not show any active prefixes, inferring that R5 never received a query from R3. R3's query could have been dropped on the radio tower connection.

### Example 5-20 Output for SIA Timers

[Click here to view code image](#)

```

R2#show ip eigrp topology active
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
      r - Reply status

A 10.1.1.0/24, 0 successors, FD is 512640000, Q
  1 replies, active 00:00:01, query-origin: Local origin
    via 10.24.1.4 (Infinity/Infinity), GigabitEthernet 0/0
  1 replies, active 00:00:01, query-origin: Local origin
    via 10.23.1.3 (Infinity/Infinity), r, GigabitEthernet 0/1
Remaining replies:
  via 10.23.1.3, r, GigabitEthernet 0/1

```

The active timer is set to 3 minutes by default. The active timer can be disabled or modified with the following command **timers active-time** {*disabled* | *1-65535-minutes*} under the EIGRP process.

The active timer is shown by examining the IP protocols on a router; the configured duration is after active timer. [Example 5-21](#) shows the default active timer value of 3 minutes.

### Example 5-21 Output for SIA Timers

[Click here to view code image](#)

```

RP/0/0/CPU0:XR1#show protocols ipv4 | include Active

SIA Active timer is 180s

R2#show ip protocols | include Active

Active Timer: 3 min

```

## STUB

EIGRP stub functionality allows an EIGRP router to conserve router resources. An EIGRP stub router announces itself as a stub within the EIGRP hello packet. Neighboring routers detect the Stub field and update the EIGRP neighbor table to reflect the router's stub status. If a route goes active, EIGRP does not send EIGRP queries to an EIGRP stub router. This provides faster convergence within an EIGRP autonomous system because it decreases the size of the query domain for that prefix.

EIGRP stubs do not advertise routes that they learn from other EIGRP peers. By default, EIGRP stubs advertise only connected and summary routes but can be configured so that they receive only routes or advertise any combination of redistributed routes, connected routes, or summary routes.

[Figure 5-23](#) shows a typical example where a remote location (R5 and XR6) has two network links between different regional offices for redundancy. Inter-region traffic should always use the links within the 172.16.0.0/16 network range; however, if a link failure occurs on the 172.16.34.0/24 link, traffic could be routed through the remote sites using the 10.0.0.0/8 network range, which is not an optimal routing path.

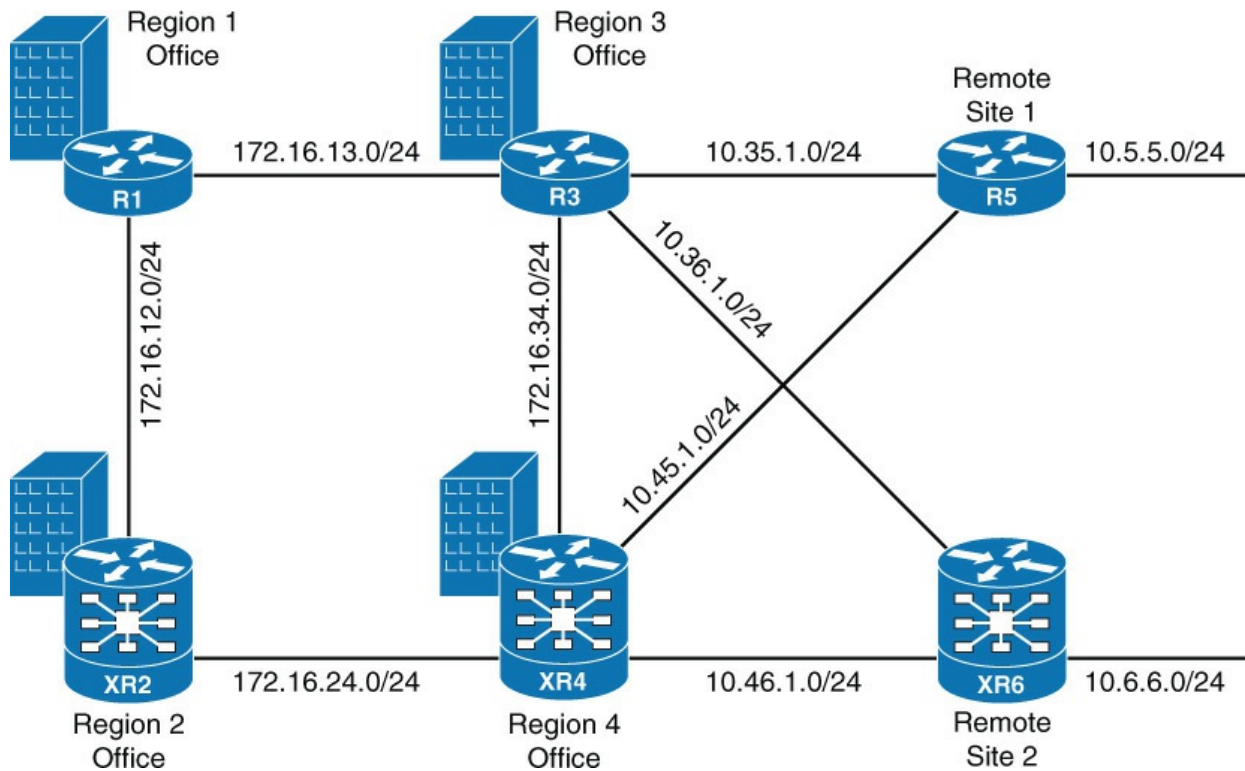


Figure 5-23 EIGRP Regional/Remote Topology

Example 5-22 provides the routing table for the regional routers R3 and XR4 after the 172.16.34.0/24 link fails. Notice that most inter-region network connectivity flows across the remote site links in the 10.0.0.0/8 range instead of using the inter-regions links in the 172.16.0.0/16 network range.

### Example 5-22 EIGRP Routes for Inter-Region Traffic

[Click here to view code image](#)

```
R3-RegionOffice3#show ip route eigrp
! Output omitted for brevity
Gateway of last resort is not set

10.0.0.0/8 is variably subnetted, 9 subnets, 2 masks
D    10.5.5.0/24 [90/3072] via 10.35.1.5, 00:02:03, GigabitEthernet0/2
D    10.6.6.0/24 [90/3072] via 10.36.1.6, 00:02:03, GigabitEthernet0/3
D    10.45.1.0/24 [90/3072] via 10.35.1.5, 00:02:04, GigabitEthernet0/2
D    10.46.1.0/24 [90/3072] via 10.36.1.6, 00:02:04, GigabitEthernet0/3
D    10.56.1.0/24 [90/3072] via 10.35.1.5, 00:02:03, GigabitEthernet0/2
172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
D    172.16.12.0/24 [90/3072] via 172.16.13.1, 00:02:03, GigabitEthernet0/1
D    172.16.24.0/24 [90/3328] via 172.16.13.1, 00:02:04, GigabitEthernet0/1
      [90/3328] via 10.36.1.6, 00:02:04, GigabitEthernet0/3
      [90/3328] via 10.35.1.5, 00:02:04, GigabitEthernet0/2
192.168.4.0/32 is subnetted, 1 subnets
D    192.168.4.4 [90/131072] via 10.36.1.6, 00:02:04, GigabitEthernet0/3
      [90/131072] via 10.35.1.5, 00:02:04, GigabitEthernet0/2
```

```
RP/0/0/CPU0:XR4-RegionOffice4#show route eigrp
```

```
D 10.5.5.0/24 [90/3072] via 10.45.1.5, 00:00:13, GigabitEthernet0/0/0/4
D 10.6.6.0/24 [90/3072] via 10.46.1.6, 00:13:44, GigabitEthernet0/0/0/0
D 10.35.1.0/24 [90/3072] via 10.45.1.5, 00:00:13, GigabitEthernet0/0/0/4
D 10.36.1.0/24 [90/3072] via 10.46.1.6, 00:03:15, GigabitEthernet0/0/0/0
D 10.56.1.0/24 [90/3072] via 10.45.1.5, 00:00:13, GigabitEthernet0/0/0/4
D 172.16.12.0/24 [90/3072] via 172.16.24.2, 00:13:44, GigabitEthernet0/0/0/2
D 172.16.13.0/24 [90/3328] via 10.45.1.5, 00:00:10, GigabitEthernet0/0/0/4
    [90/3328] via 172.16.24.2, 00:00:10, GigabitEthernet0/0/0/2
    [90/3328] via 10.46.1.6, 00:00:10, GigabitEthernet0/0/0/0
D 192.168.3.3/32 [90/131072] via 10.45.1.5, 00:00:10, GigabitEthernet0/0/0/4
    [90/131072] via 10.46.1.6, 00:00:10, GigabitEthernet0/0/0/0
```

The solution is to make R5 and XR6 EIGRP stub routers. By default, EIGRP stubs will advertise only locally connected networks and summary routes.

The command **eigrp stub {connected | receive-only | redistributed | static | summary}** is used on IOS nodes under the EIGRP process. IOS XR configuration is under the EIGRP process and appropriate address family with the command **stub {connected | receive-only | redistributed | static | summary}**.

#### Note

The **receive-only** option cannot be combined with other EIGRP stub options. The network design should be given special consideration to ensure bidirectional connectivity for any networks connected to an EIGRP router with the receive-only stub option to ensure that routers know how to send return traffic.

[Example 5-23](#) provides the configuration of R5 and XR6 with the default EIGRP stub settings.

### Example 5-23 EIGRP Stub Configuration

[Click here to view code image](#)

#### R5

```
router eigrp 100
 network 0.0.0.0 255.255.255.255
 eigrp stub
```

#### XR6

```
router eigrp 100
 address-family ipv4
 stub
 interface GigabitEthernet0/0/0/0
 !
 interface GigabitEthernet0/0/0/1
 !
 interface GigabitEthernet0/0/0/2
```

Example 5-24 shows the routing table under ideal circumstances. R5 and XR6 are able to advertise their networks while receiving updates from their upstream neighbors R3 and XR4. Notice that the traffic is now using the inter-regional network links in the 172.16.0.0/16 network range for communicating between the sites during a 172.16.34.0/24 link failure.

### **Example 5-24** *Route Tables After Link Failure with Stub*

[Click here to view code image](#)

```
R3-RegionOffice3#show ip route eigrp
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 9 subnets, 2 masks
```

```
D 10.5.5.0/24 [90/3072] via 10.35.1.5, 00:00:17, GigabitEthernet0/2
D 10.6.6.0/24 [90/3072] via 10.36.1.6, 00:00:27, GigabitEthernet0/3
D 10.45.1.0/24 [90/3072] via 10.35.1.5, 00:00:17, GigabitEthernet0/2
D 10.46.1.0/24 [90/3072] via 10.36.1.6, 00:00:27, GigabitEthernet0/3
D 10.56.1.0/24 [90/3072] via 10.35.1.5, 00:00:17, GigabitEthernet0/2
```

```
172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
```

```
D 172.16.12.0/24 [90/3072] via 172.16.13.1, 00:06:58, GigabitEthernet0/1
D 172.16.24.0/24 [90/3328] via 172.16.13.1, 00:00:20, GigabitEthernet0/1
192.168.4.0/32 is subnetted, 1 subnets
D 192.168.4.4 [90/131328] via 172.16.13.1, 00:00:20, GigabitEthernet0/1
```

```
RP/0/0/CPU0:XR4-RegionOffice4#show route eigrp
```

```
D 10.5.5.0/24 [90/3072] via 10.45.1.5, 00:01:27, GigabitEthernet0/0/0/4
D 10.6.6.0/24 [90/3072] via 10.46.1.6, 00:01:36, GigabitEthernet0/0/0/0
D 10.35.1.0/24 [90/3072] via 10.45.1.5, 00:01:27, GigabitEthernet0/0/0/4
D 10.36.1.0/24 [90/3072] via 10.46.1.6, 00:01:36, GigabitEthernet0/0/0/0
D 10.56.1.0/24 [90/3072] via 10.45.1.5, 00:01:27, GigabitEthernet0/0/0/4
D 172.16.12.0/24 [90/3072] via 172.16.24.2, 00:21:42, GigabitEthernet0/0/0/2
D 172.16.13.0/24 [90/3328] via 172.16.24.2, 00:01:30, GigabitEthernet0/0/0/2
D 192.168.3.3/32 [90/131328] via 172.16.24.2, 00:01:30, GigabitEthernet0/0/0/2
```

### **Design Considerations with EIGRP Stubs**

A common problem with EIGRP stubs is forgetting that they do not advertise EIGRP routes that they learn from another peer.

In Figure 5-24, R5 and XR6 are remote routers with a single link to the closest regional office. A circuit between R5 and XR6 provides backup connectivity in the event of a link failure with the regional office from either router. After the network was set up, a junior network engineer thought that making R5 and XR6 EIGRP stubs would be a good idea to conserve router resources at each of the remote locations.

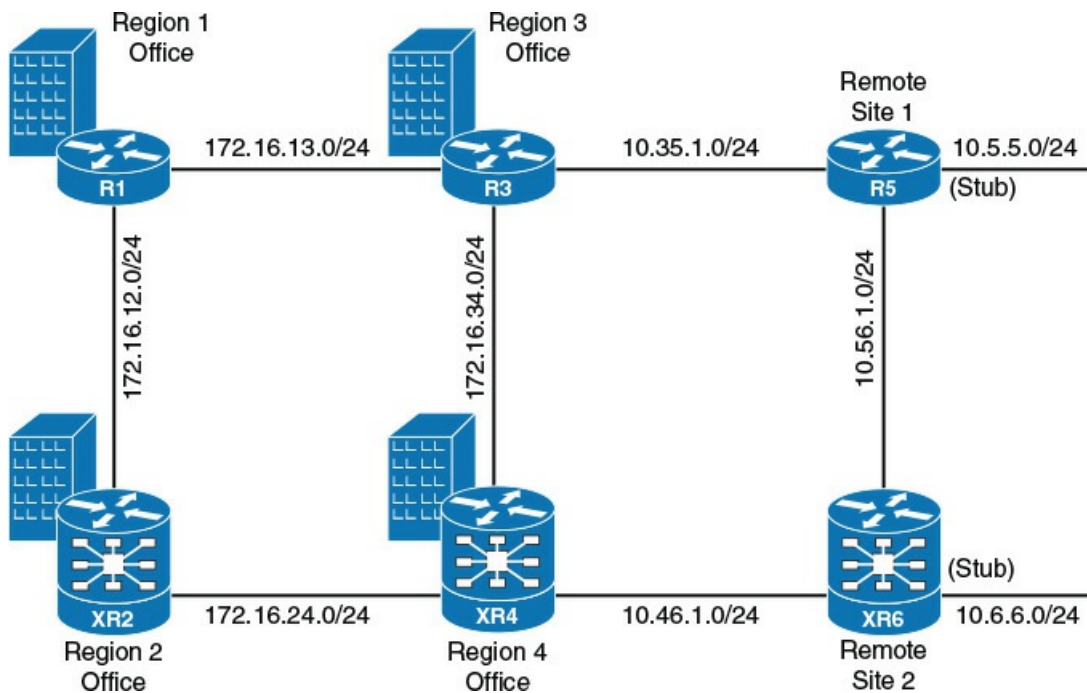


Figure 5-24 Regional Topology with Remote Sites

Figure 5-25 shows a link failure between R3 and R5 on the 10.35.1.0/24 network. Users at the remote site 1 started to complain because they could not access any resources in any regional office, but they could still communicate with devices within remote site 2.

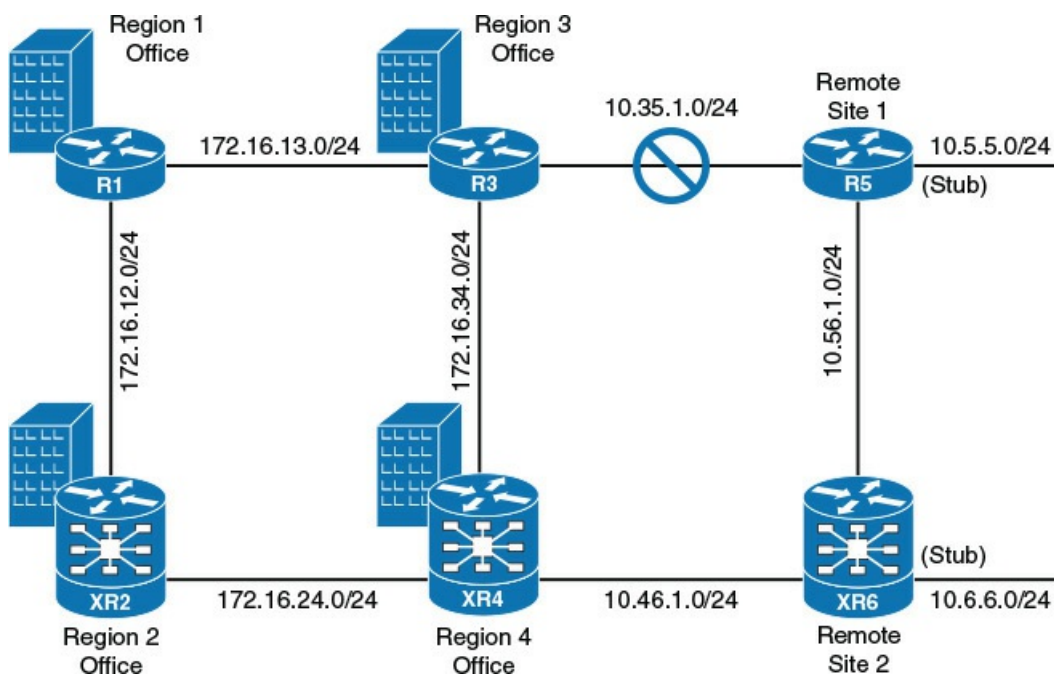


Figure 5-25 Failure on R3 to R5 Link

Example 5-25 shows the routing table of R5 and XR6 after the link failure. Notice that XR6 has full connectivity to all the networks, but R5 only has XR6's directly connected subnets. This is because XR6 is a stub and does not advertise any other routes to R5.

### Example 5-25 Remote Sites Routing Table After Link Failure

[Click here to view code image](#)



```
R5-RemoteSite1#show ip route eigrp
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
```

```
D 10.6.6.0/24 [90/3072] via 10.56.1.6, 00:00:33, GigabitEthernet0/3
```

```
D 10.46.1.0/24 [90/3072] via 10.56.1.6, 00:00:29, GigabitEthernet0/3
```

```
RP/0/0/CPU0:XR6-RemoteSite2#show route eigrp
```

```
D 10.5.5.0/24 [90/3072] via 10.56.1.5, 00:02:55, GigabitEthernet0/0/0/3
```

```
D 10.35.1.0/24 [90/3328] via 10.46.1.4, 00:01:10, GigabitEthernet0/0/0/0
```

```
D 10.36.1.0/24 [90/3328] via 10.46.1.4, 00:02:09, GigabitEthernet0/0/0/0
```

```
D 172.16.12.0/24 [90/3328] via 10.46.1.4, 00:02:59, GigabitEthernet0/0/0/0
```

```
D 172.16.13.0/24 [90/3328] via 10.46.1.4, 00:02:09, GigabitEthernet0/0/0/0
```

```
D 172.16.24.0/24 [90/3072] via 10.46.1.4, 00:08:23, GigabitEthernet0/0/0/0
```

```
D 172.16.34.0/24 [90/3072] via 10.46.1.4, 00:02:25, GigabitEthernet0/0/0/0
```

```
D 192.168.3.3/32 [90/131072] via 10.46.1.4, 00:02:09, GigabitEthernet0/0/0/0
```

```
D 192.168.4.4/32 [90/130816] via 10.46.1.4, 00:08:23, GigabitEthernet0/0/0/0
```

Because EIGRP stubs do not advertise routes learned from other routers, the functionality does not meet the design goals of having a backup link. Summarization of route prefixes provides a query boundary and conserves routing resources.

## SUMMARIZATION

The EIGRP protocol works well with minimal optimizations. Scalability of the EIGRP autonomous system depends on summarization. As the size of the EIGRP autonomous system increases, convergence may take longer. Scaling an EIGRP topology requires summarizing routes in a hierarchical fashion. Figure 5-26 shows summarization occurring at the access, distribution and core layers of the network topology. In addition to shrinking the routing table of all the routers, summarization creates a query boundary and shrinks the query domain when a route goes active during convergence reducing SIA scenarios.

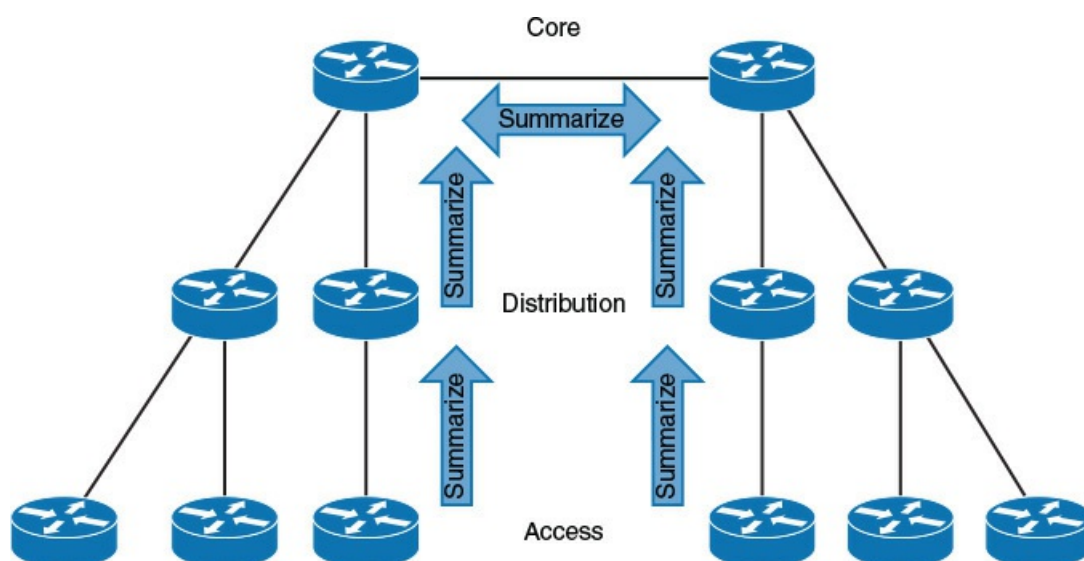


Figure 5-26 EIGRP Hierarchical Summarization

Note

Route summarization on this scale requires hierarchical deployment of an IP addressing scheme.

### Interface-Specific Summarization

EIGRP summarizes network prefixes on an interface basis. A summary aggregate is configured for the EIGRP interface. Prefixes within the summary aggregate are suppressed, and the summary aggregate prefix is advertised in lieu of the original prefixes. The summary aggregate prefix is not advertised until a prefix matches it. Interface-specific summarization can be performed at any portion of the network topology.

Figure 5-27 illustrates the concept of EIGRP summarization. Without summarization, R2 advertises the 172.16.1.0/24, 172.16.3.0/24, 172.16.12.0/24, and 172.16.23.0/24 network toward R4. R2 can summarize these network prefixes to the summary aggregate 172.16.0.0/16 prefix so that only one advertisement is sent to R4.

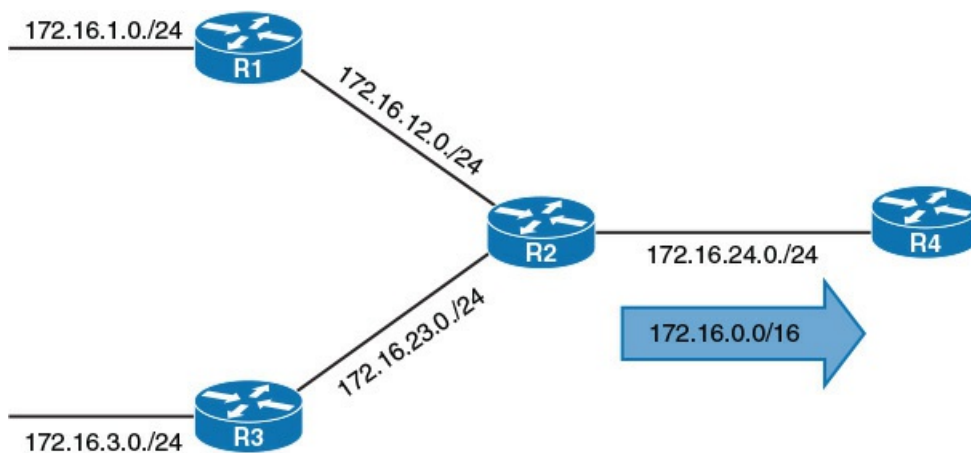


Figure 5-27 Concept of EIGRP Summarization

IOS nodes use the interface parameter command **ip summary-address eigrp as-number network subnet-mask [leak-map route-map-name]** to place an EIGRP summary aggregate on an interface. The **leak-map** option allows the advertisement of the routes identified in the route map. Because suppression is avoided, the routes are considered leaked because they are advertised along with the summary aggregate. This allows for the use of longest match routing to influence traffic patterns while suppressing a majority of the prefixes. IOS XR allows the option of setting the AD from 1 to 255 when configuring the summary aggregate. The command for IOS XR nodes is **summary-address network/prefix-length [AD-value]** underneath the interface within the EIGRP process.

Note

Configuring an EIGRP summary aggregate on an interface for IOS and IOS XR nodes will reset the EIGRP neighbors connected via that interface.

Figure 5-28 demonstrates XR1 and R4 summarizing multiple /24 networks into a single /15 network.



Figure 5-28 Sample EIGRP Summarization Topology

Example 5-26 provides the IP routing table for XR2 and R3 before summarization. Notice that only /24 networks exist in the route table on XR2 and R3.

### Example 5-26 XR2 and R3 Route Table Before EIGRP Summarization

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route eigrp
```

```
D 172.16.16.0/24 [90/3072] via 10.12.1.1, 00:00:11, GigabitEthernet0/0/0/0
D 172.17.17.0/24 [90/3072] via 10.12.1.1, 00:00:11, GigabitEthernet0/0/0/0
D 172.18.18.0/24 [90/3072] via 10.12.1.1, 00:00:11, GigabitEthernet0/0/0/0
```

```
R3#show ip route eigrp
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
172.20.0.0/24 is subnetted, 1 subnets
D 172.20.20.0 [90/3072] via 10.34.1.4, 00:00:36, GigabitEthernet0/0
172.21.0.0/24 is subnetted, 1 subnets
D 172.21.21.0 [90/3072] via 10.34.1.4, 00:00:27, GigabitEthernet0/0
172.22.0.0/24 is subnetted, 1 subnets
D 172.22.22.0 [90/3072] via 10.34.1.4, 00:00:16, GigabitEthernet0/0
```

Example 5-27 provides the EIGRP configuration with the 172.16.0.0/15 summary aggregate configured on XR1's Gi0/0/0/0 interface and the 172.20.0.0/15 summary aggregate on R4's Gi0/0 interface.

### Example 5-27 EIGRP Summary Aggregate Configuration

[Click here to view code image](#)

**XR1**

```
router eigrp 100
  address-family ipv4
    interface GigabitEthernet0/0/0/0
      summary-address 172.16.0.0/15
    !
    interface GigabitEthernet0/0/0/1
    !
    interface GigabitEthernet0/0/0/2
    !
    interface GigabitEthernet0/0/0/3
```

**R4**

```
interface GigabitEthernet0/0
  ip address 10.34.1.4 255.255.255.0
  ip summary-address eigrp 100 172.20.0.0 255.254.0.0
```

**Example 5-28** shows the route table on XR2 and R3 after route summarization. Notice that only the networks within the 172.16.0.0/15 or 172.20.0.0/15 ranges were suppressed.

**Example 5-28 XR2 and R3 Route Table After Summarization**

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route eigrp
```

```
D 172.16.0.0/15 [90/3072] via 10.12.1.1, 00:00:02, GigabitEthernet0/0/0/0
D 172.18.18.0/24 [90/3072] via 10.12.1.1, 00:00:02, GigabitEthernet0/0/0/0
```

```
R3#show ip route eigrp
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
D 172.20.0.0/15 [90/3072] via 10.34.1.4, 00:00:04, GigabitEthernet0/0
  172.22.0.0/24 is subnetted, 1 subnets
D 172.22.22.0 [90/3072] via 10.34.1.4, 00:01:20, GigabitEthernet0/0
```

EIGRP installs a prefix with the summary aggregate prefix with a destination of Null0 on the summarizing routers as a routing loop-prevention mechanism. This prevents routing loops where portions of the summarized network range do not have a more specific entry in the RIB on the summarizing router.

The AD for the Null0 route is 5 by default and can be set on IOS nodes under the EIGRP process with the command **summary-metric network {/prefix-length | subnet-mask} distance AD**. The AD can be set on IOS XR nodes by appending the AD at the end of the summary-address command, as shown earlier. Setting the AD to a higher value allows the router to install a summarized prefix learned via another routing protocol.

[Example 5-29](#) shows the EIGRP route to Null0 on XR1 and R4.

### Example 5-29 XR1 and R4 Route Table After Summarization

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
! Output omitted for brevity
Gateway of last resort is not set

C   10.12.1.0/24 is directly connected, 00:22:47, GigabitEthernet0/0/0/0
D   172.16.0.0/15 [5/2816] via 0.0.0.0, 00:19:02, Null0
C   172.16.16.0/24 is directly connected, 00:22:47, GigabitEthernet0/0/0/1
C   172.17.17.0/24 is directly connected, 00:22:47, GigabitEthernet0/0/0/2
C   172.18.18.0/24 is directly connected, 00:22:47, GigabitEthernet0/0/0/3
```

```
R4#show ip route
! Output omitted for brevity

Gateway of last resort is not set

    10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.34.1.0/24 is directly connected, GigabitEthernet0/0
D       172.20.0.0/15 is a summary, 00:11:58, Null0
        172.20.0.0/16 is variably subnetted, 2 subnets, 2 masks
C       172.20.20.0/24 is directly connected, GigabitEthernet0/1
        172.21.0.0/16 is variably subnetted, 2 subnets, 2 masks
C       172.21.21.0/24 is directly connected, GigabitEthernet0/2
        172.22.0.0/16 is variably subnetted, 2 subnets, 2 masks
C       172.22.22.0/24 is directly connected, GigabitEthernet0/3
```

As stated before, EIGRP summarizes network ranges on an interface level. All the interfaces should be taken into consideration when deploying summarization to ensure that a more specific prefix is not leaked downstream to the other EIGRP routers. In [Figure 5-29](#), R1 is summarizing only on the R1-R3 link, but the original prefixes are still advertised toward R2, which will then advertise the specific /24 prefixes to R4.

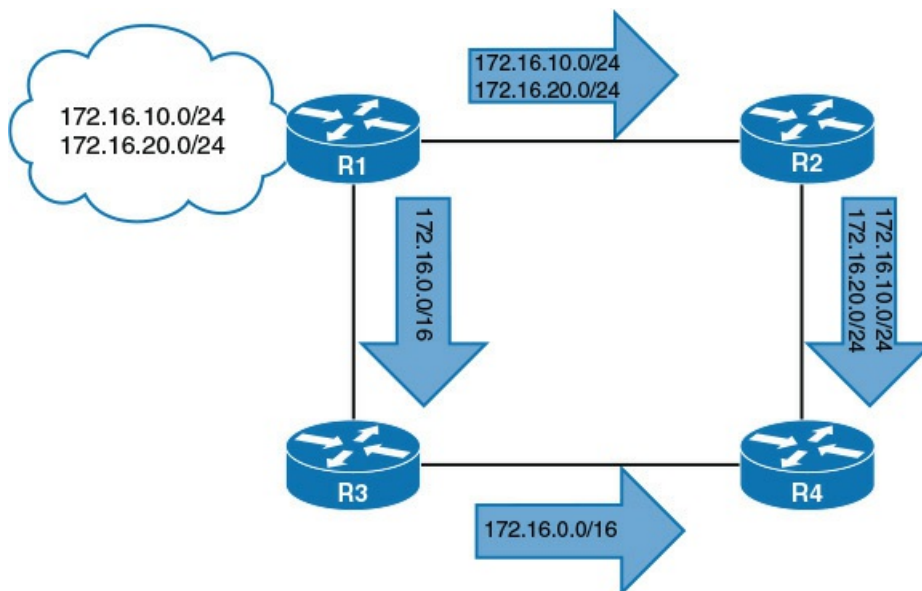


Figure 5-29 Problems with EIGRP Summarization

Example 5-30 shows the routing table for R4. Notice that R4 contains all three prefixes in the routing table. R2, R3, and R4 will send traffic to the 172.16.10.0/24 and 172.16.20.0/24 networks via the R1-R2 link because they are a longer match in the RIB. This could cause an unintended routing behavior and is resolved by placing summarization statements on both interfaces.

### Example 5-30 R4 EIGRP Routing Table

[Click here to view code image](#)

```
R4#show ip route eigrp
! Output omitted for brevity

Gateway of last resort is not set

  10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
D       10.12.1.0/24 [90/3072] via 10.24.1.2, 00:06:52, GigabitEthernet0/1
D       10.13.1.0/24 [90/3072] via 10.34.1.3, 00:06:50, GigabitEthernet0/0
D       172.16.0.0/16 [90/131072] via 10.34.1.3, 00:03:41, GigabitEthernet0/0
D       172.16.10.0/24 [90/131072] via 10.24.1.2, 00:03:41, GigabitEthernet0/1
D       172.16.20.0/24 [90/131072] via 10.24.1.2, 00:03:41, GigabitEthernet0/1
```

### Summarization Metrics

The summarizing router will use the lowest metric of the routes in the summary aggregate prefix. The path metric for the summary aggregate is based on the path attributes of the lowest metric path. EIGRP path attributes like total delay and minimum bandwidth are inserted into the summary route so that downstream routers can calculate the correct path metric for the summarized prefix.

In Figure 5-30, R2 has a path metric of 3,072 for 172.16.1.0/24 prefix and a path metric of 3328 for the 172.16.3.0/24 prefix. The 172.16.0.0/16 summary aggregate is advertised with the path metric of 3072 and the EIGRP path attributes received by R2 from R1.

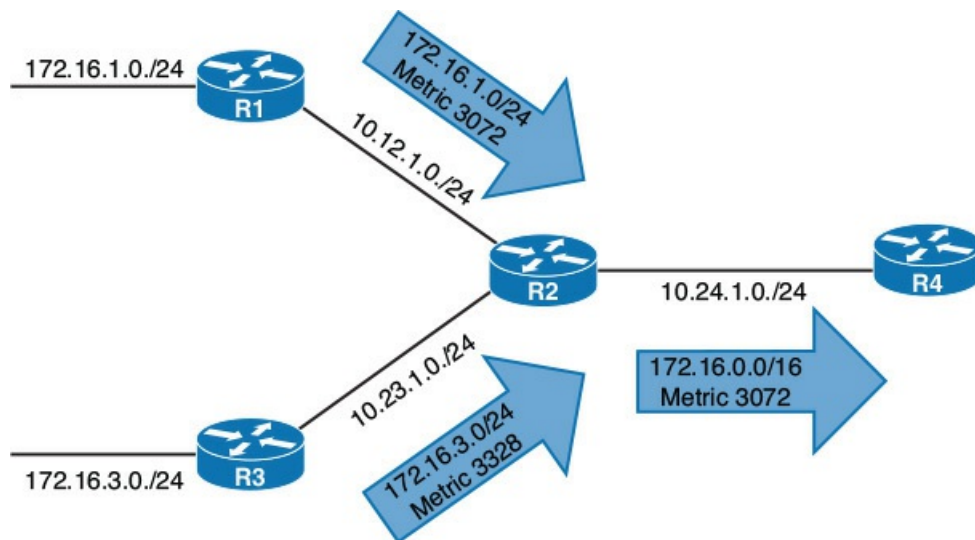


Figure 5-30 EIGRP Summarization Metrics

Every time a matching prefix for the summary aggregate is added or removed, EIGRP must verify that the advertised path is still the lowest path metric. If it is not, a new summary aggregate is advertised with updated EIGRP attributes, and downstream routes must run the DUAL algorithm again. The summary aggregate hides the smaller prefixes from downstream routers, but downstream routers are still burdened with processing updates to the summary aggregate.

There are two methods to prevent fluctuation in the path metric for a summary aggregate:

- Advertise a loopback interface within the summarization aggregate on the summarizing router. The interface will never flap and will always have the lowest metric.
- IOS nodes allow the metric to be statically set on a summary aggregate with the command **summary-metric network** {/prefix-length | subnet-mask} bandwidth delay reliability load MTU. Bandwidth is in kilobits per second (Kbps), delay is in 10 microsecond ( $\mu$ s) units, reliability and load are values between 1 and 255, and the maximum transmission unit (MTU) is the MTU for the interface. IOS XR does not support this feature at this time.

### Advertising a Default Route

Advertising a default route into EIGRP uses the summarization syntax described in the preceding list, except that the network and mask will use 0.0.0.0 0.0.0.0 (commonly referred to as *quad zeros*). [Example 5-31](#) shows configuration for advertising a default route from an IOS interface.

### Example 5-31 Sample EIGRP Default Route Advertisement

[Click here to view code image](#)

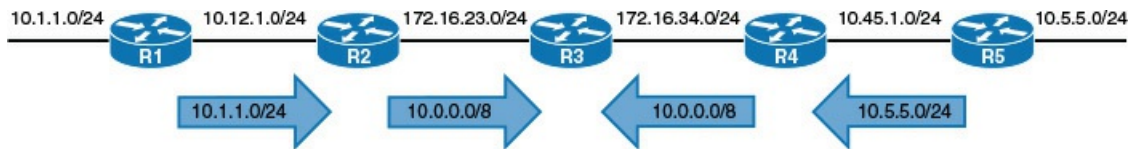
```
interface GigabitEthernet0/0
 ip address 10.34.1.4 255.255.255.0
 ip summary-address eigrp 100 0.0.0.0 0.0.0.0
```

Note

IOS XR does not support advertising a default route within the EIGRP protocol at this time; future releases may add support. The easiest solution is to redistribute the default route from another routing protocol into the EIGRP autonomous system. The downside to this technique is that the AD will be set to 170.

### Automatic Summarization

EIGRP supports automatic summarization, which automatically summarizes networks advertisements when they cross a classful network boundary. [Figure 5-31](#) demonstrates automatic summarization for the 10.1.1.0/24 route on R2 and the 10.5.5.0/24 network on R4. R2 and R4 advertise only the classful network 10.0.0.0/8 toward R3.



**Figure 5-31** Problems with EIGRP Automatic Summarization

[Example 5-32](#) provides the route table for R3. Notice that there are no routes for the 10.1.1.0/24 or 10.5.5.0/24 network, only the route for the 10.0.0.0/8 with a next hop of R2 and R4. Traffic sent to either network could be sent to out the wrong interface. This problem impacts network traffic traveling across the network in addition to traffic origination from R3.

### Example 5-32 Output for Verification of EIGRP Interfaces

[Click here to view code image](#)

```
R3#show ip route eigrp
! Output omitted for brevity

Gateway of last resort is not set

D    10.0.0.0/8 [90/3072] via 172.16.34.4, 00:08:07, GigabitEthernet0/0
      [90/3072] via 172.16.23.2, 00:08:07, GigabitEthernet0/1
```

[Example 5-33](#) shows a similar behavior for the 172.16.23.0/24 and 172.16.34.0/24 network as they are advertised as 172.16.0.0/16 networks from R2 to R1. The identical advertisement occurs from R4 to R5 too.

### Example 5-33 Output for Verification of EIGRP Interfaces

[Click here to view code image](#)



```
R1#show ip route eigrp
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
D    172.16.0.0/16 [90/3072] via 10.12.1.2, 00:09:50, GigabitEthernet0/0
```

```
R5#show ip route eigrp
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
D    172.16.0.0/16 [90/3072] via 10.12.1.2, 00:09:50, GigabitEthernet0/1
```

Current releases of IOS and IOS XR disable EIGRP classful network automatic summarization by default. Automatic summarization is enabled with the command **auto-summary** under the EIGRP process for IOS and under the address family for IOS XR nodes. To disable automatic summarization for EIGRP, use the command **no auto-summary** under the EIGRP process for IOS and under the address family for IOS XR nodes.

#### Note

Before IOS Version 12.2(8)T, EIGRP automatic summarization was enabled by default. To ensure consistency of this setting, explicitly enabling/disabling automatic summarization on all routers is recommended.

## AUTHENTICATION

Authentication is a mechanism for ensuring that only authorized routers are eligible to become EIGRP neighbors. It is possible for someone to add a router to a network and introduce invalid routes accidentally or maliciously. Authentication prevents these scenarios from happening. A precomputed password hash is included with all EIGRP packets, and the receiving router decrypts the hash. If the passwords do not match, the router discards the packet.

EIGRP encrypts the password using a message digest 5 (MD5) authentication using the key chain function. The hash consists of the key number and a password. EIGRP authentication does not encrypt the entire EIGRP packet, just the password.

#### Note

Key chain functionality allows a password to be valid for a specific time, so passwords can change at preconfigured times. Restricting key sequence to specific time feature is beyond the scope of this book, but is located in other documentation at Cisco.com

Configuration of EIGRP authentication requires enabling EIGRP authentication on the interface and the configuration of the password within the key chain.

### Enabling Authentication on the interface

Enabling authentication for IOS nodes requires the following interface parameters configured with the following commands:

```
ip authentication mode eigrp as-number md5
```

```
ip authentication key-chain eigrp as-number key-chain-name
```

Enabling authentication for IOS XR nodes uses the command **authentication keychain** *key-chain-name* under the interface within the EIGRP process.

### Key Chain Configuration

Key chain creation is identical for IOS and IOS XR:

#### Step 1. Create the keychain.

The command **key chain** *key-chain-name* creates the local key chain.

#### Step 2. Identify the key sequence.

The key sequence is specified with the command **key** *key-number*, where the key number can be anything from 0 to 2147483647.

#### Step 3. Specify the password.

The preshared password is entered with the command **key-string** *text*.

#### Note

Be careful of using a space after the password because that will be used for computing the hash.

Example 5-34 provides a sample configuration of EIGRP authentication between an IOS and IOS XR node. Remember that the hash is computed using the key sequence number and key string and must match on both nodes.

### Example 5-34 EIGRP Authentication Configuration

[Click here to view code image](#)

```
IOS
key chain EIGRPKEY
  key 2
    key-string CISCO

interface GigabitEthernet0/0
  ip address 10.34.1.3 255.255.255.0
  ip authentication mode eigrp 100 md5
  ip authentication key-chain eigrp 100 EIGRPKEY
```

**IOS XR**

```
key chain EIGRPKEY
  key 2
    key-string password CISCO
  !
!
router eigrp 100
  address-family ipv4
    interface GigabitEthernet0/0/0/0
      authentication keychain EIGRPKEY
```

The command **show key chain** provides verification of the key chain. [Example 5-12](#) shows that each key sequence provides the lifetime and password. Notice that IOS-XR automatically encrypts the password from what was entered.

Notice that in [Example 5-35](#) that each key sequence provides the lifetime and password.

**Example 5-35** *Verification of Key Chain Settings*

[Click here to view code image](#)

```
IOS-ROUTER#show key chain
Key-chain EIGRPKEY:
  key 1 -- text "CISCO"
    accept lifetime (always valid) - (always valid) [valid now]
    send lifetime (always valid) - (always valid) [valid now]
```

```
RP/0/0/CPU0:IOSXR-Router#show key chain
```

```
Key-chain: EIGRPKEY/ -
Key 2 -- text "14343B382F2B"
  cryptographic-algorithm -- Not configured
  Send lifetime: Not configured
  Accept lifetime: Not configured
```

Looking at the EIGRP interface detail view provides verification of EIGRP authentication on a specific interface. [Example 5-36](#) provides detailed EIGRP interface output.

**Example 5-36** *Verification of EIGRP Authentication*

[Click here to view code image](#)

```
IOS-ROUTER#show ip eigrp interface detail
```

```
EIGRP-IPv4 Interfaces for AS(100)
```

Interface	Peers	Xmit Queue Un/Reliable	PeerQ Un/Reliable	SRTT	Mean Un/Reliable	Pacing Time Flow Timer	Multicast Routes	Pen
Gi0/0	0	0/0	0/0	0	0/0	50	0	

```
Hello-interval is 5, Hold-time is 15  
Split-horizon is enabled  
Next xmit serial <none>  
Packetized sent/expedited: 10/1  
Hello's sent/expedited: 673/12  
Un/reliable mcasts: 0/9 Un/reliable ucasts: 6/19  
Mcast exceptions: 0 CR packets: 0 ACKs suppressed: 0  
Retransmissions sent: 16 Out-of-sequence rcvd: 1  
Topology-ids on interface - 0  
Authentication mode is md5, key-chain is "EIGRPKEY"
```

```
RP/0/0/CPU0:IOSXR-Router#show eigrp interfaces detail
```

```
IPv4-EIGRP interfaces for AS(100)
```

```
Gi0/0/0/3      0      0/0      0      0/10      6392      0  
Hello interval is 5 sec, hold time is 15 sec  
Next xmit serial <none>  
Un/reliable mcasts: 0/3 Un/reliable ucasts: 9/6  
Mcast exceptions: 0 CR packets: 0 ACKs suppressed: 2  
Retransmissions sent: 1 Out-of-sequence rcvd: 0  
Bandwidth percent is 50  
Total packets received: 222  
Authentication mode: MD5 Key chain: EIGRPKEY  
No active key found in keychain database  
Valid authenticated packets received: 0  
Packets dropped due to wrong keychain config: 64  
Packets dropped due to missing authentication: 44  
Packets dropped due to invalid authentication: 0  
Effective Metric:  
Bandwidth: 1000000, Delay: 1, Reliability: 255, Load: 1, MTU: 1500
```

## WAN Considerations

EIGRP does not change behavior based on the media type of an interface. Serial and Ethernet interfaces are treated the same. Some WAN topologies may require special consideration for bandwidth utilization, split horizon and next-hop self. The following sections explain each scenario in more detail.

### IP Bandwidth Percent

RIP and other routing protocols can consume all of the bandwidth on slow circuits. Although the routers will have an accurate routing table, a router is worthless if no bandwidth is available for sending data packets. EIGRP overcomes this deficiency by setting the maximum available bandwidth for all circuits to 50 percent. This allows EIGRP to use 50 percent of the bandwidth and 50 percent of the bandwidth for data packets.

The interface parameter command **ip bandwidth-percent eigrp as-number percentage** changes the EIGRP available bandwidth for a link on IOS nodes. IOS XR nodes use the command **bandwidth-percent percentage** under the interface within the EIGRP process to accomplish the same task.

Note

Link speed and bandwidth are often confused. By default, bandwidth uses the link speed but can be changed with the **bandwidth** interface parameter command.

**Example 5-37** provides the EIGRP configuration for 25 percent bandwidth usages on the Ethernet links.

### **Example 5-37** EIGRP Bandwidth Percent Configuration

[Click here to view code image](#)

**IOS**

```
interface GigabitEthernet0/0
ip address 10.34.1.4 255.255.255.0
ip bandwidth-percent eigrp 100 25
```

**IOS XR**

```
router eigrp 100
address-family ipv4
interface GigabitEthernet0/0/0/0
bandwidth-percent 25
```

The EIGRP bandwidth settings are shown by looking at the EIGRP interfaces with the **detail** option. **Example 5-38** shows the EIGRP bandwidth settings.

### **Example 5-38** EIGRP Bandwidth Percent

[Click here to view code image](#)

```
R4#show ip eigrp interfaces detail
EIGRP-IPv4 Interfaces for AS(100)
          Xmit Queue  PeerQ      Mean  Pacing Time  Multicast  Pending
Interface Peers  Un/Reliable Un/Reliable SRTT  Un/Reliable Flow Timer  Routes
Gi0/0      1      0/0        0/0      1     0/0          50         0
  Hello-interval is 5, Hold-time is 15
  Split-horizon is enabled
  Next xmit serial <none>
  Packetized sent/expedited: 19/0
  Hello's sent/expedited: 1282/2
  Un/reliable mcasts: 0/8  Un/reliable ucasts: 21/10
  Mcast exceptions: 0  CR packets: 0  ACKs suppressed: 0
  Retransmissions sent: 2  Out-of-sequence rcvd: 1
  Topology-ids on interface - 0
  Interface BW percentage is 25
  Authentication mode is not set
```

```

RP/0/0/CPU0:XR#show eigrp interfaces gigabitEthernet 0/0/0/0 detail

IPv4-EIGRP interfaces for AS(100)

          Xmit Queue  Mean   Pacing Time  Multicast   Pending
Interface  Peers  Un/Reliable  SRTT   Un/Reliable  Flow Timer  Routes
Gi0/0/0/0   1      0/0         5      0/10        50          0
  Hello interval is 5 sec, hold time is 15 sec
  Next xmit serial <none>
  Un/reliable mcasts: 0/2  Un/reliable ucasts: 3/2
  Mcast exceptions: 0  CR packets: 0  ACKs suppressed: 1
  Retransmissions sent: 0  Out-of-sequence rcvd: 0
  Bandwidth percent is 25
  Total packets received: 1230
  Authentication mode is not set
  Effective Metric:
    Bandwidth: 1000000, Delay: 1, Reliability: 255, Load: 1, MTU: 1500

```

### Split Horizon

The first distance vector routing protocols advertised network prefixes out all interfaces for all known routes. Figure 5-32 demonstrates this behavior as three routers process the advertisements.

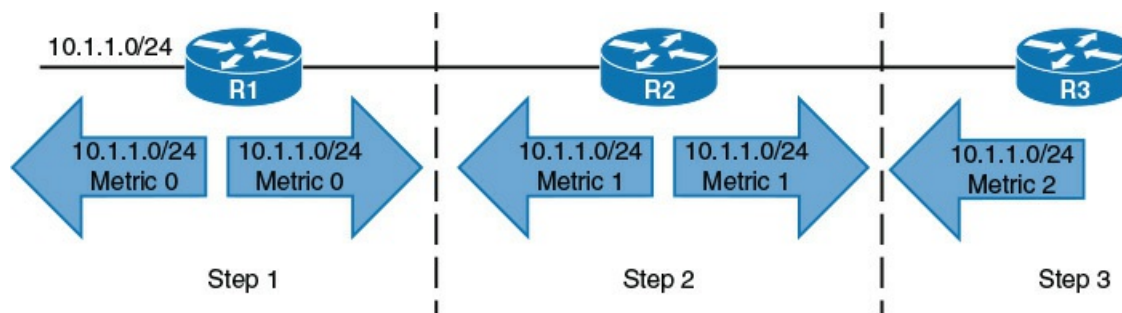


Figure 5-32 EIGRP Autonomous Systems

**Step 1.** R1 advertises the 10.1.1.0/24 network out all its interfaces.

**Step 2.** R2 adds to the metric and re-advertises the network to R1 and R3. Advertising a route (10.1.1.0/24) back to the originating router (R1) is known as a *reverse route*. Reverse routes waste network resources because R1 discards the route from R2, as the 10.1.1.0/24 is connected to the network and has a higher AD.

**Step 3.** R3 adds to the metric and advertises the reverse route to R2. R2 discards the route from R3 because it has a higher metric than the route from R1.

Figure 5-33 demonstrates a link failure between R1 and R2. R2 will remove the 10.1.1.0/24 route learned from R1. It is possible that before R2 announces that the 10.1.1.0/24 network is unreachable, R3 advertises the 10.1.1.0/24 route with a metric of 2 out all interfaces

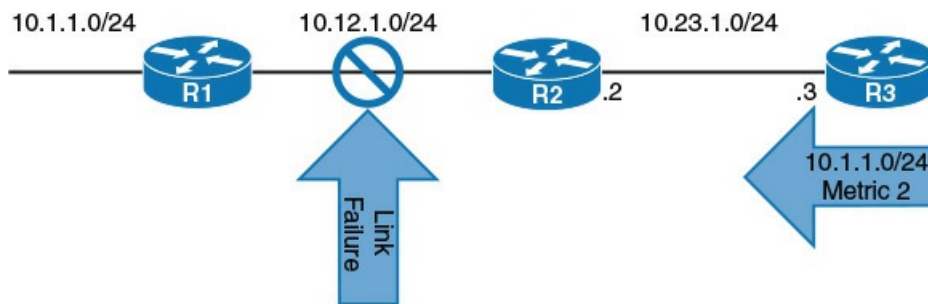


Figure 5-33 Link Failure Between R1 and R2

R2 will install the route advertised from R3, which has a next-hop IP address of 10.23.1.3. R3 will still maintain the original route advertised from R2 with a next-hop IP address of 10.23.1.2. This will cause a routing loop if a packet is sent from R2 or R3 to the 10.1.1.0/24 network. Eventually, the route entries will time out and end the routing loop.

Split horizon prevents the advertisement of reverse routes and prevents scenarios like that shown in Figure 5-33 from happening. Figure 5-34 shows the same scenario with split horizon.

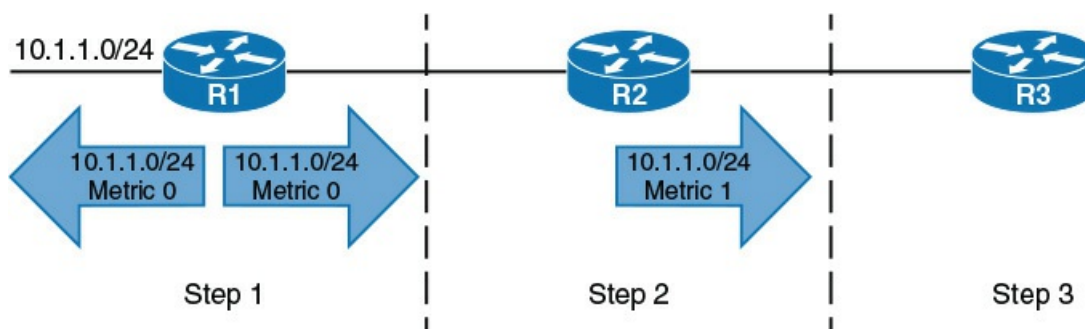


Figure 5-34 Routing Updates with Split Horizon Enabled

**Step 1.** R1 advertises the 10.1.1.0/24 network out all its interfaces.

**Step 2.** R2 adds to the metric and re-advertises the network to R3 but does not advertise the route back to R1 because of split horizon.

**Step 3.** R3 receives the route from R2 but does not advertise the route back to R2 because of split horizon.

EIGRP enables split horizon on all interfaces by default. When an interface connects to a multiaccess media that does not support full-mesh connectivity for all nodes, split horizon will need to be disabled. Commonly, this scenario is found on hub-and-spoke topologies such as Frame Relay, Dynamic Multipoint VPN (DMVPN), or Layer 2 Virtual Private Network (L2VPN).

Figure 5-35 demonstrates a hub-and-spoke topology where XR3 and R4 are the hubs. The spoke routers (XR2, R1, and R5) can communicate only with the hub router, and the hub routers (XR3, R4) use the same interface for both Frame Relay data-link connection identifiers (DLCIs) (circuits). Split horizon prevents routes received from one spoke from being advertised to the other spoke.

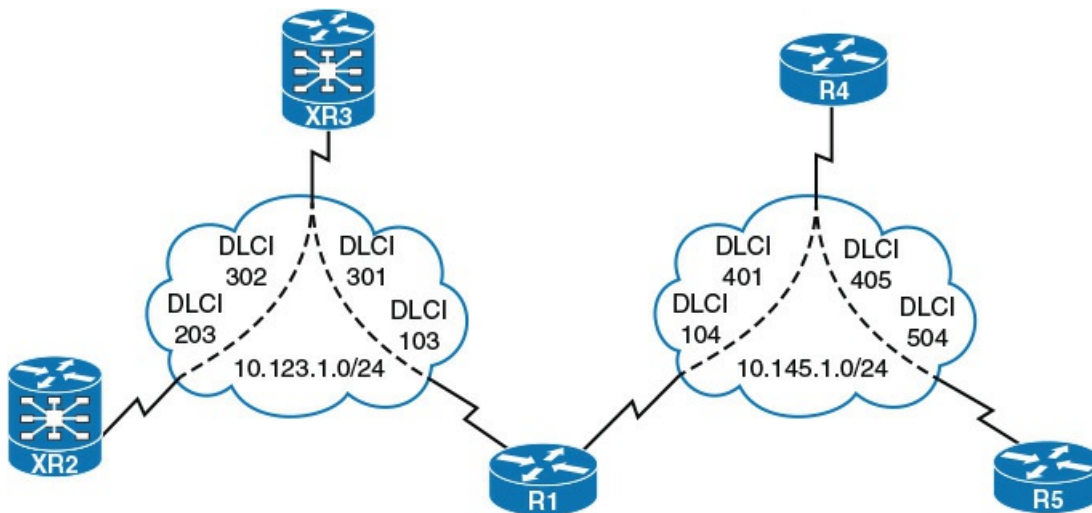


Figure 5-35 EIGRP Autonomous Systems

Every router advertises a loopback address in addition to the serial interfaces. Notice that the EIGRP routing table is not complete for all the routers. XR2 has only a route for the XR3's loopback address (192.168.3.3), and R5 has only a route for R4's loopback address (192.168.4.4). R1 does not have the route to XR2's loopback (192.168.2.2) or to R5's loopback address (192.168.5.5). Split horizon prevents routes received from one spoke from being advertised to the other spoke.

Split horizon on XR3 is preventing XR2's routes from being advertised to R1 and vice versa. Split horizon on R4 is blocking R1's routes from being advertised to R5 and vice versa. Disabling split horizon on XR3 and R4 will fix all connectivity issues for this topology.

Split horizon is disabled on a specific interface using the interface parameter command **no ip split-horizon eigrp as-number** on IOS nodes. The equivalent command for IOS XR nodes is **split-horizon disable** under the interface within the EIGRP process.

Example 5-39 shows XR3 and R4's configuration with split horizon disabled.

### Example 5-39 Configuration to Disable Split Horizon

[Click here to view code image](#)

```
XR3
router eigrp 100
 address-family ipv4
  interface Loopback0
  !
  interface Serial0/1/0/0
  split-horizon disable
```

```
R4
interface Serial11/0
 ip address 10.145.1.4 255.255.255.0
 no ip split-horizon eigrp 100
 encapsulation frame-relay
```



Figure 5-36 shows the routing table of all the routers after disabling split-horizon on XR3 and R4. Notice that all routers have complete EIGRP topologies.

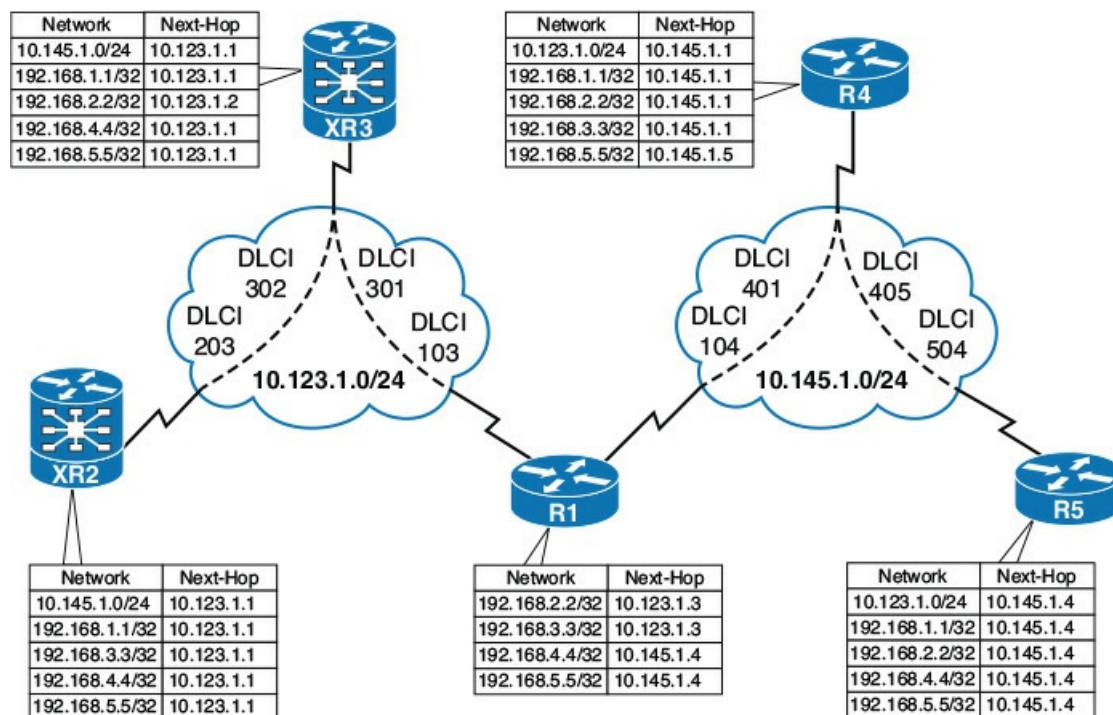


Figure 5-36 EIGRP Autonomous Systems with Split-Horizon Disabled

### Next-Hop Self

EIGRP sets the next-hop IP address within the update packet to the IP address of the interface advertising the route. This informs the downstream router how to populate the next-hop forwarding address for its route table.

DMVPN provides scalable virtual private networks (VPNs) through a centralized architecture. Branch routers communicate with the centralized site in a hub-to-spoke fashion, but direct spoke-to-spoke communication can occur dynamically. Although the spokes can communicate directly with each other, routes are exchanged through the hub. This requires the hub router not to advertise itself as the next-hop IP address for spoke-to-spoke communication to work properly.

In Figure 5-37, R2 and R3 have established DMVPN tunnels with R1. R3 advertises the 10.3.3.0/24 network to R1. R1 advertises the 10.3.3.0/24 network to R2 after modifying the next-hop IP address to 10.123.1.1. Traffic from R2 to the 10.3.3.0/24 will flow through the VPN tunnel to XR1 and down to XR3.

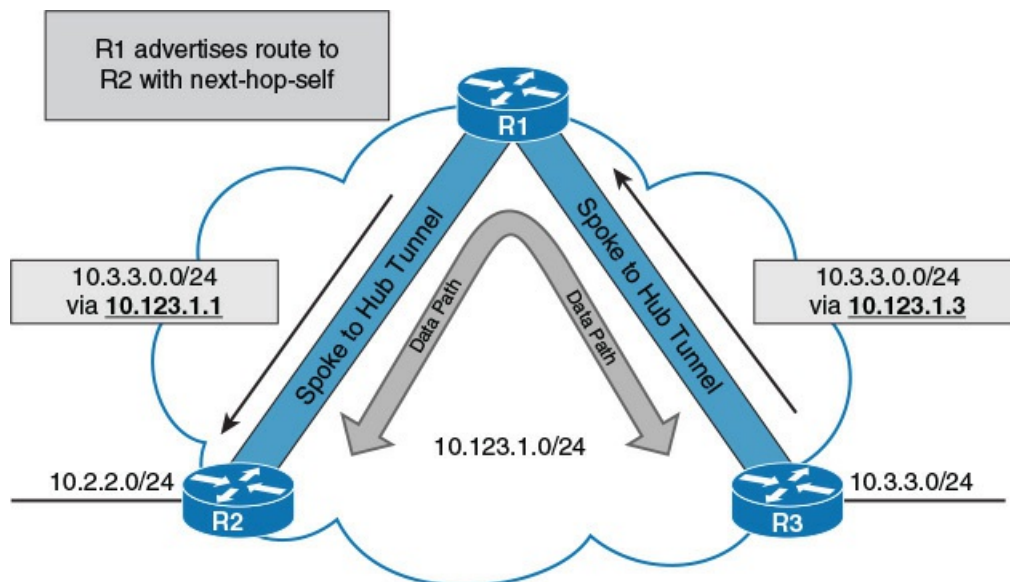


Figure 5-37 Route Advertisement with Next-Hop Address Modified

EIGRP's default behavior of modifying the next-hop address can be disabled. By disabling the next-hop self behavior in EIGRP, EIGRP will advertise the next-hop IP address that it maintains in the topology table.

To disable the next-hop self behavior on an IOS node, use interface parameter command **no ip next-hop-self eigrp as-number**. The command **next-hop-self disable** under the interface within the EIGRP process disables next-hop self on IOS XR nodes.

**Example 5-40** demonstrates the EIGRP configuration for R1 to stop the modification of the next-hop IP address as the routers are advertised. DMVPN configuration is beyond the scope of this book, so the configuration is limited to only EIGRP to demonstrate the concept.

#### Example 5-40 EIGRP Configuration for Next-Hop Self Disabled

[Click here to view code image](#)

```
R1
! Configuration related to DMVPN has been removed as it is outside the scope
! of this book
interface Tunnel
 ip address 10.123.1.1 255.255.255.0
 no ip next-hop-self eigrp 100
 no ip split-horizon eigrp 100
```

In **Figure 5-38**, R2 and R3 have established DMVPN tunnels with R1. XR3 advertises the 10.3.3.0/24 network to R1. XR1 is configured to not modify the next-hop IP address and advertises the 10.3.3.0/24 network to R2 with the original next-hop IP address of 10.123.1.3. Traffic from R2 to the 10.3.3.0/24 will flow through the spoke-to-spoke VPN tunnel between R2 and R3 bypassing R1.

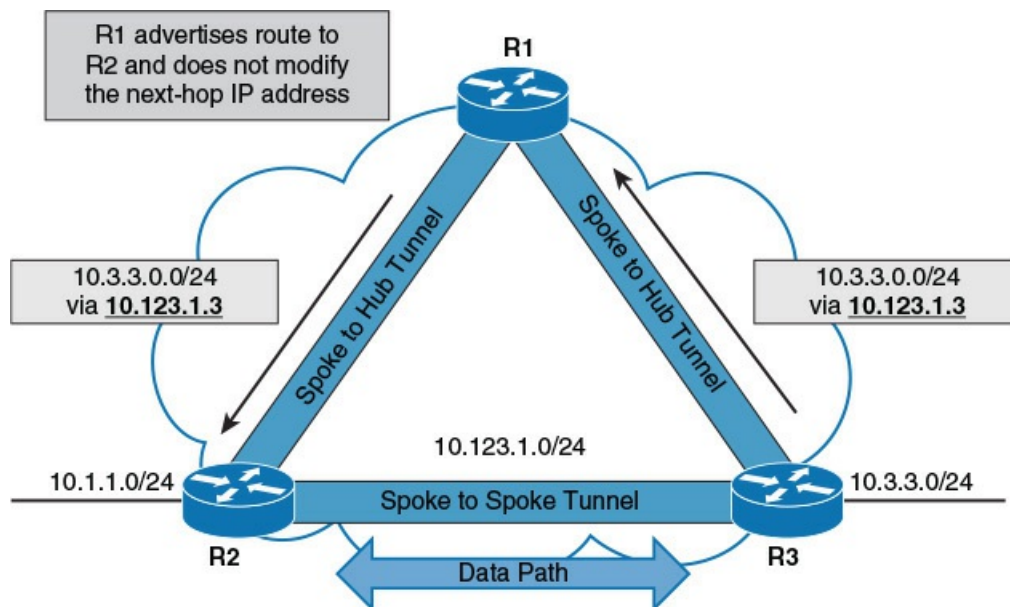


Figure 5-38 Route Advertisement Without Modifying Next-Hop Address

## EIGRP NAMED CONFIGURATION

IOS Releases 15.0(1)M and later introduced a new method of EIGRP configuration called *EIGRP named configuration*. EIGRP named configuration overcomes some of the difficulties network engineers have with the classic EIGRP autonomous system configuration, such as the following:

- Scattered configurations
- Unclear scope of commands

EIGRP named configuration provides the following benefits:

- All the EIGRP configuration under one location.
- Supports current EIGRP features and future developments.
- Support of multiple address families (including Virtual Routing and Forwarding [VRF] instances).
- Commands are clear to the scope of their configuration.

### Note

The two styles of EIGRP configuration are independent. Using the configuration options from classic EIGRP autonomous system configuration will not modify settings on a router running EIGRP named configuration.

### Address Family Instance Configuration

EIGRP named configuration supports running multiple instances under the same EIGRP process. The process for enabling EIGRP interfaces on a specific instance is as follows:

#### Step 1. Initialize the EIGRP process.

The EIGRP process is initialized with the command **router eigrp process-name**.

## Step 2. Define the instance.

The EIGRP instance is initialized for the appropriate address family with the command **address-family {IPv4| IPv6} {unicast | vrf vrf-name} autonomous-system as-number**. Multiple instances can exist as long as the ASN is unique.

## Step 3. Enable the interface.

EIGRP is enabled on interfaces with the command `network network wildcard-mask` and works in the same manner as EIGRP autonomous system configuration mode.

[Example 5-41](#) provides a sample EIGRP named configuration where the process is named EIGRP and all the interfaces are enabled to use the protocol. The *topology base* portion of the configuration is created automatically and is discussed later in this section.

### Example 5-41 Sample EIGRP Named Configuration

[Click here to view code image](#)

```
router eigrp CISCO
!
address-family ipv4 unicast autonomous-system 100
!
topology base
exit-af-topology
network 0.0.0.0
exit-address-family
```

[Table 5-7](#) provides a list of relevant address family instance configuration commands covered in this chapter. The last two commands are used to change the context of configuration in EIGRP named configuration mode.

Command	Description
<code>eigrp router-id router-id</code>	Statically sets the EIGRP route ID to a 32-bit value.
<code>eigrp stub {connected   receive-only   redistributed   static   summary}</code>	Configures the active router to be an EIGRP stub router. The default stub value is connected and summary routes. Note that the <code>receive-only</code> option cannot be combined.
<code>network network {subnet-mask   /prefix-length}</code>	Enables EIGRP on interfaces matching within the wildcard mask range.
<code>metric weights TOS K1 K2 K3 K4 K5 [K6]</code>	Sets the K values used for the path metric calculation.
<code>af-interface {default   interface-type interface-number}</code>	Enters the address family interface configuration mode for all interfaces or for a specific interface.
<code>topology base</code>	Enters the address family topology configuration mode for the base topology.

Table 5-7 Address Family Instance Commands

Note

The topology terminology comes from multitopology routing (MTR), which supports different forwarding (routing) patterns based on quality of service (QoS) markings in a packet. The base topology encompasses any subtopologies. Content outside of the base topology is not covered in this book.

### Address Family Interface Configuration

EIGRP-specific interface configurations are placed in the address family interface configuration. Configuration can be entered for each interface or could be placed under the **af-interface default**, which sets the parameter for all EIGRP interfaces in that instance. Interface-specific configurations will always preempt the **af-interface default** parameter.

Table 5-8 provides a list of relevant address family interface configuration commands covered in this chapter.

Command	Description
<b>authentication key-chain</b> <i>key-chain-name</i>	Enables authentication on interface and specifies the key chain associated for authentication.
<b>authentication mode</b> {md5   hmac-sha-256 <i>password</i> }	Specifies the encryption mechanism when enabling authentication on an interface.  Note that SHA encryption does key chain functionality and does not support authentication key rotation.
<b>bandwidth-percent</b> <i>percent</i>	Assigns the maximum percentage of bandwidth used on an interface for EIGRP traffic.
<b>hello-interval</b> <i>seconds</i>	Sets the frequency that EIGRP packets are sent out an interface.
<b>hold-time</b> <i>seconds</i>	Sets the timer that an EIGRP hello must be received before declaring a neighbor dead.
<b>next-hop-self</b>	Sets the next-hop IP address to the interface advertising the update.
<b>passive-interface</b>	Sets the listed interface as Passive.
<b>split-horizon</b>	Combined with the <b>no</b> keyword can disable split horizon on an interface.
<b>summary-address</b> <i>network</i> {/ <i>prefix-length</i>   <i>subnet-mask</i> }	Configures an EIGRP network summarization range for networks advertised out this interface. Network updates within this range are suppressed from leaving this interface.  Note that this option is not available on <b>af-interface default</b> .

Table 5-8 Address Family Interface Configuration Commands

Example 5-42 demonstrates the address family interface configuration. R1 has enabled EIGRP on all interfaces, set the hello timer to 10 seconds, and set the hold timer to 30 seconds under the **af-interface default**. The Gigabit Ethernet 0/0 interface has the hello timer set to 5 seconds and the hold timer set to 15 seconds to demonstrate inheritance and preemption.

#### Example 5-42 Example of Address Family Interface Configuration

[Click here to view code image](#)

```

router eigrp EIGRP
!
address-family ipv4 unicast autonomous-system 100
!
af-interface default
hello-interval 10
hold-time 30
exit-af-interface
!
af-interface GigabitEthernet0/0
hello-interval 5
hold-time 15
exit-af-interface
!
topology base
exit-af-topology
network 0.0.0.0
exit-address-family

```

**Example 5-43** verifies that all the EIGRP-enabled interfaces use a hello timer of 10 seconds and a hold time of 30 seconds. Notice that Gigabit Ethernet 0/0 preempts the **af-interface default** configuration with the explicitly configured 5-second hello timer and 15 second hold timer.

### **Example 5-43** *Verification of Inheritance*

[Click here to view code image](#)

```

R1#show ip eigrp interfaces detail | include Gi0|Hold
Gi0/0      1      0/0      0/0      1      0/2      50      0
  Hello-interval is 5, Hold-time is 15
Gi0/1      1      0/0      0/0      4      0/2      50      0
  Hello-interval is 10, Hold-time is 30
Gi0/2      1      0/0      0/0      3      0/2      50      0
  Hello-interval is 10, Hold-time is 30

```

### **Address Family Topology Configuration**

Address family topology configuration contains specific configuration options like SIA timers, redistribution, and maximum paths. [Table 5-9](#) provides a list of the relevant address family topology configuration commands covered earlier in this chapter.

Command	Description
<code>auto-summary</code>	Displays the EIGRP interfaces
<code>maximum-paths [1-32]</code>	Configures the number of paths that can install into the topology table for a prefix
<code>summary-metric network {/prefix-length   subnet-mask} bandwidth delay reliability load MTU [AD]</code>	Sets fixed metrics for EIGRP summary network addresses
<code>timers active-time {disabled   1-65535_minutes}</code>	Sets the maximum amount of time that a route can be “active” during path computation
<code>variance variance-multiplier</code>	Specifies the variance multiplier for unequal load balancing

Table 5-9 Address Family Interface Configuration Commands

Example 5-44 demonstrates the address family topology configuration. R1 has enabled EIGRP on all interfaces, set the SIA timer to 4 minutes, and set the variance multiplier to 3.

### Example 5-44 Address Family Interface Configuration Example

[Click here to view code image](#)

```
router eigrp EIGRP
!
address-family ipv4 unicast autonomous-system 100
!
  topology base
  variance 3
  timers active-time 4
exit-af-topology
network 0.0.0.0
exit-address-family
```

#### Note

Classic EIGRP autonomous system configuration can be upgraded to an EIGRP named mode with the command `eigrp upgrade-cli`.

## SUMMARY

EIGRP is an enhanced distance vector routing protocols that overcomes the problems associated with other distance vector routing protocols. It is highly valued for its ease of deployment and fast convergence. EIGRP is commonly used in many large enterprise networks. EIGRP provides fast convergence and uses logic to prevent routing loops that are commonly found in normal distance vector routing protocols.

## REFERENCES IN THIS CHAPTER

Informational RFC, *Enhanced Interior Gateway Routing Protocol*, D. Savage, D. Slice, J. Ng, S. Moore, R. White, IETF, <http://tools.ietf.org/html/draft-savage-eigrp-02>, April 2014.

Cisco. *Cisco IOS Software EIGRP configuration guides*. <http://www.cisco.com>

Cisco. *Cisco IOS XR Software EIGRP configuration guides*. <http://www.cisco.com>





## Chapter 6. OSPF

This chapter covers the following topics:

- OSPF fundamentals
- Inter-router communication
- OSPF network types
- OSPF configuration
- Failure detection
- OSPF security

The Open Shortest Path First (OSPF) Protocol is the first link-state routing protocol covered in this book. OSPF is a nonproprietary interior gateway protocol (IGP) that overcomes the deficiencies of other distance vector routing protocols and distributes routing information within a single OSPF routing domain. OSPF introduced concepts of variable-length subnet mask (VLSM), which supports classless routing, summarization, authentication, and external route tagging, and was later expanded to provide IPv6 support in OSPFv3.

### OSPF FUNDAMENTALS

OSPF advertises link-state advertisements (LSAs) that contain the link state and link metric to neighboring routers. Received LSAs are stored in a local database called the *link-state database* (LSDB) and advertise the link-state information to neighboring routers exactly as the original advertising router advertised it. This process floods the LSA throughout the OSPF routing domain just as the advertising router advertised it. All OSPF routers maintain a synchronized identical copy of the LSDB.

The LSDB provides the topology of the network, in essence providing the router a complete map of the network. All OSPF routers run the Dijkstra Shortest Path First (SPF) algorithm to construct a loop-free topology of shortest paths. OSPF dynamically detects topology changes within the network and calculates loop-free paths in a short amount of time with minimal routing protocol traffic.

Each router sees itself as the root or top of the SPF tree (SPT), and the SPT contains all network destinations within the OSPF domain. The *SPT* will differ for each OSPF router, but the LSDB used to calculate the SPT is identical for all OSPF routers.

Figure 6-1 demonstrates a simple OSPF topology, and the SPT from R1's and R4's perspective. Notice that the local router's perspective will always be the root (top of the tree). There is a difference in connectivity to the 10.3.3.0/24 network from R1's and R4's SPT. From R1's perspective, the serial link between R3 and R4 is missing; and from R4's perspective, the Ethernet link between R1 and R3 is missing.

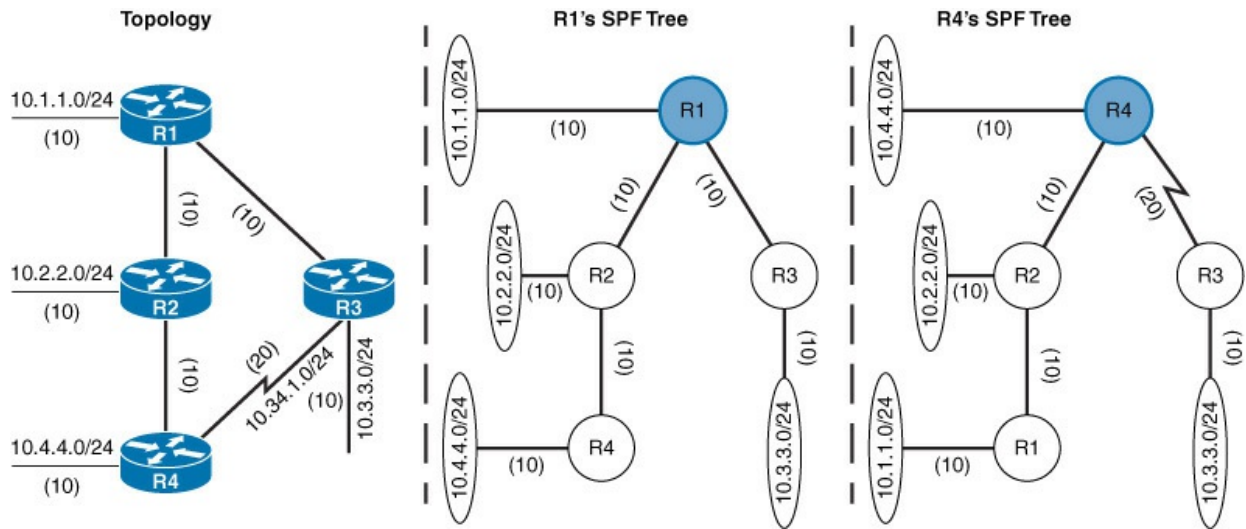


Figure 6-1 OSPF SPF Tree

The SPTs give the illusion that no redundancy exists to the networks, but remember that the SPT shows the shortest path to reach a network, and is built from the LSDB, which contains all the links for an area. During a topology change, the SPT is rebuilt and may change.

Note

OSPF builds the topology on top of IPv4 links, so network prefixes are interpreted as nodes.

OSPF provides scalability for the routing table by using multiple OSPF areas within the routing domain. Each OSPF area provides a collection of connected networks and hosts that are grouped together. OSPF uses a two-tier hierarchical architecture, where Area 0 is a special area known as the *backbone* and all other areas must connect to Area 0. In other words, Area 0 provides transit connectivity between nonbackbone areas. Nonbackbone areas advertise routes into the backbone, and the backbone then advertises routes into other nonback areas.

Figure 6-2 shows the route advertisement into other areas. Area 12 routes are advertised to Area 0 and then into Area 34. Area 34 routes are advertised to Area 0 and then into Area 12. Area 0 routes are advertised into all other OSPF areas.

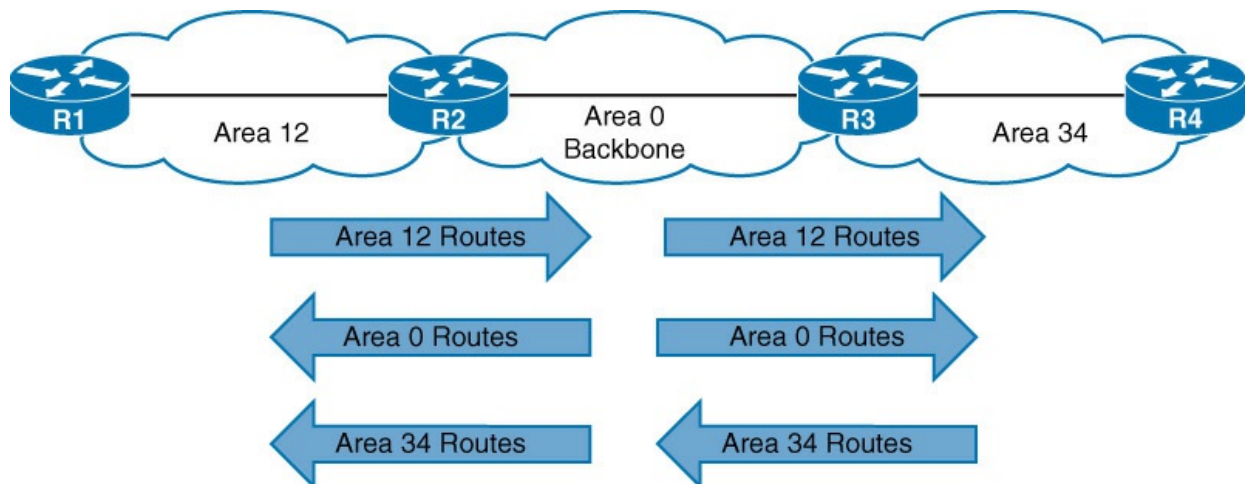


Figure 6-2 Two-Tier Hierarchical Area Structure

The exact topology of the area is invisible from outside of the area while still providing connectivity to routers outside of the area. This means that routers outside the area do not have a complete topological map for that area, which reduces network traffic in that area. By segmenting an OSPF routing domain into multiple areas, it is no longer true that all OSPF routers will have identical LSDBs; however, all routers *within the same area* will have identical area LSDBs.

The reduction in routing traffic uses less router memory and resources providing scalability. The next chapter explains areas in greater depth; this chapter focuses on the core OSPF concepts. For the remainder of this chapter, OSPF Area 0 is used as a reference area.

A router can run multiple OSPF processes. Each process maintains its own unique database, and routes learned in one OSPF process are not available to a different OSPF process without redistribution of routes between processes. The OSPF process numbers are locally significant and do not have to match among routers. Running OSPF process number 1 on one router and running OSPF process number 1234 will still allow the two routers to become neighbors.

### Inter-Router Communication

OSPF runs directly over IPv4 using its own protocol 89 reserved for OSPF by Internet Assigned Numbers Authority (IANA). OSPF uses multicast where possible to reduce unnecessary traffic. The two OSPF multicast addresses are as follows:

- **AllSPFRouters:** IPv4 address 224.0.0.5 or MAC 01:00:5E:00:00:05. All routers running OSPF should be able to receive these packets.

- **AllDRouters:** IPv4 address 224.0.0.6 or MAC 01:00:5E:00:00:06. Communication with designated routers (DRs) uses this address.

Within the OSPF protocol, five types of packets are communicated. [Table 6-1](#) briefly describes these OSPF packet types.

<b>Type</b>	<b>Packet Name</b>	<b>Functional Overview</b>
1	Hello	<b>Discover a maintain neighbors</b> Packets are sent out periodically on all OSPF interfaces to discover new neighbors while ensuring that other neighbors are still online.
2	Database description (DBD) or (DDP)	<b>Summarize database contents</b> Packets are exchanged when an OSPF adjacency is first being formed. These packets are used to describe the contents of the LSDB.
3	Link-state request (LSR)	<b>Database download</b> When a router thinks that part of its LSDB is stale, it may request a portion of a neighbor's database using this packet type.
4	Link-state update (LSU)	<b>Database update</b> This is an explicit LSA for a specific network link, and normally is sent in direct response to an LSR
5	Link-state ack	<b>Flooding acknowledgement</b> These packets are sent in response to the flooding of LSAs, thus making the flooding a reliable transport feature.

**Table 6-1** *OSPF Packet Types*

### **OSPF Hello Packets**

OSPF Hello packets are responsible for discovering and maintaining neighbors. In most instances, the router sends Hello packets to the AllSPFRouters address (224.0.0.5). [Table 6-2](#) lists some of the data contained within an OSPF Hello packet.

Data Field	Description
Router ID (RID)	A unique 32-bit ID within an OSPF domain.
Authentication Options	Allows secure communication between OSPF routers to prevent malicious activity. Options are none, clear text, or message digest 5 (MD5) authentication.
Area ID	OSPF area that the OSPF interface belongs to. It is a 32-bit number that can be written in dotted-decimal format (0.0.1.0) or decimal (256).
Interface Address Mask	The network mask for the primary IP for the interface that the Hello is sent out.
Interface Priority	Router interface priority for DR elections.
Hello Interval	Time span, in seconds, that a router sends out Hello packets on the interface.
Dead Interval	Time span, in seconds, that a router will wait to hear a Hello from a neighbor router before it declares that router down.
Designated Router & Backup Designated Router	IP address of the DR and backup DR (BDR) for that network link.
Active Neighbor	A list of OSPF neighbors seen on that network segment. Must have received a Hello from the neighbor within the dead interval.

Table 6-2 OSPF Hello Fields

Figure 6-3 shows a packet capture of an OSPF Hello and the appropriate data fields from Table 6-2.

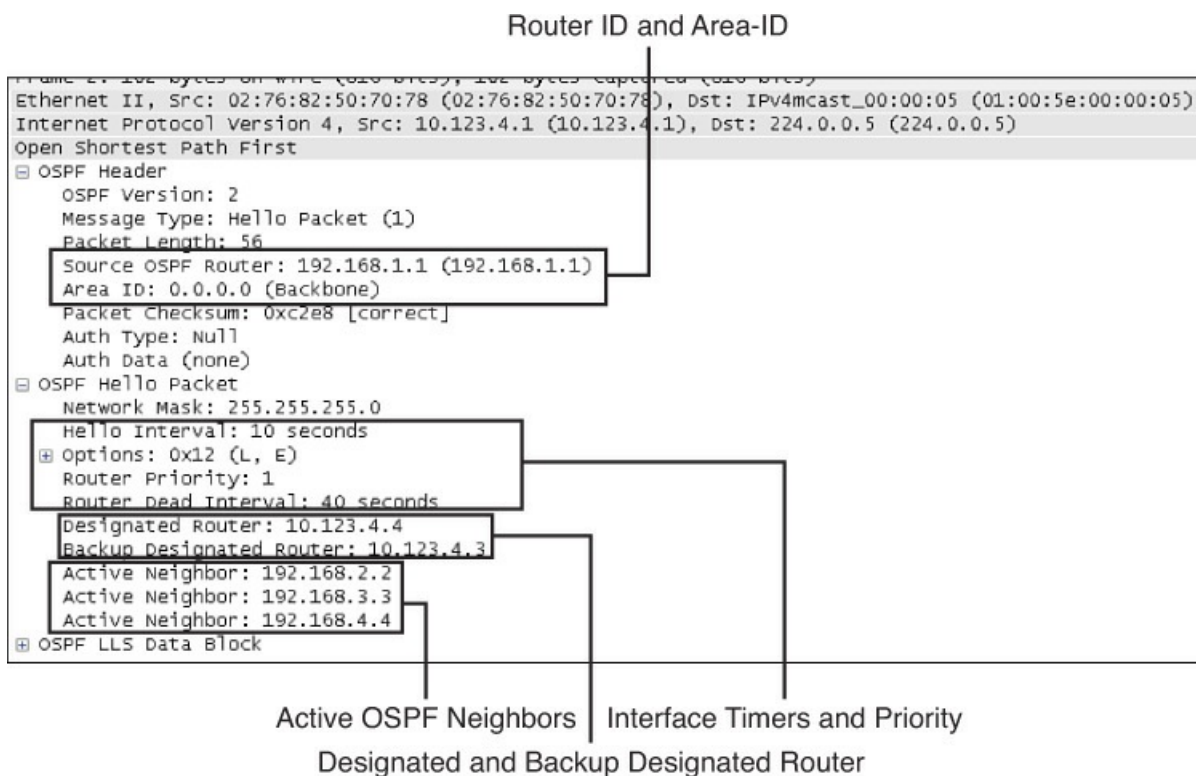


Figure 6-3 OSPF Hello Packet

## Router ID

The OSPF router ID (RID) is a 32-bit number that uniquely identifies an OSPF router. In some OSPF output commands, *neighbor ID* refers to the RID because the terms are synonymous. The RID must be unique for each OSPF process in an OSPF domain, and must be unique between OSPF processes on a router. Dynamic allocation logic varies from IOS to IOS XR:

■ **IOS:** IOS nodes use the highest IP address of the any up loopback interfaces. If there are no up loopback interfaces, the highest IP address of any active up physical interfaces becomes the RID when the OSPF process initializes.

■ **IOS XR:** IOS XR nodes use the IP address of the lowest up loopback interface. If there are no up loopback interfaces, the lowest interface IP address of any active up physical interface becomes the RID when the OSPF process initializes.

RIDs typically represent an IPv4 address that resides on the router, such as loopback address. Any IPv4 address can be used, including IP addresses not configured on the router. The command **router-id** *router-id* statically assigns the OSPF RID under the OSPF process.

The OSPF process selects the RID when the OSPF process initializes, and will not change until the process restarts on IOS nodes. The command **clear ip ospf process** restarts the OSPF process on IOS nodes so that OSPF can use the new RID. On IOS XR nodes, the change processes immediately upon the commit, and will reset adjacencies with other OSPF neighbor routers.

Interface changes (addition/removal of IP addresses) on an IOS node will be detected when the OSPF process restarts, and the RID will change accordingly. IOS XR uses a checkpoint during OSPF process restarts, and will not select a new RID even if the dynamic allocation method is used.

### Note

The OSPF topology is built upon the RID. Setting a static RID helps with troubleshooting, and reduces LSAs when a RID changes in an OSPF environment. Specifying the IP address for a loopback interface is the most common technique, and most organizations provide a block of network addresses for loopback interfaces.

## Neighbors

An OSPF neighbor is a router that shares a common OSPF-enabled network link. OSPF routers discover other neighbors via the OSPF Hello packets. An adjacent OSPF neighbor is an OSPF neighbor that shares a synchronized OSPF database between the two neighbors.

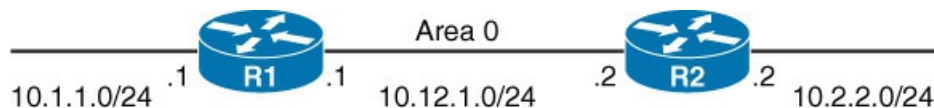
Each OSPF process maintains a table for adjacent OSPF neighbors, and the state of each router. [Table 6-3](#) briefly describes the OSPF neighbor states.

State	Description
Down	Initial state of a neighbor relationship. It indicates that the router has not received any OSPF Hello packets.
Attempt	This state is relevant to NBMA networks that do not support broadcast and require explicit neighbor configuration. This state indicates that no recent information has been received, but the router is still attempting communication.
Init	A Hello packet has been received from another a router, but bidirectional communication has not been established.
2-Way	Bidirectional communication has been established. If a DR or BDR is needed, the election occurs during this state.
ExStart	This is the first state of forming an adjacency. Routers identify which router will be the master or slave for the LSDB synchronization.
Exchange	During this state, routers are exchanging link states via DBD packets.
Loading	LSR packets are sent to the neighbor asking for the more recent LSAs that have been discovered (but not received) in the Exchange state.
Full	Neighboring routers are fully adjacent.

**Table 6-3** *OSPF Neighbor States*

### Forming OSPF Neighbor Adjacencies

Figure 6-4 demonstrates a simple topology of two OSPF routers trying to form a neighbor adjacency. R1 and R2 share a common network link (10.12.1.0/24), and have enabled OSPF for Area 0, including their loopback addresses. The process for forming an OSPF adjacency follows.



**Figure 6-4** *Simple OSPF Topology*

**Step 1.** R1 sends out a Hello packet. R1 has not detected an OSPF Hello packet from another OSPF router. Notice that the Active Neighbor field is not in the Hello packet in [Figure 6-5](#).



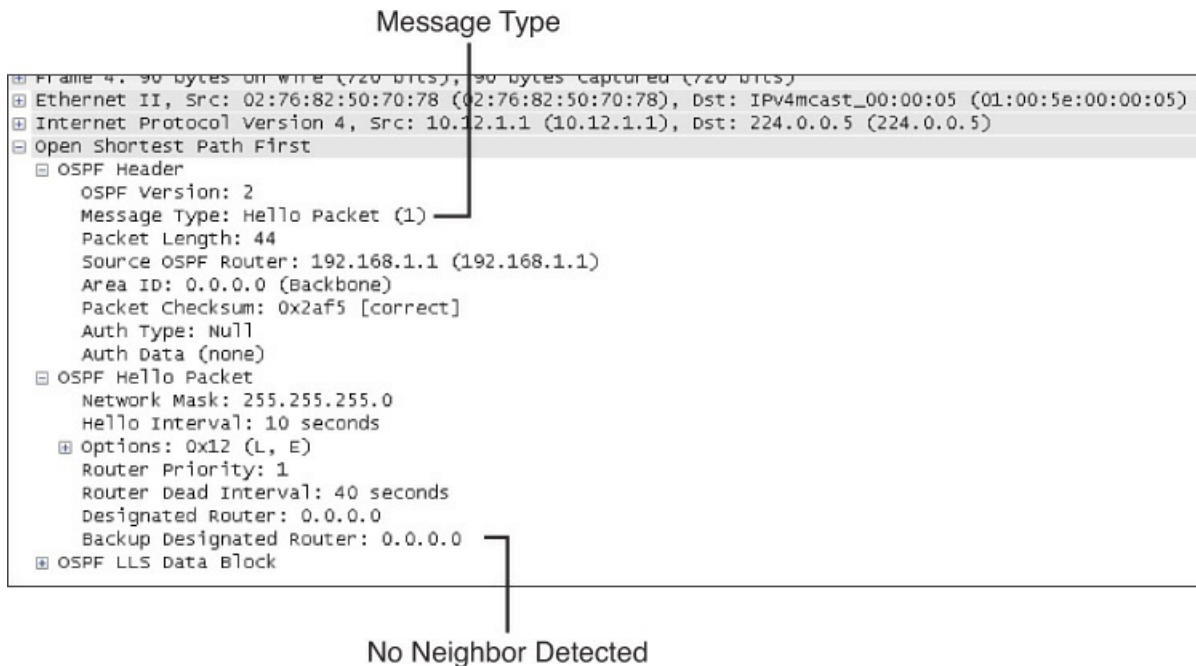


Figure 6-5 OSPF Hello with No Neighbors Detected

**Step 2.** R2 sends out a Hello packet. R2 has received R1's OSPF Hello packet from Step 1, and lists R1's RID as an active neighbor. Notice that the Active Neighbor field is in [Figure 6-6](#).



Figure 6-6 OSPF Hello Packet with Neighbors Detected

**Step 3.** R1 and R2 both verify that the following parameters match:

- Authentication type and credentials (if any).
- Area ID.
- OSPF hello and dead timers.
- RIDs are unique.



- Ensures that the interfaces share a common subnet. OSPF uses the interface’s primary IP address when sending out OSPF Hellos. The *network mask (netmask)* in the Hello packet is used to extract the network ID of the Hello packet. (The only exception is for point-to-point OSPF network types and IP unnumbered interfaces.)

- MTU (maximum transmission unit). The OSPF protocol does not support the fragmentation, so the MTU on the interfaces should match. Adjacencies may form with the IOS configuration command **ip ospf mtu-ignore** or IOS XR configuration command **mtu-ignore**, but could cause issues later on. It is recommended that IP MTU matches between OSPF neighbors.

- Need for DR on the segment.

- Area type flags match (discussed in Chapter 7, “Advanced OSPF”).

**Step 4.** R1 and R2 decide which router will be the master and thus control the database synchronization process. The router with the higher RID is elected the master.

**Step 5.** R1 and R2 exchange DBD packets one at a time. A DBD packet is unicast and may contain several LSAs headers within the packet. The receiving router sends an acknowledgement, and a follow-up DBD packet is sent. This process continues until the LSDB headers are synchronized.

Figure 6-7 shows the DBD packet. Notice the DD sequence number.

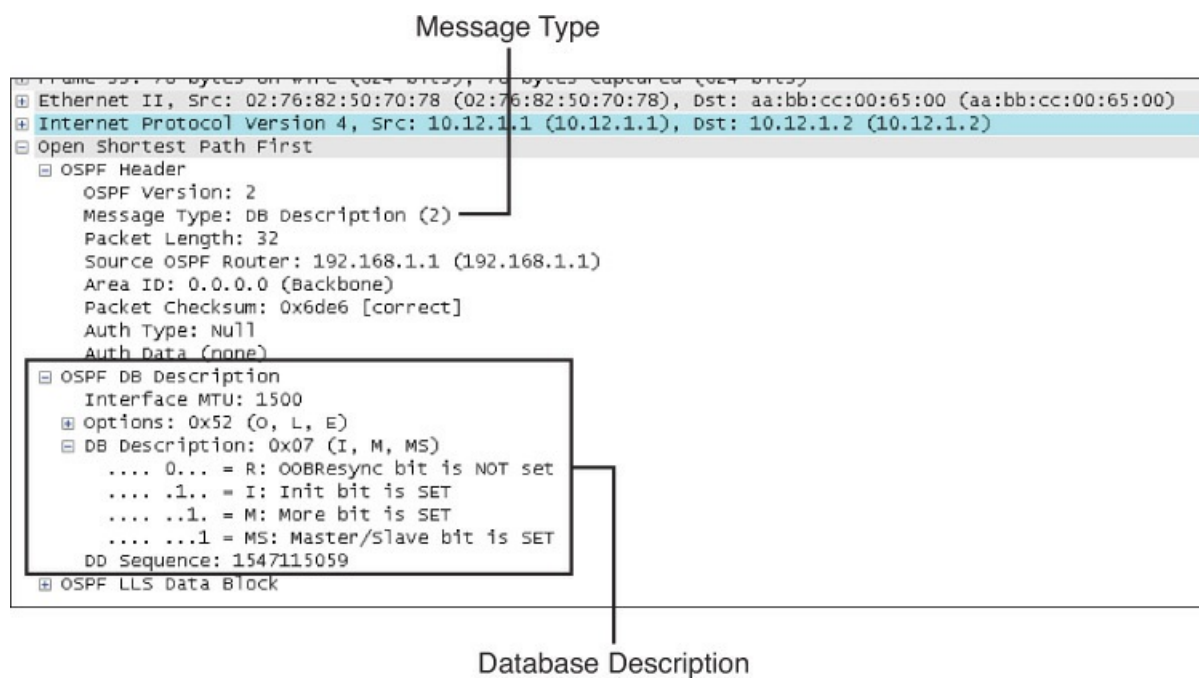


Figure 6-7 OSPF DDP Packet

**Step 6.** After R1 and R2 have a full copy of the LSDB headers, any missing specific LSAs referenced in the headers are requested from the other router using a unicast LSR.

Figure 6-8 displays R1 sending a LSR to R2.

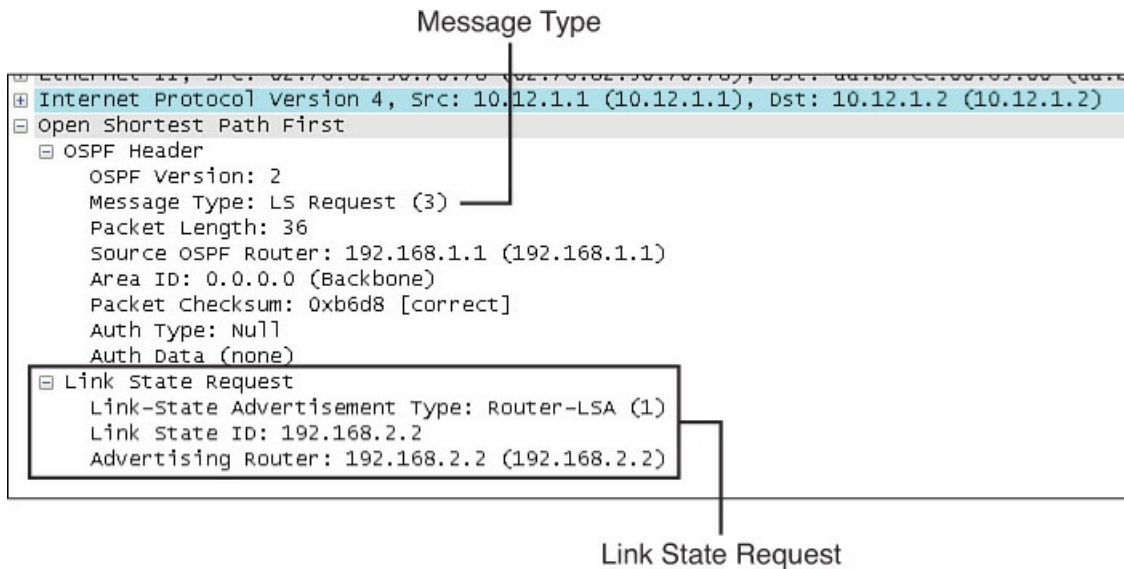


Figure 6-8 OSPF Link-State Request

**Step 7.** The missing LSAs are provided to the LSR request via a unicast LSU. Figure 6-9 shows R2 sending the LSU to R1. Notice that R1 is providing all three network entries in the LSU.

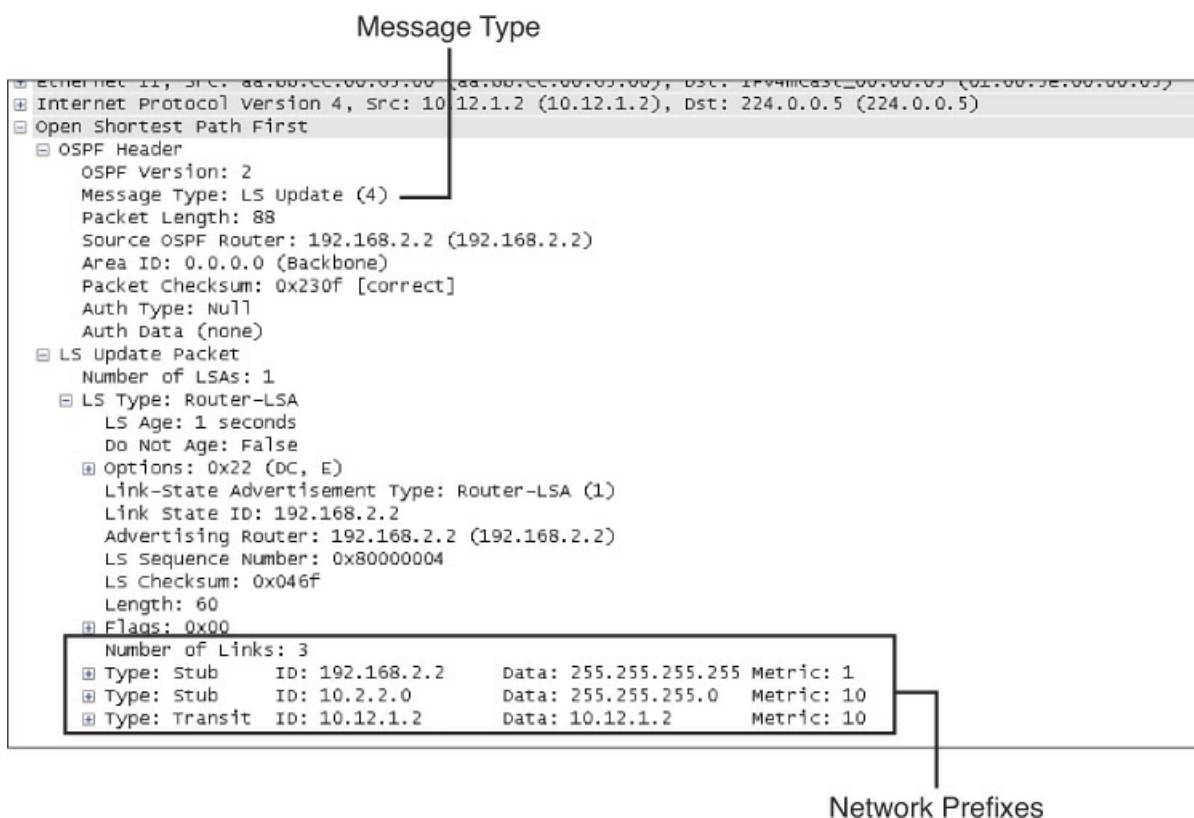
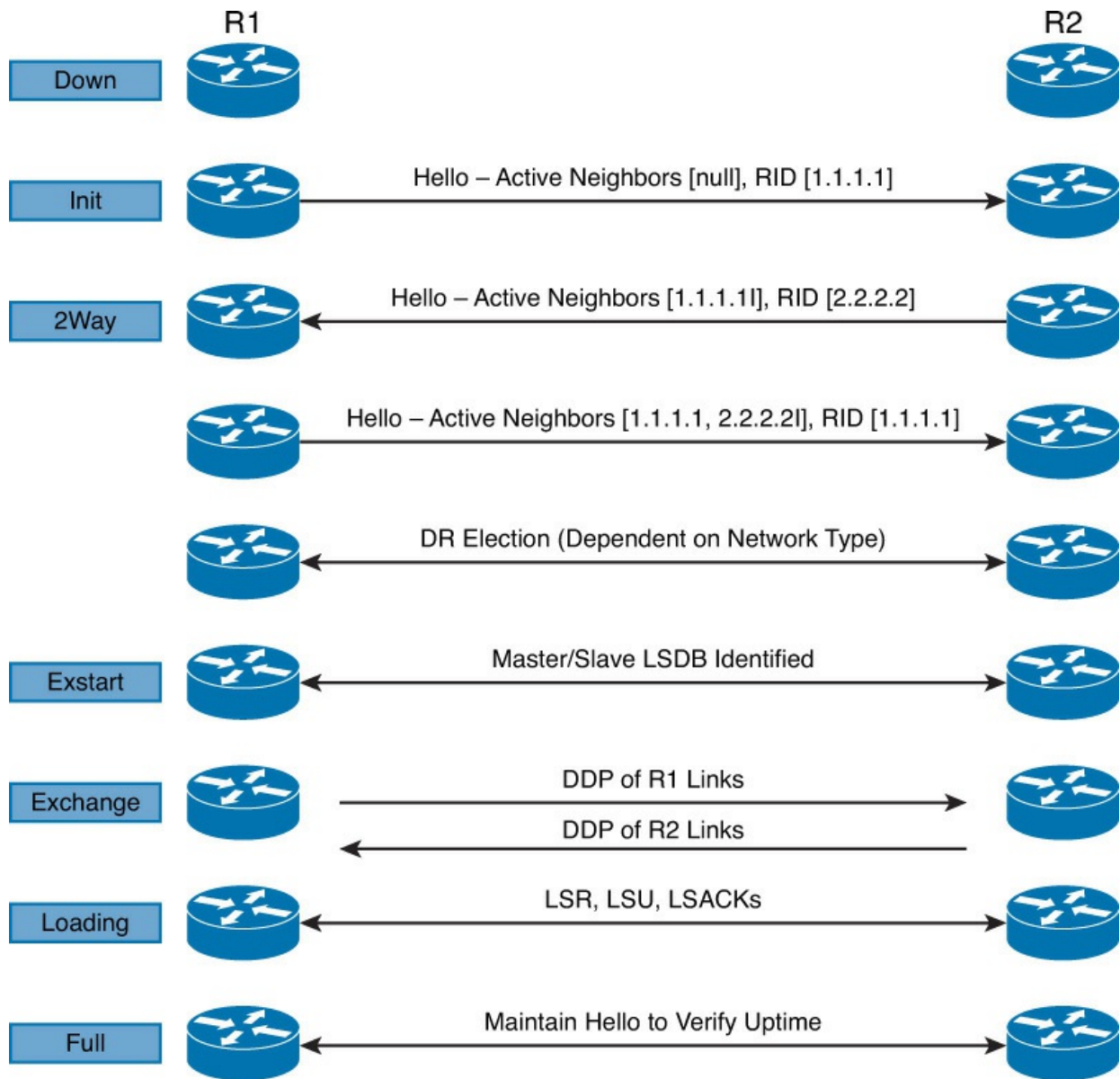


Figure 6-9 OSPF Link-State Update

**Step 8.** After both routers have fully loaded their LSDB with headers and LSAs, both LSDBs are synchronized, and the routers transition into the Full state.

**Step 9.** Each router runs the Dijkstra Shortest Path First (SPF) algorithm. Each router constructs an SPF tree using the LSDB, and installs the routes into the OSPF database.

Figure 6-10 shows the OSPF neighbor adjacency process.



**Figure 6-10** Process for Forming OSPF Neighbor Adjacencies

Example 6-1 shows each of the steps performed when an adjacency forms from a debug perspective.

### Example 6-1 OSPF Adjacency Debug Output

[Click here to view code image](#)

```
R1#debug ip ospf adj
```

```
OSPF adjacency events debugging is on
```

```
*21:10:01.735: OSPF: Build router LSA for area 0, router ID 192.168.1.1, seq
0x80000001, process 1
*21:10:09.203: OSPF: 2 Way Communication to 192.168.2.2 on GigabitEthernet0/0, state
2WAY
*21:10:39.855: OSPF: Rcv DBD from 192.168.2.2 on GigabitEthernet0/0 seq 0x1823 opt
0x52 flag 0x7 len 32 mtu 1500 state 2WAY
*21:10:39.855: OSPF: Nbr state is 2WAY
*21:10:41.235: OSPF: end of Wait on interface GigabitEthernet0/0
*21:10:41.235: OSPF: DR/BDR election on GigabitEthernet0/0
*21:10:41.235: OSPF: Elect BDR 192.168.2.2
*21:10:41.235: OSPF: Elect DR 192.168.2.2
*21:10:41.235: DR: 192.168.2.2 (Id) BDR: 192.168.2.2 (Id)
*21:10:41.235: OSPF: GigabitEthernet0/0 Nbr 192.168.2.2: Prepare dbase exchange
*21:10:41.235: OSPF: Send DBD to 192.168.2.2 on GigabitEthernet0/0 seq 0xFA9 opt
0x52 flag 0x7 len 32
*21:10:44.735: OSPF: Rcv DBD from 192.168.2.2 on GigabitEthernet0/0 seq 0x1823 opt
0x52 flag 0x7 len 32 mtu 1500 state EXSTART
*21:10:44.735: OSPF: NBR Negotiation Done. We are the SLAVE
*21:10:44.735: OSPF: GigabitEthernet0/0 Nbr 2.2.2.2: Summary list built, size 1
*21:10:44.735: OSPF: Send DBD to 192.168.2.2 on GigabitEthernet0/0 seq 0x1823 opt
0x52 flag 0x2 len 52
*21:10:44.743: OSPF: Rcv DBD from 192.168.2.2 on GigabitEthernet0/0 seq 0x1824 opt
0x52 flag 0x1 len 52 mtu 1500 state EXCHANGE
*21:10:44.743: OSPF: Exchange Done with 192.168.2.2 on GigabitEthernet0/0
*21:10:44.743: OSPF: Send LS REQ to 192.168.2.2 length 12 LSA count 1
*21:10:44.743: OSPF: Send DBD to 192.168.2.2 on GigabitEthernet0/0 seq 0x1824 opt
0x52 flag 0x0 len 32
*21:10:44.747: OSPF: Rcv LS UPD from 192.168.2.2 on GigabitEthernet0/0 length 76 LSA
count 1
*21:10:44.747: OSPF: Synchronized with 192.168.2.2 on GigabitEthernet0/0, state FULL
*21:10:44.747: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.2.2 on GigabitEthernet0/0 from
LOADING to FULL, Loading Done
```

## Basic OSPF Configuration

The configuration process for OSPF on IOS and IOS XR is different. On IOS nodes, most of the configuration resides under the OSPF process, but some OSPF options go directly on the interface configuration submode. IOS XR's configuration is hierarchical, and everything related to the OSPF configuration resides under the OSPF process. The command **router ospf process-id** defines and initializes the OSPF process for IOS and IOS XR.

### IOS network Statement

IOS routers can enable OSPF on an interface with a **network** statement or by associating an OSPF process under the interface.

The **network** statement identifies the interfaces that the OSPF process will use and the area that those interfaces participate in. The **network** statements match against the primary IPv4 address associated with an interface.

A common misconception is that the **network** statement advertises the networks into OSPF; in reality, though, the **network** statement is enabling OSPF on the interface. The interface is then advertised in OSPF through the LSA. The **network** statement uses a wildcard mask, which allows the configuration to be as specific or vague as necessary. The syntax for the **network**

statement is as follows:

**network** *ip-address wildcard-mask area area-id*

The concept is similar to the configuration of Enhanced Interior Gateway Routing Protocol (EIGRP), except that the OSPF area is specified. If the IP address for an interface matches two **network** statements with different areas, the most explicit **network** statement (longest match) preempts the other **network** statements for area allocation. On older versions of IOS, **network** statements process in the order they were entered versus using the longest match methodology used today. This behavior led to interfaces being associated to the wrong area.

The connected network for the OSPF-enabled interface is added in to the OSPF LSDB under the corresponding OSPF area in which the interface participates. Secondary connected networks are added to the LSDB only if the secondary IP address matches a **network** statement associated with the same area.

To help illustrate the concept, the following scenarios explain potential use cases of the **network** statement for a router with four interfaces. [Table 6-4](#) provides IP addresses and interfaces.

IOS Interface	IOS XR Interface	IP Address
Gigabit Ethernet 0/0	Gigabit Ethernet 0/0/0/0	10.0.0.10/24
Gigabit Ethernet 0/1	Gigabit Ethernet 0/0/0/1	10.0.10.10/24
Gigabit Ethernet 0/2	Gigabit Ethernet 0/0/0/2	192.0.0.10/24
Gigabit Ethernet 0/3	Gigabit Ethernet 0/0/0/3	192.10.0.10/24

**Table 6-4** Table of Sample Interfaces and IP Addresses

The configuration in [Example 6-2](#) enables OSPF for Area 0 only on the interfaces that explicitly match the IP addresses in [Table 6-4](#).

### **Example 6-2** OSPF Configuration with Explicit IP Addresses

[Click here to view code image](#)

```
router ospf 1
  network 10.0.0.10 0.0.0.0 area 0
  network 10.0.10.10 0.0.0.0 area 0
  network 192.0.0.10 0.0.0.0 area 0
  network 192.10.0.10 0.0.0.0 area 0
```

[Example 6-3](#) displays the OSPF configuration for Area 0 using **network** statements that match the subnets used in [Table 6-4](#). If we set the last octet of the IP address to 0 and changing the wildcard mask to 255, the **network** statements match all IP addresses within the /24 network.

### **Example 6-3** OSPF Configuration with Explicit Subnet

[Click here to view code image](#)

```
router ospf 1
  network 10.0.0.0 0.0.0.255 area 0
  network 10.0.10.0 0.0.0.255 area 0
  network 192.0.0.0 0.0.0.255 area 0
  network 192.10.0.0 0.0.0.255 area 0
```

**Example 6-4** displays the OSPF configuration for Area 0 using **network** statements for interfaces that are within the 10.0.0.0/8 or 192.0.0.0/8 network ranges and will result in OSPF enabled on all four interfaces, as in the previous two examples.

#### **Example 6-4** *OSPF Configuration with Large Subnet Ranges*

[Click here to view code image](#)

```
router ospf 1
  network 10.0.0.0 0.255.255.255 area 0
  network 192.0.0.0 0.255.255.255 area 0
```

**Example 6-5** displays the OSPF configuration for Area 0 to enable OSPF on all interfaces.

#### **Example 6-5** *OSPF Configuration for All Interfaces*

[Click here to view code image](#)

```
router ospf 1
  network 0.0.0.0 255.255.255.255
```

#### Note

For simplicity, this chapter focuses on OSPF operation from a single area, Area 0. The next chapter explains multi-area OSPF behavior in detail.

#### **IOS Interface Specific**

The second method for enabling OSPF on an interface for IOS is to configure it specifically on an interface with the command **ip ospf process-id area area-id [secondaries none]**. **Example 6-6** provides a sample configuration.

#### **Example 6-6** *OSPF Configuration on IOS for Specific Interface*

[Click here to view code image](#)

```
interface GigabitEthernet 0/0
  ip address 10.0.0.1 255.255.255.0
  ip ospf 1 area 0
```

This method provides explicit control for enabling OSPF; however, the configuration is not centralized, and increases complexity as the number of interfaces on your routers increase. Interface-specific settings will take precedence over the **network** statement with the assignment of the areas if a hybrid configuration exists on a router.

This method will also add secondary connected networks to the LSDB unless the **secondaries**

**none** option is used.

## IOS XR

IOS XR's protocol configuration is hierarchical and allows for a simpler configuration with nesting and inheritance. The benefits of the hierarchical configuration will be apparent as this book explains more of the OSPF features. Maintaining the entire OSPF configuration under one section simplifies troubleshooting of misconfigurations.

After you initialize the router process with the command **router ospf process-id**, OSPF areas are defined with the command **area area-id**. Interfaces are associated under the area with the command **interface interface-type interface-number**.

Table 6-4 displays IP addresses and interfaces for configuring OSPF on an IOS XR router.

Example 6-7 demonstrates the IOS XR configuration for Area 0.

### Example 6-7 Sample IOS-XR OSPF Configuration

[Click here to view code image](#)

```
router ospf 1
  area 0
    interface GigabitEthernet 0/0/0/0
    interface GigabitEthernet 0/0/0/1
    interface GigabitEthernet 0/0/0/2
    interface GigabitEthernet 0/0/0/3
```

### Passive Interfaces

Enabling an interface with OSPF is the quickest way to advertise the network segment to other OSPF routers. However, it might be easy for someone to plug in an unauthorized OSPF router on an OSPF-enabled network segment, and introduce false routes, thus causing havoc in the network. Making the network interface passive still adds the network segment into the LSDB, but prohibits the interface from forming OSPF adjacencies. A passive interface does not send out OSPF Hellos and will not process any received OSPF packets.

On IOS nodes, the command **passive interface interface-type interface-number** will make the interface passive, and the command **passive interface default** will make all interfaces passive. To allow for an interface to process OSPF packets, the command **no passive interface interface-type interface-number** is used.

On IOS XR nodes, the commands **passive enable** and **passive disable** are used. Placing the **passive enable** command in the global process will make all interfaces passive due to inheritance. Placing the command **passive disable** on the area or interface preempts the inheritance and allows the interface to process OSPF packets. Depending on where the passive commands are placed—globally, area, or interface—dictates the final configuration.

### Sample Topology and Configuration

Figure 6-11 shows a topology example of a basic OSPF configuration. All four routers will have loopback IP addresses to match their RIDs (192.168.1.1, 192.168.2.2, and so on).

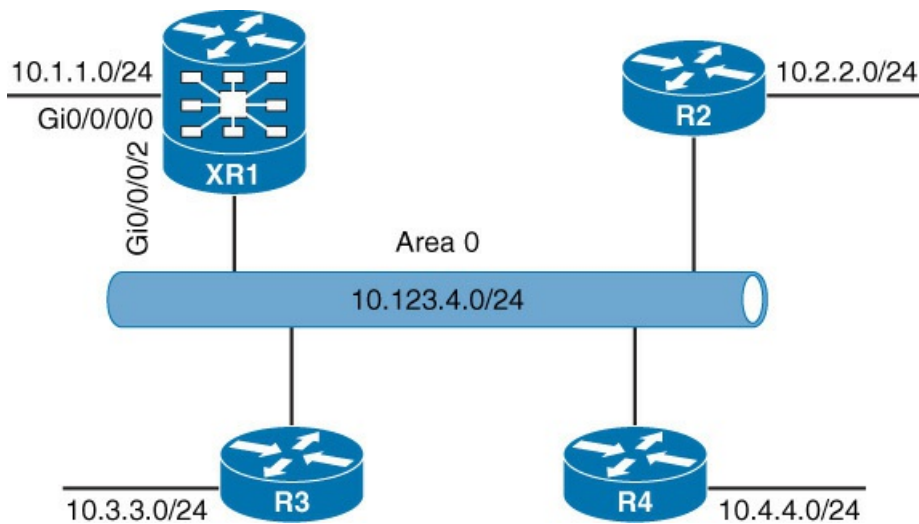


Figure 6-11 Sample OSPF Topology

On R2, specific network-based statements will be used, R3 will use interface-specific commands, and R4 will enable OSPF on all interfaces with one **network** statement. All interfaces except those connected to the 10.123.4.0/24 network will be set to passive. R2 will state explicit passive interfaces, and XR1, R3, and R4 will set the passive interface default setting.

Example 6-8 provides the sample configuration for all four routers.

### Example 6-8 OSPF Configurations for Topology Example

[Click here to view code image](#)

```

XR1
interface Loopback0
  ipv4 address 192.168.1.1 255.255.255.255
!
interface GigabitEthernet0/0/0/0
  ipv4 address 10.1.1.1 255.255.255.0
!
interface GigabitEthernet0/0/0/2
  ipv4 address 10.123.4.1 255.255.255.0
!
router ospf 1
  router-id 192.168.1.1
  passive enable
  area 0
    interface Loopback0
      !
    interface GigabitEthernet0/0/0/0
      !
    interface GigabitEthernet0/0/0/2
      passive disable
  
```



**R2**

```
interface Loopback0
  ip address 192.168.2.2 255.255.255.255
!
interface GigabitEthernet0/0
  ip address 10.123.4.2 255.255.255.0
!
interface GigabitEthernet0/1
  ip address 10.2.2.2 255.255.255.0
!
router ospf 1
  router-id 192.168.2.2
  passive-interface GigabitEthernet0/1
  passive-interface Loopback0
  network 10.2.2.2 0.0.0.0 area 0
  network 10.123.4.2 0.0.0.0 area 0
  network 192.168.2.2 0.0.0.0 area 0
```

**R3**

```
interface Loopback0
  ip address 192.168.3.3 255.255.255.255
  ip ospf 1 area 0
!
interface GigabitEthernet0/0
  ip address 10.123.4.3 255.255.255.0
  ip ospf 1 area 0
!
interface GigabitEthernet0/1
  ip address 10.3.3.3 255.255.255.0
  ip ospf 1 area 0
!
router ospf 1
  router-id 192.168.3.3
  passive-interface default
  no passive-interface GigabitEthernet0/0
```

**R4**

```
interface Loopback0
  ip address 192.168.4.4 255.255.255.255
!
interface GigabitEthernet0/0
  ip address 10.123.4.4 255.255.255.0
!
interface GigabitEthernet0/1
  ip address 10.4.4.4 255.255.255.0
!
router ospf 1
  router-id 192.168.4.4
  passive-interface default
  no passive-interface GigabitEthernet0/0
  network 0.0.0.0 255.255.255.255 area 0
```

## Confirmation of Interfaces

It is a good practice to verify that the correct interfaces are running OSPF after making changes to the OSPF configuration. The command **show ip ospf interface [brief]** displays the OSPF-enabled interfaces on IOS nodes, and the command **show ospf interface [brief]** provides identical output for IOS XR nodes.

[Example 6-9](#) displays output for the brief version of the commands.

### Example 6-9 OSPF Interface Output in Brief Format

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show ospf interface brief
```

```
* Indicates MADJ interface, (P) Indicates fast detect hold down state
```

```
Interfaces for OSPF 1
```

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Lo0	1	0	192.168.1.1/32	1	LOOP	0/0	
Gi0/0/0/0	1	0	10.1.1.1/24	1	DR	0/0	
Gi0/0/0/2	1	0	10.123.4.1/24	1	DROTH	2/3	

```
R2#show ip ospf interface brief
```

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Lo0	1	0	192.168.2.2/32	1	LOOP	0/0	
Gi0/1	1	0	10.2.2.2/24	1	DR	0/0	
Gi0/0	1	0	10.123.4.2/24	1	DROTH	3/3	

```
R3#show ip ospf interface brief
```

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Lo0	1	0	192.168.3.3/32	1	LOOP	0/0	
Gi0/1	1	0	10.3.3.3/24	1	DR	0/0	
Gi0/0	1	0	10.123.4.3/24	1	BDR	2/3	

```
R4#show ip ospf interface brief
```

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Lo0	1	0	192.168.4.4/32	1	LOOP	0/0	
Gi0/1	1	0	10.4.4.4/24	1	DR	0/0	
Gi0/0	1	0	10.123.4.4/24	1	DR	3/3	

[Table 6-5](#) provides an overview of the fields in the output from [Example 6-9](#).

Field	Description
Interface	Interfaces with OSPF enabled
PID	OSPF process ID associated with this interface
Area	The area that this interface is associated with
IP Address/Mask	The IP address and subnet mask for the interface
Cost	Cost is used to calculate a metric for a path by the SPF algorithm
State	Current interface state: DR, BDR, DROTHER, LOOP, or Down
Nbrs F	Number of neighbor OSPF routers for a segment that have are fully adjacent
Nbrs C	Number of neighbor OSPF routers for a segment that have been detected and are in a 2-WAY state

**Table 6-5** OSPF Interface Columns

### Verification of OSPF Neighbor Adjacencies

The command **show ip ospf neighbor [detail]** provides the OSPF neighbor table for IOS nodes, and the equivalent command **show ospf neighbor [detail]** provides similar output for IOS XR nodes. [Example 6-10](#) provides sample outputs on XR1 and R2-R4.

### Example 6-10 OSPF Neighbor Output

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show ospf neighbor
```

```
Neighbors for OSPF 1
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
192.168.4.4	1	FULL/DR	00:00:34	10.123.4.4	GigabitEthernet0/0/0/2
Neighbor is up for 00:54:54					
192.168.2.2	1	2WAY/DROTHER	00:00:32	10.123.4.2	GigabitEthernet0/0/0/2
Neighbor is up for 00:46:45					
192.168.3.3	1	FULL/BDR	00:00:35	10.123.4.3	GigabitEthernet0/0/0/2
Neighbor is up for 00:49:55					

```
Total neighbor count: 3
```

```
R2#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
192.168.1.1	1	2WAY/DROTHER	00:00:32	10.123.4.1	GigabitEthernet0/0
192.168.3.3	1	FULL/BDR	00:00:34	10.123.4.3	GigabitEthernet0/0
192.168.4.4	1	FULL/DR	00:00:33	10.123.4.4	GigabitEthernet0/0

```
R3#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
192.168.1.1	1	FULL/DROTHER	00:00:35	10.123.4.1	GigabitEthernet0/0
192.168.2.2	1	FULL/DROTHER	00:00:34	10.123.4.2	GigabitEthernet0/0
192.168.4.4	1	FULL/DR	00:00:31	10.123.4.4	GigabitEthernet0/0

```
R4#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
192.168.1.1	1	FULL/DROTHER	00:00:36	10.123.4.1	GigabitEthernet0/0
192.168.2.2	1	FULL/DROTHER	00:00:34	10.123.4.2	GigabitEthernet0/0
192.168.3.3	1	FULL/BDR	00:00:35	10.123.4.3	GigabitEthernet0/0

Table 6-6 provides a brief overview of the fields used in Example 6-10.

Field	Description
Neighbor ID	RID of neighboring router.
PRI	Priority for the neighbor's interface. This is used for DR/BDR elections.
State	The first field is the Neighbor state, as described in Table 6-3. The second field is the DR, BDR, or DROTHER role if the interface requires a DR. For non-DR network links, the second field will show a hyphen (-).
Dead Time	Dead time left until the router is declared unreachable.
Address	Primary IP address for the OSPF neighbor.
Interface	Local interface to which the OSPF neighbor is attached.
Uptime	IOS XR routers include the time that the adjacency has been up. IOS neighbor uptime is shown with the <b>detailed</b> option.

Table 6-6 OSPF Neighbor State Fields

#### Note

Looking at an OSPF Neighbor's state is helpful when troubleshooting adjacency issues. Depending on the LSDB size, the state may transition faster than you can see.

### Verification of OSPF Routes

The next step is to verify the OSPF routes installed in the IP routing table. OSPF routes that install into the Routing Information Base (RIB) are shown with the command **show ip route ospf** on IOS nodes and **show route ospf** on IOS XR nodes.

Example 6-11 provides sample output of the OSPF routing table for XR1 and R2. In the output, two sets of numbers are in the brackets, and look like [110/2]. The first number is the administrative distance (AD), which is 110 by default for OSPF, and the second number is the metric of the path used for that network. Output for R3 and R4 will be similar.

#### Note

The terms *path cost* and *path metric* are synonymous from OSPF's perspective.

### Example 6-11 OSPF Routes Installed in the RIB

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route ospf
```

```
O 10.2.2.0/24 [110/2] via 10.123.4.2, 00:21:49, GigabitEthernet0/0/0/2
O 10.3.3.0/24 [110/2] via 10.123.4.3, 00:25:00, GigabitEthernet0/0/0/2
O 10.4.4.0/24 [110/2] via 10.123.4.4, 00:25:44, GigabitEthernet0/0/0/2
O 192.168.2.2/32 [110/2] via 10.123.4.2, 00:21:49, GigabitEthernet0/0/0/2
O 192.168.3.3/32 [110/2] via 10.123.4.3, 00:25:00, GigabitEthernet0/0/0/2
O 192.168.4.4/32 [110/2] via 10.123.4.4, 00:25:44, GigabitEthernet0/0/0/2
```

```
R2#show ip route ospf
```

```
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       + - replicated route, % - next hop override
```

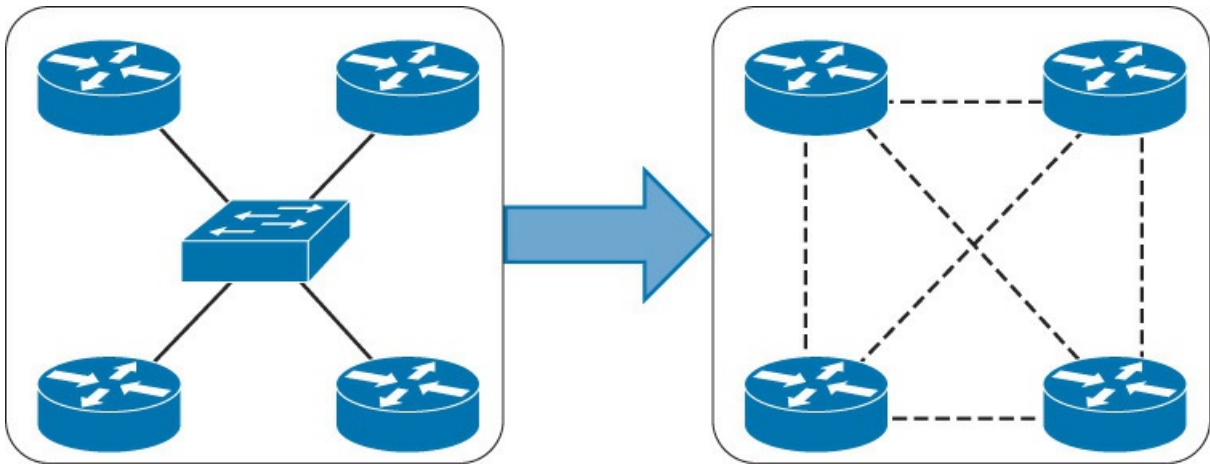
```
Gateway of last resort is not set
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
O 10.1.1.0/24 [110/2] via 10.123.4.1, 00:22:04, GigabitEthernet0/0
O 10.3.3.0/24 [110/2] via 10.123.4.3, 00:22:04, GigabitEthernet0/0
O 10.4.4.0/24 [110/2] via 10.123.4.4, 00:22:04, GigabitEthernet0/0
192.168.1.0/32 is subnetted, 4 subnets
O 192.168.1.1 [110/2] via 10.123.4.1, 00:22:04, GigabitEthernet0/0
192.168.1.0/32 is subnetted, 4 subnets
O 192.168.1.2 [110/2] via 10.123.4.1, 00:22:04, GigabitEthernet0/0
192.168.1.0/32 is subnetted, 4 subnets
O 192.168.1.3 [110/2] via 10.123.4.1, 00:22:04, GigabitEthernet0/0
192.168.1.0/32 is subnetted, 4 subnets
O 192.168.1.4 [110/2] via 10.123.4.1, 00:22:04, GigabitEthernet0/0
```

## DESIGNATED ROUTER AND BACKUP DESIGNATED ROUTER

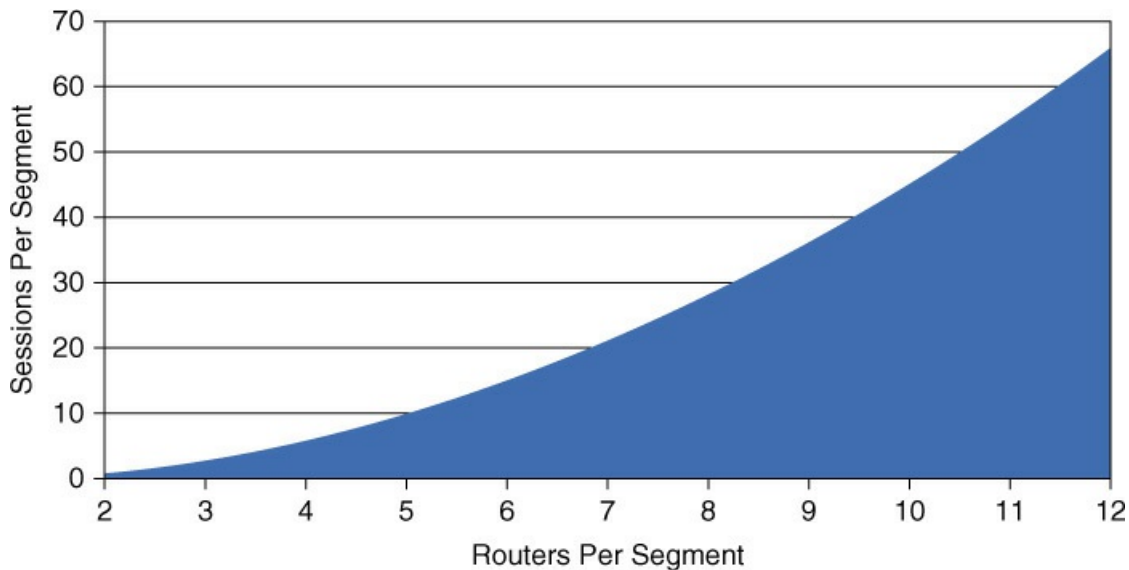
Multi-access networks such as Ethernet (LANs) and Frame Relay allow more than two routers to exist on a network segment. This could cause scalability problems with OSPF as the number of routers on a segment increase. Additional routers will flood more LSAs on the segment, and OSPF traffic becomes excessive as OSPF neighbor adjacencies increase. If four routers share the same multi-access network, six OSPF adjacencies would form, along with six occurrences of database flooding on a network. [Figure 6-12](#) shows a simple four-router physical topology and the adjacencies established.



**Figure 6-12** Multi-Access Physical Topology Versus Logical Topology

Using the number of edge's formula in complete graph  $n(n - 1) / 2$ , where  $n$  represents the number of routers, if five routers were present on a segment, ten OSPF adjacencies would exist for that segment. Continuing the logic, adding 1 additional router would makes 15 OSPF adjacencies on a network segment. Having this many adjacencies per segment consumes more bandwidth, more CPU processing, and more memory to maintain each of the neighbor states.

**Figure 6-13** illustrates the exponential rate of OSPF adjacencies needed as routers on a network segment increase.



**Figure 6-13** Exponential LSA Sessions per Routers on the Same Segment

Multi-access networks overcome this inefficiency by using a DR. DRs reduce the number of OSPF adjacencies on a multi-access network segment because routers only form a full OSPF adjacency with the DR and not each other. The DR is then responsible for flooding the update to all OSPF routers on that segment as updates occur. **Figure 6-14** demonstrates how this simplifies a four-router topology using only three neighbor adjacencies.

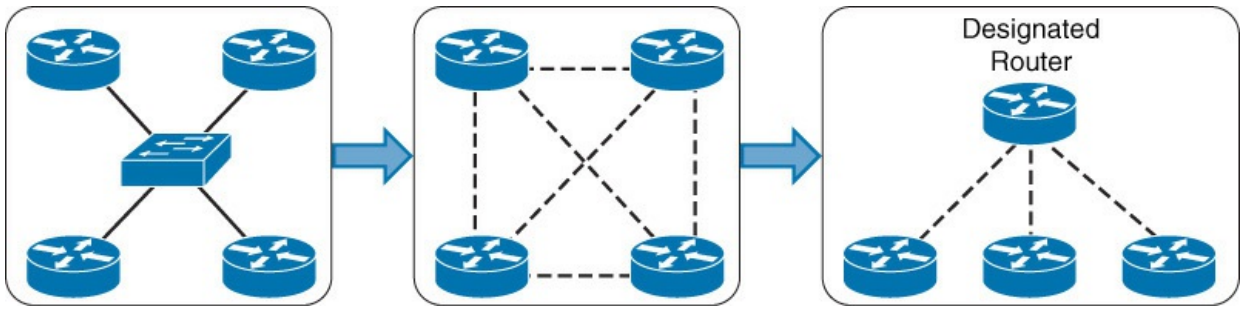


Figure 6-14 OSPF DR Concept

If the DR were to fail, OSPF would need to form new adjacencies invoking all new LSAs, and could potentially cause a temporary loss of routes. In the event of DR failure, a BDR will become the new DR; then an election occurs to replace the BDR. To minimize transition time, the BDR also forms a full OSPF adjacency with all OSPF routers on that segment.

The DR/BDR process distributes LSAs in the following manner:

**Step 1.** All OSPF routers (DR, BDR, and DROther) on a segment form a full OSPF adjacency with the DR and BDR.

**Step 2.** As an OSPF router learns of a new route, it sends the updated LSA to the AllDRouters (224.0.0.6) address, which only the DR and BDR receive and process, as illustrated in Step 1 of Figure 6-15.

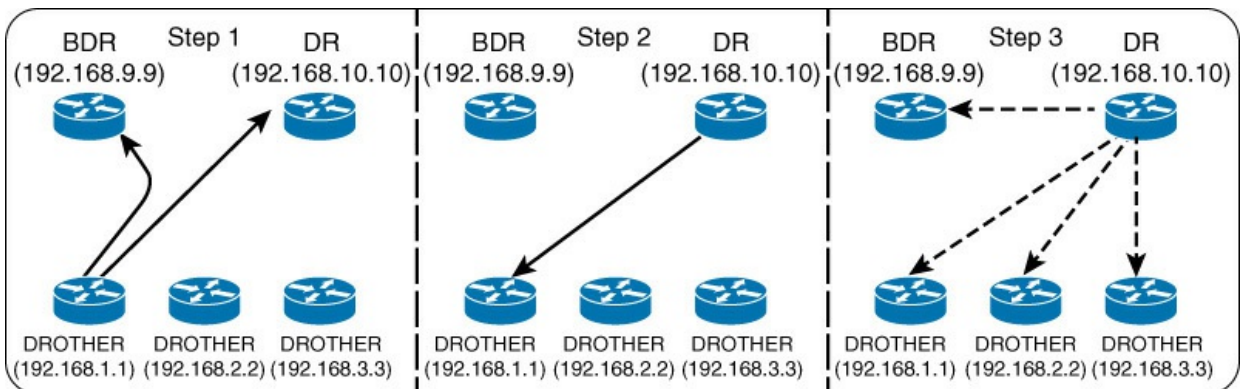


Figure 6-15 Network Prefix Advertisement with DR Segments

**Step 3.** The DR then sends a unicast acknowledgement to the router that send the initial LSA update, as illustrated in Step 2 of Figure 6-15.

**Step 4.** The DR then floods the LSA to all the routers on the segment via the AllSPFRouters (224.0.0.5) address, as shown in Step 3 of Figure 6-15.

#### Designated Router Elections

The DR/BDR election occurs during the last phase of the 2-Way neighbor state, and just before the ExStart state. When a router enters the 2-Way state, it has already received a Hello from the neighbor. If the Hello packet includes a RID other than 0.0.0.0 for the DR or BDR, the new router assumes that the current routers are the actual DR and BDR.

Note

OSPF DRs and BDRs cannot be preempted after the DR/BDR election.

Any router with the OSPF priority of 1–255 on the interface will attempt to become the DR. Setting the OSPF priority to 0 on an interface prohibits the router from attempting DR or BDR election for that segment. By default, all OSPF interfaces use a priority of 1. The routers will place their RID and OSPF priority in their OSPF Hellos for that segment.

Routers will then receive and examine OSPF Hellos from neighboring routers. If a router identifies itself as more favorable router than the OSPF Hellos it receives, it will continue to send out Hellos with its RID and priority listed. If the Hello received is more favorable, the router will update its OSPF Hello packet to use the more preferable RID in the DR field.

OSPF deems a router more preferable if the priority for the interface is the highest for that segment. If the OSPF priority is the same, the higher RID is more favorable.

Note

Technically per RFC 2328, the BDR is elected first to ensure an orderly transition of the BDR to DR role during DR failure. Because a DR does not exist on a new segment, the BDR becomes the DR, and the election for BDR occurs a second time.

Once all the routers have agreed on the same DR, all routers for that segment will become adjacent with the DR. Then the election for the BDR will take place. The election follows the same logic for the DR election, except that the DR will not add its RID to the BDR field of the Hello packet.

The DROther is a router on the DR-enabled segment that is not the DR or the BDR; it is simply the other router.

Note

To ensure that all routers on segment have fully initialized, OSPF will initiate a *wait timer* when OSPF Hello packets do not contain a DR/BDR router for a segment. The default value for the Wait timer is the dead interval timer. Once the wait timer has expired, a router will participate in the DR election. The wait timer starts when the OSPF first starts on an interface; so that a router can still elect itself as the DR for a segment without other OSPF routers, it will wait until the wait timer expires.

The easiest way to determine the interface role is by viewing the OSPF interfaces state, as shown in [Example 6-12](#). Notice that R4 is the DR and R3 is the BDR for the 10.123.4.0/24 network segment. XR1 and R2 are DROther.

### **Example 6-12** *OSPF Interface State*

[Click here to view code image](#)



```
RP/0/0/CPU0:XR1#show ospf interface brief
```

\* Indicates MADJ interface, (P) Indicates fast detect hold down state

Interfaces for OSPF 1

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Lo0	1	0	192.168.1.1/32	1	LOOP	0/0	
Gi0/0/0/0	1	0	10.1.1.1/24	1	DR	0/0	
Gi0/0/0/2	1	0	10.123.4.1/24	1	DROTH	2/3	

```
R2#show ip ospf interface brief
```

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Lo0	1	0	192.168.2.2/32	1	LOOP	0/0	
Gi0/1	1	0	10.2.2.2/24	1	DR	0/0	
Gi0/0	1	0	10.123.4.2/24	1	DROTH	2/3	

```
R3#show ip ospf interface brief
```

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Lo0	1	0	192.168.3.3/32	1	LOOP	0/0	
Gi0/1	1	0	10.3.3.3/24	1	DR	0/0	
Gi0/0	1	0	10.123.4.3/24	1	BDR	3/3	

```
R4#show ip ospf int br
```

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Lo0	1	0	192.168.4.4/32	1	LOOP	0/0	
Gi0/1	1	0	10.4.4.4/24	1	DR	0/0	
Gi0/0	1	0	10.123.4.4/24	1	DR	3/3	

The *neighbors full adjacency* field reflects the number of routers that have become adjacent on that network segment; whereas the *neighbors count* field is the number of other OSPF routers on that segment. The first assumption is that all routers will become adjacent with each other, but that defeats the purpose of using a DR. Only the DR and BDR become adjacent with routers on a network segment.

### DR and BDR Placement

In [Example 6-12](#), R4 won the DR election, and R3 won the BDR election. This was because all the OSPF routers had the same OSPF priority, so the next decision point uses the higher RID. The RIDs matched the Loopback 0 interface IP addresses, and R4's loopback address is the highest on that segment with R3's being the second highest.

Modifying a router's RID for DR placement is a bad design strategy, and a better technique involves modifying the interface priority to a higher value than the existing DR. In our current topology, the DR role for the segment (10.123.4.0/24) requires that the priority changes to a higher value than 1 (the existing DR's priority) on the desired node. Remember that OSPF does not preempt the DR or BDR roles and may require restarting the OSPF process on the current DR/BDR for the changes to take effect.

The priority can be set manually under the interface configuration submode with the command

**ip ospf priority 0-255** for IOS nodes or with the command **priority 0-255** under the global OSPF process, area, or per specific interface for IOS XR nodes.

Setting an interface priority to 0 removes that interface from the DR/BDR election immediately. Raising the priority above the default value (1) makes that interface more favorable over interfaces with the default value.

Figure 6-16 provides a topology example to illustrate modification of DR/BDR placement in a network segment. R4 should never become the DR/BDR for the 10.123.4.0/24 segment, and XR1 should always become the DR for the 10.123.4.0/24 segment.

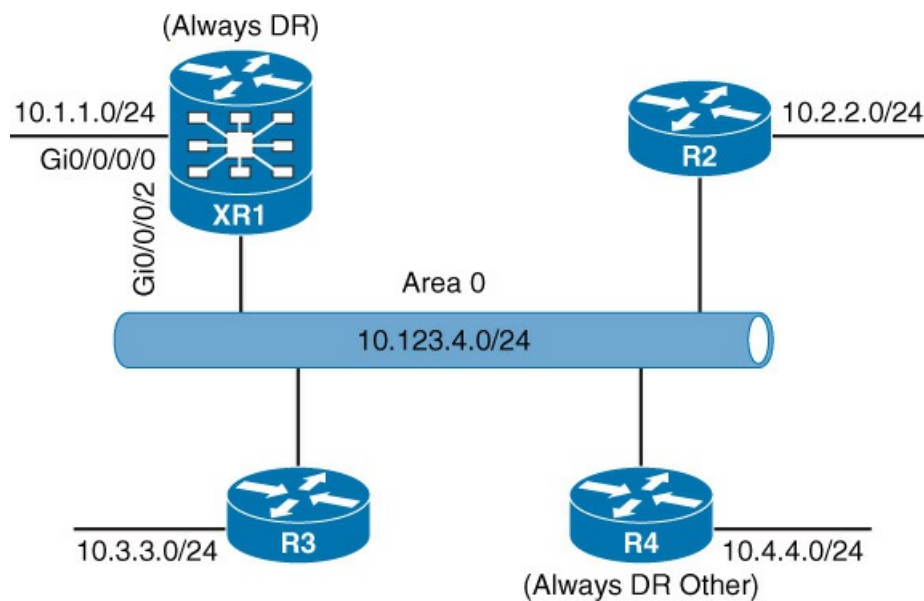


Figure 6-16 OSPF Topology with DR/BDR Manipulation

To prevent R4 from entering into the DR/BDR election, the OSPF priority changes to 0. XR1's interface priority will change to a value higher than 1 to ensure that it always wins the DR election.

Example 6-13 provides the relevant configuration for XR1 and R4. No configuration changes have occurred on R2 and R3.

### Example 6-13 OSPF Configuration with DR Manipulation

[Click here to view code image](#)

**XR1**

```
router ospf 1
  router-id 192.168.1.1
  passive enable
  area 0
    interface Loopback0
    !
    interface GigabitEthernet0/0/0/0
    !
    interface GigabitEthernet0/0/0/2
      passive disable
      priority 100
```

**R4**

```
interface GigabitEthernet0/0
  ip ospf priority 0
  ip address 10.123.4.4 255.255.255.0
!
router ospf 1
  router-id 192.168.4.4
  passive-interface default
  no passive-interface GigabitEthernet0/0
  network 0.0.0.0 255.255.255.255 area 0
```

**Example 6-14** provides verification of the change. XR1 is the DR with a priority of 100, and XR4 participates as the DROther role because it has a priority of 0.

**Example 6-14** *Verification of DR Manipulation*

[Click here to view code image](#)

```
R2#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
192.168.1.1	100	FULL/DR	00:00:38	10.10.10.1	GigabitEthernet0/0
192.168.3.3	1	EXSTART/BDR	00:00:39	10.10.10.3	GigabitEthernet0/0
192.168.4.4	0	2WAY/DROTHER	00:00:33	10.10.10.4	GigabitEthernet0/0

A key concept involving IOS XR is the hierarchical configuration and inheritance. **Example 6-15** demonstrates how all OSPF interfaces in Area 0 will have a priority of 200.

**Example 6-15** *Setting Interface Priority for an Entire Area*

[Click here to view code image](#)

**XR1**

```
router ospf 1
  router-id 192.168.1.1
  passive enable
  area 0
    priority 200
    interface Loopback0
    !
    interface GigabitEthernet0/0/0/0
    !
    interface GigabitEthernet0/0/0/2
      passive disable
```

**Example 6-16** expands on the hierarchical configuration and demonstrates how a lower-level setting can preempt an inherited setting. This can save a lot of time on a router that has hundreds of interfaces. Assume that the global OSPF priority should be set to 200 for all interfaces and 150 for the Gigabit Ethernet 0/0/0/0 interface.

### **Example 6-16** Area Interface Priority with Interface Preemption

[Click here to view code image](#)

**XR1**

```
router ospf 1
  router-id 192.168.1.1
  priority 200

  area 0
    interface Loopback0
    !
    interface GigabitEthernet0/0/0/0
      priority 150
    !
    interface GigabitEthernet0/0/0/2
```

**Example 6-17** provides verification that the area priority is in effect, while the interface priority preempted the value set at Area 0.

### **Example 6-17** Verification of Priority Settings

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show ospf interface | i "line|Priority"
```

```
Loopback0 is up, line protocol is up
GigabitEthernet0/0/0/0 is up, line protocol is up
  Transmit Delay is 1 sec, State DR, Priority 150, MTU 1500, MaxPktSz 1500
GigabitEthernet0/0/0/2 is up, line protocol is up
  Transmit Delay is 1 sec, State DROTHER, Priority 200, MTU 1500, MaxPktSz 1500
```

## FAILURE DETECTION

A secondary function of the OSPF Hello packets is to ensure that adjacent OSPF neighbors are still healthy and available. OSPF sends Hello packets at set intervals called the *hello timer*. OSPF uses a second timer called the *OSPF dead interval timer*, which defaults to four times the hello timer. Upon receipt of the Hello packet from a neighboring router, the OSPF dead timer resets to the initial value and then starts to decrement again.

If a router does not receive a Hello before the OSPF dead interval timer reaches 0, the neighbor state is changed to down. The OSPF router will immediately send out the appropriate LSA reflecting the topology change and the SPF algorithm processes on all routers within the area.

### Hello Timer

The default OSPF hello timer interval varies based on the OSPF network type. OSPF allows modification to the hello timer interval with values between 1 and 65,535 seconds. Changing the hello timer interval modifies the default dead interval, too. IOS uses the command **ip ospf hello-interval 1-65535** under the interface to modify the OSPF hello interval timer, and the equivalent command **hello-interval 1-65535** is used on IOS XR nodes within the OSPF global process, area, or interface.

### Dead Interval Timer

The dead interval timer can be changed to a value between 1-65535 seconds. The OSPF dead interval timer can be changed with the **ip ospf dead-interval 1-65535** under the interface configuration submode on IOS nodes. IOS XR nodes use the equivalent command **dead-interval 1-65535** on the interface, area, or within the OSPF global process.

#### Note

Always make sure that the dead interval timer is greater than the hello timer to ensure that the dead interval timer does not reach 0 before in between Hello packets.

## Verifying OSPF Timers

The timers for an OSPF interfaces are shown with the command **show ip ospf interface** for IOS nodes and with the command **show ospf interface** on IOS XR nodes. [Example 6-18](#) provides output for these commands.

### Example 6-18 OSPF Neighbor Output

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show ospf interface | i "Timer|line"
Loopback0 is up, line protocol is up
GigabitEthernet0/0/0/2 is up, line protocol is up
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5

R2#show ip ospf interface | i Timer|line
Loopback0 is up, line protocol is up
Ethernet0/0 is up, line protocol is up
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
```

#### Note

Hello and dead interval timers must match for OSPF Neighbors to become adjacent.

## OSPF Fast Packet Hellos

To detect failures quickly, the hello and dead interval timers need to be smaller. OSPF supports subsecond hello intervals, which sets the dead interval to 1 second. The OSPF hello timer can be changed with the command **ip ospf dead-interval minimal hello-multiplier 3-20** under the interface on IOS nodes. The **hello-multiplier** value is set to the number of hello packets OSPF sends in a 1-second interval.

[Example 6-19](#) provides a configuration with OSPF sending Hellos every 250 ms.

### Example 6-19 OSPF Fast Hello Configuration

[Click here to view code image](#)

```
R2
interface GigabitEthernet0/0
 ip address 10.12.1.2 255.255.255.0
 ip ospf dead-interval minimal hello-multiplier 4
```

[Example 6-20](#) shows that OSPF sends Hellos every 250 ms and a dead interval of 1 second.

### Example 6-20 OSPF Neighbor Output

[Click here to view code image](#)

```
R2#show ip ospf interface | i line|Timer
Loopback0 is up, line protocol is up
GigabitEthernet0/0 is up, line protocol is up
Timer intervals configured, Hello 250 msec, Dead 1, Wait 1, Retransmit 5
```

#### Note

IOS XR does not support OSPF fast-packet Hellos. IOS and IOS XR support other technologies that are less likely to provide false positives and are less resource-intensive. Technologies such as bidirectional forwarding detection (BFD) provide fast convergence and are more preferred and are covered in [Chapter 21, "High Availability."](#)

## OSPF NETWORK TYPES

Different media can provide different characteristics or might limit the number of nodes allowed on a segment. Frame Relay and Ethernet are a common multi-access media, and because they support more than two nodes on a network segment, the need for a DR exists. Other network circuits, such as serial links (HDLC or PPP encapsulation), do not require a DR and would just waste router CPU cycles. Cisco's implementation of OSPF takes into account the various mediums and provides five OSPF network types.

The default OSPF network type is set based on the media used for the connection and can be changed independently of the actual media type used. The following sections describe the various OSPF network types.

### Broadcast

Broadcast media such as Ethernet are better defined as broadcast multi-access to distinguish them from non-broadcast multi-access (NBMA) networks. Broadcast networks are multi-access in that they are capable of connecting more than two devices, and broadcasts sent out of one interface are capable of reaching all interfaces attached to that segment.

The OSPF network type is set to broadcast by default for Ethernet interfaces. A DR is required for

this OSPF network types because of the possibility that multiple nodes can exist on a segment and LSA flooding needs to be controlled. The hello timer defaults to 10 seconds, as defined in RFC 2328.

The IOS interface parameter command **ip ospf network broadcast** sets an interface as an OSPF broadcast network type. IOS XR uses the equivalent command **network broadcast** to set the interface as an OSPF broadcast network type at the global, area, or specific interface.

### Non-Broadcast

Frame Relay, ATM, and X.25 are considered non-broadcast multi-access (NBMA) in that they can connect more than two devices, and broadcasts sent out of one interface might not always be capable of reaching all the interfaces attached to the segment. Dynamic virtual circuits may provide connectivity, but the topology may not be a full mesh and might provide only a hub-and-spoke topology.

Frame Relay interfaces set the OSPF network type to non-broadcast by default. (Frame Relay subinterfaces use a different OSPF network type.) The Hello protocol interval takes 30 seconds on this OSPF network type. Multiple routers can exist on a segment, so the DR functionality is used. Neighbors are statically defined with the **neighbor ip-address** command because multicast and broadcast functionality do not exist on this type of circuit. Configuring a static neighbor will send OSPF Hellos using unicast.

The IOS interface parameter command **ip ospf network non-broadcast** sets an interface as an OSPF non-broadcast network type. IOS XR uses the equivalent command **network non-broadcast** to set the interface as an OSPF non-broadcast network type at the global, area, or specific interface.

Figure 6-17 demonstrates a Frame Relay topology.

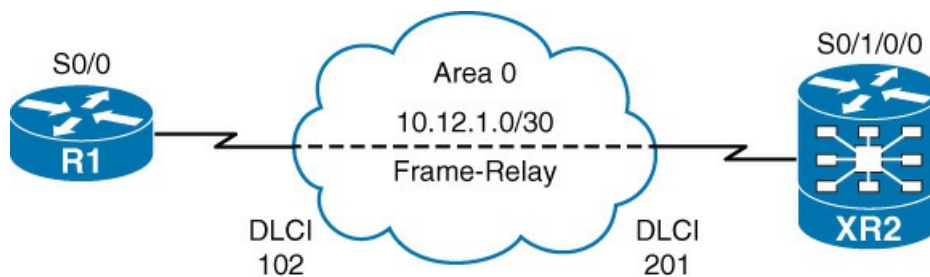


Figure 6-17 OSPF Topology Using Frame Relay

Example 6-21 provides the OSPF configuration over a Frame Relay interface. Notice that the static neighbor configuration is required when OSPF packets cannot be received via broadcast (multicast) discovery.

### Example 6-21 OSPF Configuration for Frame Relay Interfaces

[Click here to view code image](#)

**R1**

```
interface Serial 0/0
  ip address 10.12.1.1 255.255.255.252
  encapsulation frame-relay
  no frame-relay inverse-arp
  frame-relay map ip address 10.12.1.2 102
!
router ospf 1
  router-id 192.168.1.1
  neighbor 10.12.1.2
  network 0.0.0.0 255.255.255.255 area 0
```

**XR2**

```
interface Serial 0/1/0/0
  encapsulation frame-relay
  ipv4 address 10.12.1.2 255.255.255.252
  pvc 201
!
router ospf 1
  router-id 192.168.2.2
  area 0
  interface Loopback0
  !
  interface Serial 0/1/0/0
  neighbor 10.12.1.1
```

**Example 6-22** shows verification of the OSPF network type by filtering the results with the *Network Type* keywords. Reviewing the output confirms that the interfaces operate as non-broadcast.

**Example 6-22 Verification of Non-Broadcast OSPF Interfaces**

[Click here to view code image](#)

```
R1#show ip ospf interface Serial 0/0 | include Type
  Process ID 1, Router ID 192.168.1.1, Network Type NON_BROADCAST, Cost: 64

RP/0/0/CPU0:XR2#show ospf interface s0/1/0/0 | include Type
  Process ID 1, Router ID 192.168.2.2, Network Type NON_BROADCAST, Cost: 64
```

**Example 6-23** shows both routers have formed an adjacency, a DR exists, and the dead interval timer has increased 120 seconds corresponding to the 30-second hello interval. R1 is the DR, and XR2 is the BDR.

**Example 6-23 Verification of OSPF Non-Broadcast Neighbors**

[Click here to view code image](#)



```
R1#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
192.168.2.2	1	FULL/DR	00:01:47	10.12.1.2	Serial0/0

```
RP/0/0/CPU0:XR2#show ospf neighbor
```

\* Indicates MADJ interface

Neighbors for OSPF 1

Neighbor ID	Pri	State	Dead Time	Address	Interface
192.168.1.1	1	FULL/BDR	00:01:34	10.12.1.1	Serial0/1/0/0

Total neighbor count: 1

### Point-to-Point Networks

A network circuit that allows only two devices to communicate is considered a *point-to-point* (P2P) network. Because of the nature of the medium, point-to-point networks do not use Address Resolution Protocol (ARP), and broadcast traffic does not become limiting factors.

The OSPF network type is set to point-to-point by default for serial interfaces (HDLC or PPP encapsulation), Generic routing encapsulation (GRE) tunnels, and P2P Frame Relay subinterfaces. Only two nodes can exist on this type of network media, so OSPF does not waste CPU cycles on DR functionality. The hello timer is set to 10 seconds on OSPF P2P network types.

The IOS interface parameter command **ip ospf network point-to-point** sets an interface as an OSPF P2P network type. IOS XR uses the equivalent command **network point-to-point** to set the interface network type at the global, area, or specific interface.

Figure 6-18 shows a serial connection between R1 and XR2.

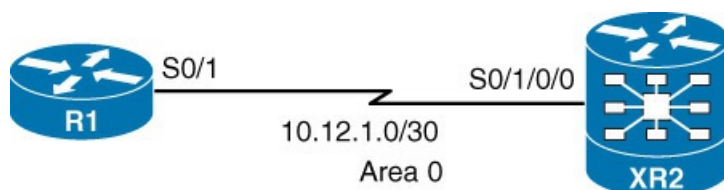


Figure 6-18 OSPF Topology with Serial Interfaces

Example 6-24 demonstrates the OSPF configuration of R1 and XR2. Notice that the network type was not set in the configuration and will automatically be set to P2P.

### Example 6-24 Point-to-Point OSPF Configuration

[Click here to view code image](#)

**R1**

```
router ospf 1
  router-id 192.168.1.1
  network 0.0.0.0 255.255.255.255 area 0
```

**XR2**

```
router ospf 1
  router-id 192.168.2.2
  area 0
  interface Loopback0
  !
  interface Serial 0/1/0/0
```

**Example 6-25** verifies that the OSPF network type is set to P2P.

**Example 6-25 Verification of OSPF P2P Interfaces**

[Click here to view code image](#)

```
R1#show ip ospf interface s0/0 | include Type
  Process ID 1, Router ID 192.168.1.1, Network Type POINT_TO_POINT, Cost: 64
```

```
RP/0/0/CPU0:XR2#show ospf interface s0/1/0/0 | include Type
  Process ID 1, Router ID 192.168.2.2, Network Type POINT_TO_POINT, Cost: 64
```

**Example 6-26** shows that P2P OSPF network types do not use a DR. Notice the hyphen (-) in the State field.

**Example 6-26 Verification of OSPF Neighbors on P2P Interfaces**

[Click here to view code image](#)

```
R1#show ip ospf neighbor

Neighbor ID      Pri   State           Dead Time   Address      Interface
192.168.2.2      0    FULL/-          00:00:36   10.12.1.2   Serial0/0

RP/0/0/CPU0:XR2#show ospf neighbor

* Indicates MADJ interface

Neighbors for OSPF 1

Neighbor ID      Pri   State           Dead Time   Address      Interface
192.168.1.1      1    FULL/-          00:00:33   10.12.1.1   Serial 0/1/0/0
  Neighbor is up for 00:01:14

Total neighbor count: 1
```

Note

Interfaces using an OSPF P2P network type form an OSPF adjacency quicker because the DR election is bypassed and there is no wait timer. Ethernet interfaces that are directly connected with only two OSPF speakers in the subnet should be changed to the OSPF P2P network type for forming adjacency faster and simplifying the SPF computation.

### Point-to-Multipoint Networks

The OSPF network type point-to-multipoint is not enabled by default for any medium. It requires manual configuration. A DR is not enabled for this OSPF network type, and the hello timer is set 30 seconds. A point-to-multipoint OSPF network type supports *hub-and-spoke* connectivity while using the same IP subnet and is commonly found in Frame Relay, Dynamic Multipoint VPN (DMVPN), and Layer 2 VPN (L2VPN) topologies.

Interfaces set for the OSPF point-to-multipoint network type add the interface's IP address into the OSPF LSDB as a /32 network. When advertising routes to OSPF peers on that interface, the next-hop address is set to the IP address of the interface even if the next-hop IP address resides on the same IP subnet.

The IOS interface parameter command **ip ospf network point-to-multipoint** sets an interface as an OSPF point-to-multipoint network type. The equivalent command for IOS XR is **network point-to-multipoint** to set the interface as an OSPF point-to-multipoint network type at the global, area, or specific interface.

Figure 6-19 provides a topology example with R1, R2, and R3 all using Frame Relay point-to-multipoint subinterfaces using the same subnet.

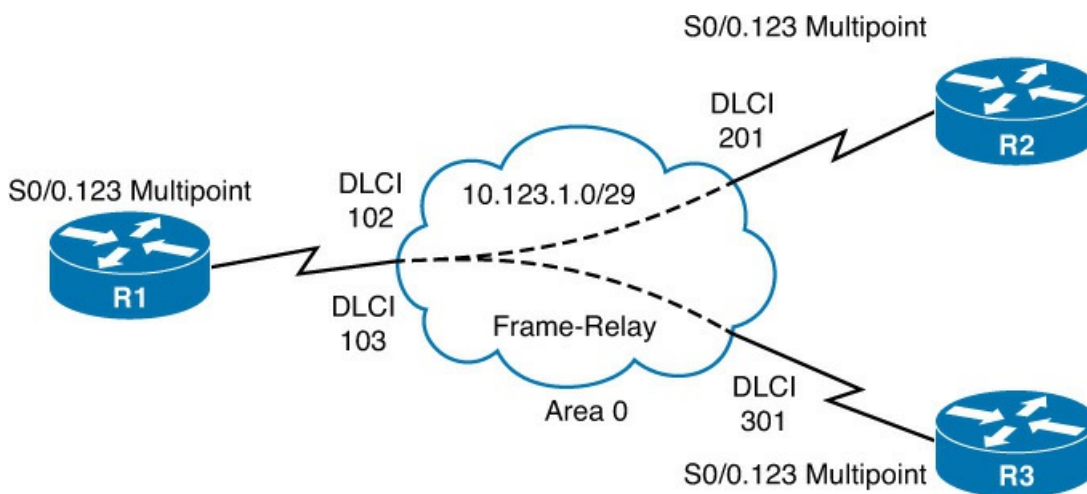


Figure 6-19 OSPF Topology with Frame Relay Multipoint Interfaces

Note

IOS XR does not support point-to-multipoint Frame Relay subinterfaces and was excluded from this example.

Example 6-27 demonstrates the relevant configuration for all three routers.

### Example 6-27 OSPF Point-to-Multipoint Configuration

[Click here to view code image](#)

**R1**

```
interface Serial 0/0
  encapsulation frame-relay
  no frame-relay inverse-arp
!
interface Serial 0/0.123 multipoint
  ip address 10.123.1.1 255.255.255.248
  frame-relay map ip 10.123.1.2 102 broadcast
  frame-relay map ip 10.123.1.3 103 broadcast
  ip ospf network point-to-multipoint
!
router ospf 1
  router-id 192.168.1.1
  network 0.0.0.0 255.255.255.255 area 0
```

**R2**

```
interface Serial 0/1/0
  encapsulation frame-relay
  no frame-relay inverse-arp
!
interface Serial 0/1/0/0.123 multipoint
  ip address 10.123.1.2 255.255.255.248
  frame-relay map ip 10.123.1.1 201 broadcast
  ip ospf network point-to-multipoint
!
router ospf 1
  router-id 192.168.2.2
  network 0.0.0.0 255.255.255.255 area 0
```

**R3**

```
interface Serial 0/0
  encapsulation frame-relay
  no frame-relay inverse-arp
!
interface Serial 0/0.123 multipoint
  ip address 10.123.1.3 255.255.255.248
  frame-relay map ip 10.123.1.1 301 broadcast
  ip ospf network point-to-multipoint
!
router ospf 1
  router-id 192.168.3.3
  network 0.0.0.0 255.255.255.255 area 0
```

**Example 6-28** verifies that the interfaces are the OSPF point-to-multipoint network type.

**Example 6-28** *Verification of OSPF Network Type Point-to-Multipoint*

**Click here to view code image**

```
R1#show ip ospf interface Serial 0/0.123 | include Type
  Process ID 1, Router ID 192.168.1.1, Network Type POINT_TO_MULTIPOINT, Cost: 64
```

```
R2#show ip ospf interface Serial 0/0.123 | include Type
Process ID 1, Router ID 192.168.2.2, Network Type POINT_TO_MULTIPOINT, Cost: 64
```

```
R3#show ip ospf interface Serial 0/0.123 | include Type
Process ID 1, Router ID 192.168.3.3, Network Type POINT_TO_MULTIPOINT, Cost: 64
```

**Example 6-29** displays that OSPF does not use a DR for the OSPF point-to-multipoint network type. Notice that all three routers are on the same subnet, but R2 and R3 do not establish an adjacency with each other.

### **Example 6-29** OSPF Neighbor Adjacency on a Hub-and-Spoke Topology

[Click here to view code image](#)

```
R1#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
192.168.3.3	0	FULL/ -	00:01:33	10.123.1.3	Serial0/0.123
192.168.2.2	0	FULL/ -	00:01:40	10.123.1.2	Serial0/0.123

```
R2#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
192.168.1.1	0	FULL/ -	00:01:49	10.123.1.1	Serial0/0.123

```
R3#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
192.168.1.1	0	FULL/ -	00:01:46	10.123.1.1	Serial0/0.123

**Example 6-30** shows that all the Serial 0/0.123 interfaces are advertised into OSPF as a /32 network and that the next-hop address is set (by R1) when advertised to the spokes nodes.

### **Example 6-30** OSPF Point-to-Multipoint Routing Tables

[Click here to view code image](#)

```
R1#show ip route ospf | begin Gateway
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
O    10.123.1.2/32 [110/64] via 10.123.1.2, 00:07:32, Serial0/0.123
O    10.123.1.3/32 [110/64] via 10.123.1.3, 00:03:58, Serial0/0.123
192.168.2.0/32 is subnetted, 1 subnets
O    192.168.2.2 [110/65] via 10.123.1.2, 00:07:32, Serial0/0.123
192.168.3.0/32 is subnetted, 1 subnets
O    192.168.3.3 [110/65] via 10.123.1.3, 00:03:58, Serial0/0.123
```

```
R2#show ip route ospf | begin Gateway
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
O   10.123.1.1/32 [110/64] via 10.123.1.1, 00:07:17, Serial0/0.123
O   10.123.1.3/32 [110/128] via 10.123.1.1, 00:03:39, Serial0/0.123
192.168.1.0/32 is subnetted, 1 subnets
O   192.168.1.1 [110/65] via 10.123.1.1, 00:07:17, Serial0/0.123
192.168.3.0/32 is subnetted, 1 subnets
O   192.168.3.3 [110/129] via 10.123.1.1, 00:03:39, Serial0/0.123
```

```
R3#show ip route ospf | begin Gateway
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
O   10.123.1.1/32 [110/64] via 10.123.1.1, 00:04:27, Serial0/0.123
O   10.123.1.2/32 [110/128] via 10.123.1.1, 00:04:27, Serial0/0.123
192.168.1.0/32 is subnetted, 1 subnets
O   192.168.1.1 [110/65] via 10.123.1.1, 00:04:27, Serial0/0.123
192.168.2.0/32 is subnetted, 1 subnets
O   192.168.2.2 [110/129] via 10.123.1.1, 00:04:27, Serial0/0.123
```

#### Note

Hub-and-spoke topologies are not limited to Frame Relay. DMVPN and Multiprotocol Label Switching (MPLS) L2VPN technologies allow for hub-and-spoke topologies, too.

## Loopback Networks

The OSPF network type loopback is enabled by default for loopback interfaces and can be used only on loopback interfaces. The OSPF loopback network type states that the IP is always advertised with a /32 prefix length, even if IP address configured on the loopback interface does not have a /32 prefix length.

Figure 6-20 shows R1 and R2 both advertising their loopback 0 interfaces.

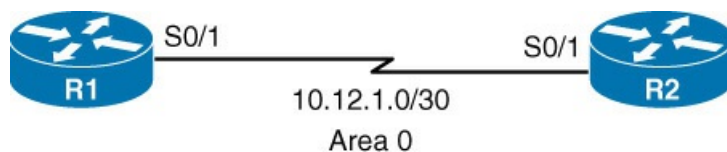


Figure 6-20 OSPF Topology for Loopback Network Type

Example 6-31 shows a sample configuration. Notice that R2 sets the OSPF network type to P2P.

### Example 6-31 OSPF Loopback Network Type

[Click here to view code image](#)

```
R1
interface Loopback0
  ip address 192.168.1.1 255.255.255.0
!
interface Serial 0/0
  ip address 10.12.1.1 255.255.255.252
!
router ospf 1
  router-id 192.168.1.1
  network 0.0.0.0 255.255.255.255 area 0
```

```
R2
interface Loopback0
  ip address 192.168.2.2 255.255.255.0
  ip ospf network point-to-point
!
interface Serial 0/0
  ip address 10.12.1.2 255.255.255.252
!
router ospf 1
  router-id 192.168.2.2
  network 0.0.0.0 255.255.255.255 area 0
```

[Example 6-32](#) verifies that the OSPF network type for R1 is loopback and R2 is different.

### **Example 6-32** *Display of OSPF Network Type for Loopback Interfaces*

[Click here to view code image](#)

```
R1#show ip ospf interface Loopback 0 | include Type
  Process ID 1, Router ID 192.168.1.1, Network Type LOOPBACK, Cost: 1

R2#show ip ospf interface Loopback 0 | include Type
  Process ID 1, Router ID 192.168.2.2, Network Type POINT_TO_POINT, Cost: 1
```

After examining [Example 6-33](#), notice that R1's loopback is a /32 and R2's loopback is a /24. Both loopbacks were configured with a /24 network; but because R1's Loo is OSPF network type of loopback, it is advertised as a /32.

### **Example 6-33** *OSPF Route Table for OSPF Loopback Network Types*

[Click here to view code image](#)

```
R1#show ip route ospf
! Output omitted for brevity

Gateway of last resort is not set

O          192.168.2.0/24 [110/65] via 10.12.1.2, 00:02:49, Serial0/0
```

```
R2#show ip route ospf
! Output omitted for brevity

Gateway of last resort is not set

    192.168.1.0/32 is subnetted, 1 subnets
O       192.168.1.1 [110/65] via 10.12.1.1, 00:37:15, Serial0/0
```

### Review of OSPF Network Types

OSPF chooses the OSPF network type based on the interface type. The command **ip ospf network {broadcast | non-broadcast | point-to-point | point-to-multipoint}** allows for IOS nodes to change the OSPF network type when placed under the interface. IOS XR uses the command **network {broadcast | non-broadcast | point-to-point | point-to-multipoint}** on the OSPF global process, area, or for a specific interface.

Table 6-7 summarizes the characteristics of the various OSPF network types.

Type	Description	DR	Timers
Broadcast	Default setting on OSPF-enabled Ethernet links.	Yes	Hello: 10 Wait: 40 Dead: 40
Non-broadcast	Default setting on enabled OSPF Frame Relay main interface or Frame Relay multipoint subinterfaces.	Yes	Hello: 30 Wait: 120 Dead: 120
Point-to-point	Default setting on enabled OSPF Frame Relay point-to-point subinterfaces.	No	Hello: 10 Wait: 40 Dead: 40
Point-to-multipoint	Not enabled by default on any interface type. The interface is advertised as a host route (/32) and sets the next-hop address to outbound interface. Primarily used for hub-and-spoke topologies.	No	Hello: 30 Wait: 120 Dead: 120
Loopback	Default setting on OSPF-enabled loopback interfaces. The interface is advertised as a host route (/32).	—	—

Table 6-7 OSPF Network Types

### OSPF Adjacency with Different OSPF Network Types

A common misconception is that the OSPF network types must match to form an adjacency and exchange routes. OSPF neighbor adjacency requires that the hello timers and usage of a DR must match.

This means that two routers can form an adjacency if one has a non-broadcast OSPF network type, and the other neighbor has a broadcast OSPF network type. Alternatively, a router can form an adjacency between P2P OSPF network type and point-to-multipoint OSPF network type. In both instances, the hello timers must be changed on at least one node for the timers to match.



Note

If you use a network type that uses a DR with a network type that does not use a DR, the adjacency may still form, but routes will not install correctly because each type uses a different type of LSA for route advertisement.

Figure 6-21 demonstrates this concept. R1's Serial 0/0.123 interface is a multipoint interface, and R2's and R3's Serial 0/0.123 interface are P2P interfaces. R1's hello timers have been set to match the 10-second interval used by the P2P OSPF network type, as shown here.

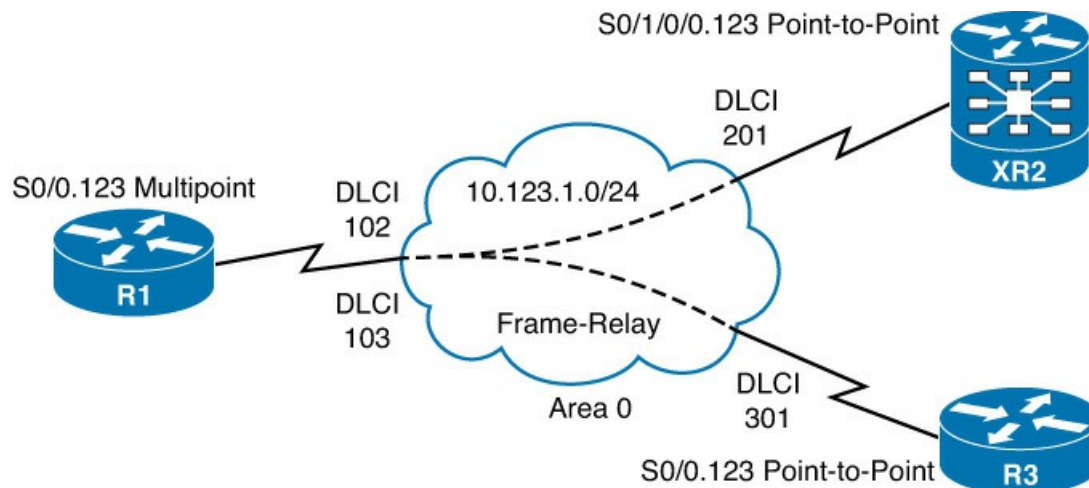


Figure 6-21 OSPF Topology with Dissimilar OSPF Network Types

Example 6-34 provides the sample configuration for the topology. Notice how the subinterfaces on XR2 and R3 are P2P and R1's is a multipoint interface. The P2P subinterfaces are automatically set to the OSPF P2P network type.

### Example 6-34 OSPF Configuration with Dissimilar OSPF Network Types

[Click here to view code image](#)

```
R1
interface Serial 0/0
  encapsulation frame-relay
  no frame-relay inverse-arp
!
interface Serial 0/0.123 multipoint
  ip address 10.123.1.1 255.255.255.248
  frame-relay map ip 10.123.1.3 103 broadcast
  frame-relay map ip 10.123.1.2 102 broadcast
  ip ospf hello-interval 10
  ip ospf network point-to-multipoint
!
router ospf 1
  router-id 192.168.1.1
  network 0.0.0.0 255.255.255.255 area 0
```

**XR2**

```
interface Serial 0/1/0/0
  encapsulation frame-relay

interface Serial 0/1/0/0.123 point-to-point
  ip address 10.123.1.2 255.255.255.248
  pvc 201
!
router ospf 1
  router-id 192.168.2.2
  area 0
    interface Loopback 0
    !
    Interface Serial 0/1/0/0/.201
```

**R3**

```
interface Serial 0/0
  encapsulation frame-relay
  no frame-relay inverse-arp
!
interface Serial 0/0.123 point-to-point
  ip address 10.123.1.3 255.255.255.248
  frame-relay interface-dlci 301
!
router ospf 1
  router-id 192.168.3.3
  network 0.0.0.0 255.255.255.255 area 0
```

**Example 6-35** provides verification that the OSPF network types do not match, but the OSPF timers do match, which allows the routers to form an adjacency.

**Example 6-35** *Verification of Dissimilar OSPF Network Types*

[Click here to view code image](#)

```
R1#show ip ospf interface Serial 0/0.123 | in Type|Hello
Process ID 1, Router ID 192.168.1.1, Network Type POINT_TO_MULTIPPOINT, Cost: 64
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
Hello due in 00:00:01
```

```
RP/0/0/CPU0:XR2#show ip ospf interface Serial 0/0.123 | in Type|Hello
Process ID 1, Router ID 192.168.2.2, Network Type POINT_TO_POINT, Cost: 64
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
Hello due in 00:00:05
```

```
R3#show ip ospf interface Serial 0/0.123 | in Type|Hello
Process ID 1, Router ID 192.168.3.3, Network Type POINT_TO_POINT, Cost: 64
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
Hello due in 00:00:03
```

Example 6-36 provides routing tables for all three routers; the Serial 0/0.123 interfaces on R2 and R3 are not advertised by OSPF, whereas R1's Serial 0/0.123 interface is advertised as a /32.

### Example 6-36 OSPF Routes for Dissimilar OSPF Network Types

[Click here to view code image](#)

```
R1#show ip route ospf
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
192.168.2.0/32 is subnetted, 1 subnets
O    192.168.2.2 [110/65] via 10.123.1.2, 00:23:20, Serial0/0.123
192.168.2.0/32 is subnetted, 1 subnets
O    192.168.3.3 [110/65] via 10.123.1.3, 00:19:46, Serial0/0.123
```

```
R2#show ip route ospf
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
O    10.123.1.1/32 [110/64] via 10.123.1.1, 00:04:44, Serial0/0.123
192.168.1.0/32 is subnetted, 1 subnets
O    192.168.1.1 [110/65] via 10.123.1.1, 00:04:44, Serial0/0.123
192.168.3.0/32 is subnetted, 1 subnets
O    192.168.3.3 [110/129] via 10.123.1.1, 00:04:34, Serial0/0.123
```

```
R3#show ip route ospf
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
O    10.123.1.1/32 [110/64] via 10.123.1.1, 00:04:52, Serial0/0.123
192.168.1.0/32 is subnetted, 1 subnets
O    192.168.1.1 [110/65] via 10.123.1.1, 00:04:52, Serial0/0.123
192.168.2.0/32 is subnetted, 1 subnets
O    192.168.2.2 [110/129] via 10.123.1.1, 00:04:52, Serial0/0.123
```

#### Note

Although scenarios like this are not common, the protocol supports adjacencies of different OSPF network types as long as the timers match and the usage of a DR is the same. Troubleshooting is simplified if the OSPF network types are identical for all neighbors.

## LINK COSTS

Interface cost is an essential component for Dijkstra's SPF calculation because the shortest path metric is based on the cumulative interface cost (that is, metric) from the router to the destination. OSPF assigns the OSPF link cost (that is, metric) for an interface using the formula in [Figure 6-22](#).

$$\text{Cost} = \frac{\text{Reference Bandwidth}}{\text{Interface Bandwidth}}$$

**Figure 6-22** OSPF Interface Cost Formula

The default reference bandwidth is 100 Mbps (108 bps). [Table 6-8](#) provides the OSPF cost for common network interface types using the default reference bandwidth.

Interface Type	OSPF Cost
T1	64
Ethernet	10
Fast Ethernet	1
Gigabit Ethernet	1
10-Gigabit Ethernet	1

**Table 6-8** OSPF Interface Costs Using Default Settings

Notice in [Table 6-8](#) that there is no differentiation in the link cost associated to a Fast Ethernet interface and a 10-Gigabit Ethernet interface. Changing the reference bandwidth to a higher value allows for a differentiation of cost between higher speed interfaces. Changing the value to too high of a value could cause issues because low bandwidth interfaces would not be distinguishable. The OSPF LSA metric field is 16-bits and the interface cost cannot exceed 65,535.

Under the OSPF process, the command **auto-cost reference-bandwidth** *bandwidth-in-mbps* changes the reference bandwidth.

Note

If the reference bandwidth is changed on one router, then the reference bandwidth should be changed on all OSPF routers to ensure SPF uses the same logic. NX-OS nodes use a default reference bandwidth of 40 Gbps.

The OSPF cost can be set manually under the interface using the command **ip ospf cost 1-65535** for IOS nodes. The cost can be set manually on IOS XR with the command **cost 1-65535** under the global OSPF process, area, or per specific interface.

Note

While the interface cost is limited to 65,535 because of LSA field limitations, the path metric can exceed a 16-bit value (65,535) because all the link metrics are calculated locally.

## AUTHENTICATION

An attacker can forge OSPF packets or gain physical access to the network. After they have been able to manipulate the routing table, they can send traffic down links that allow for traffic interception, create a denial-of-service attack, or perform some other malicious behavior.

OSPF supports two types authentication:

■ **Plain text:** Provides little security, as anyone with access to the link can see the password with a network sniffer

■ **MD5 cryptographic hash:** Uses a hash instead, so the password is never sent out the wire, and this technique is widely accepted as being the more secure mode

### IOS Support for OSPF Authentication

IOS nodes support enabling OSPF authentication on an interface-by-interface basis or all interfaces in an area. The password can be set only as an interface parameter and must be set for every interface. Missing an interface sets the default password of a null value.

Plaintext authentication is enabled for an OSPF area with the command **area area-id authentication**, and the interface parameter command **ip ospf authentication** sets plaintext authentication only on that interface. The plaintext password is configured with the interface parameter command **ip ospf authentication-key password**.

MD5 authentication is enabled for an OSPF area with the command **area area-id authentication message-digest**, and the interface parameter command **ip ospf authentication message-digest** sets MD5 authentication for that interface. The MD5 password is configured with the interface parameter command **ip ospf message-digest-key key-number md5 password**.

#### Note

MD5 authentication is a hash of the key number and password combined. If the keys do not match, the hash will differ between the nodes.

### IOS XR Support for OSPF Authentication

Enabling OSPF authentication on IOS XR is much simpler thanks to the hierarchical configuration. The authentication configuration can be placed globally, per specific area, or on a specific interface depending on the specific topology. Remember that an area setting takes priority over a global setting, and an interface setting takes priority over an area setting.

The command **authentication** is used to enable plaintext authentication, and the command **authentication message-digest** is used to enable MD5 authentication. Once authentication is enabled, the password must be configured. The command **authentication-key password** is used to set the plaintext password, and the command **message-digest-key key-number md5 password** is used to set MD5 password.

Figure 6-23 provides a simple topology to demonstrate the OSPF authentication configuration. XR2 and R3 will configure authentication using only OSPF area commands, and XR1 and R4 will use interface-specific commands. Area 12 and Area 34 will use MD5 authentication, and Area 0 will use plaintext authentication. The password for all areas will be CISCO.



**Figure 6-23** Authentication Topology

Example 6-37 provides the OSPF authentication configuration. All routers will use the password of CISCO.

### Example 6-37 OSPF Authentication Configuration

[Click here to view code image](#)

```

XR1
router ospf 1
 area 12
  interface Loopback0
  !
  interface GigabitEthernet0/0/0/0
  authentication message-digest
  message-digest-key 1 md5 CISCO
  
```

```

XR2
router ospf 1
 area 0
  authentication-key CISCO
  authentication
  interface Loopback0
  !
  interface GigabitEthernet0/0/0/2
  !
  !
 area 12
  authentication message-digest
  message-digest-key 1 md5 CISCO
  interface GigabitEthernet0/0/0/0
  
```

**R3**

```
interface GigabitEthernet0/0
 ip address 10.23.1.3 255.255.255.0
 ip ospf authentication-key CISCO
!
interface GigabitEthernet0/1
 ip address 10.34.1.3 255.255.255.0
 ip ospf message-digest-key 1 md5 CISCO
!
router ospf 1
 area 0 authentication
 area 34 authentication message-digest
 network 10.23.1.0 0.0.0.255 area 0
 network 10.34.1.0 0.0.0.255 area 34
 network 192.168.0.0 0.0.255.255 area 0
```

**R4**

```
interface GigabitEthernet0/1
 ip address 10.34.1.4 255.255.255.0
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 CISCO
!
router ospf 1
 network 0.0.0.0 255.255.255.255 area 34
```

Verification of authentications settings is accomplished by examining the OSPF interface without the brief option. [Example 6-38](#) provides sample output from R3, where the Gi0/0 interface uses MD5 authentication and the Gi0/1 interface uses plaintext authentication. MD5 authentication will also identify the key number that the interface uses.

**Example 6-38 IOS OSPF Authentication Verification**

[Click here to view code image](#)

```
R3#show ip ospf interface GigabitEthernet 0/0
GigabitEthernet0/0 is up, line protocol is up
 Internet Address 10.23.1.3/24, Area 0, Attached via Network Statement
! Output omitted for brevity
 Suppress hello for 0 neighbor(s)
 Simple password authentication enabled

R3#show ip ospf interface GigabitEthernet 0/1
GigabitEthernet0/1 is up, line protocol is up
 Internet Address 10.34.1.3/24, Area 34, Attached via Network Statement
! Output omitted for brevity
 Suppress hello for 0 neighbor(s)
 Message digest authentication enabled
 Youngest key id is 1
```

[Example 6-39](#) verifies the authentication settings for XR2, where the Gi0/0/0/0 interface uses MD5 authentication with key number 1, and the Gi0/0/0/2 interface uses plaintext authentication. Notice that IOS XR does not identify plaintext authentication. Verification of

plaintext authentication can be done only by looking at the running configuration.

### Example 6-39 IOS XR OSPF Authentication Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show ospf interface GigabitEthernet 0/0/0/0
```

```
GigabitEthernet0/0/0/0 is up, line protocol is up  
Internet Address 10.12.1.2/24, Area 12
```

```
! Output omitted for brevity
```

```
Suppress hello for 0 neighbor(s)
```

```
Message digest authentication enabled
```

```
Youngest key id is 1
```

```
Multi-area interface Count is 0
```

```
RP/0/0/CPU0:XR2#show ospf interface GigabitEthernet 0/0/0/2
```

```
GigabitEthernet0/0/0/2 is up, line protocol is up  
Internet Address 10.23.1.2/24, Area 0
```

```
! Output omitted for brevity
```

```
Suppress hello for 0 neighbor(s)
```

```
Multi-area interface Count is 0
```

## SUMMARY

OSPF is one of the first link-state IGPs explained in this book. Link-state protocols overcome the issues associated with classic distance vector routing protocols. OSPF is commonly found in enterprises and service provider networks.

This chapter provided a brief overview of the OSPF protocol and the core functionality within a single OSPF Area. OSPF inter-router communication may seem complicated at first, but it is vital knowledge when troubleshooting issues. Forming an OSPF adjacency requires that the following parameters match on OSPF-enabled interfaces:

- Hello and dead timers
- Network mask (for broadcast network types only)
- Need for a DR
- Matching MTU (unless configured to ignore MTU)
- Unique RIDs
- Interfaces sharing a common subnet with matching OSPF area ID
- Authentication type and credentials
- Area flags

OSPF use a designated router (DR) and backup designated router (BDR) to optimize multi-access



networks by reducing the number of complete adjacencies for that network segment. OSPF automatically classifies multiple media types based on operating characteristics and identifies whether a DR is required and the ideal OSPF hello and dead timer values. OSPF detects topology changes quickly, and after alerting other OSPF nodes within the area, all routers recalculate the SPF tree to identify a new path.

Passive interfaces provide a method of advertising network prefixes into OSPF, while preventing an adjacency from forming on that interface. Enabling OSPF authentication ensures that only authorized routers can form an adjacency and exchange routes. This prevents manipulation of routing tables, which could lead to data interception, distributed DoS (DDoS) attacks, or other malicious activity.

The next chapter further explains other advanced OSPF topics involving the LSDB, multi-area topologies, SPF path selection, summarization techniques, and common problematic OSPF design scenarios.

## REFERENCES IN THIS CHAPTER

RFC 2328, *OSPF Version 2*, John Moy, IETF, <http://www.ietf.org/rfc/rfc2328.txt>, April 1998

Doyle, Jeff, Carrol, Jennifer. *Routing TCP/IP Volume I*, 2nd Edition. Indianapolis: Cisco Press, 2006.

Cisco. Cisco IOS Software Configuration Guides. <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides. <http://www.cisco.com>

## Chapter 7. Advanced OSPF

This chapter covers the following topics:

- OSPF areas
- OSPF router roles and functions
- Link-state advertisements (LSAs)
- Path selection
- OSPF summarization

The Open Shortest Path First (OSPF) Protocol scales well with proper network planning. IP addressing schemes, area segmentation, address summarization, and hardware capabilities for each area should all be taken into consideration for the network design.

This chapter expands on [Chapter 6, “OSPF,”](#) and explains the functions and features found in larger enterprise and service provider networks. By the end of this chapter, you should have a solid understanding of the route advertisement within a multi-area OSPF domain, path selection, and techniques to optimize an OSPF environment.

### AREAS

OSPF areas are a logical grouping of routers, or more specifically a logical grouping of router interfaces. Area membership is set at the interface level and the area ID is included in the OSPF Hello packet. An interface can only belong to one area. All routers within the same OSPF area maintain an identical copy of the link-state database (LSDB).

An OSPF area will grow in size as network links and number of routers increase in the area. While using a single area simplifies the topology, there are trade-offs:

- Full shortest-path first (SPF) tree calculation will run when a link flaps within the area.
- The LSDB increases in size and becomes unmanageable.
- The LSDB for the area grows, consuming more memory, and taking longer during the SPF computation process.
- No summarization of route information.

Proper design addresses each of these issues by segmenting the routers into multiple OSPF areas, thereby keeping the LSDB to a manageable size. Sizing and design of OSPF networks should account for the hardware constraints of the smallest router in that area.

If a router has interfaces in multiple areas, the router has multiple LSDBs (one for each area). The internal topology of one area is invisible from outside of that area. If a topology change occurs (link flap or additional network added) within an area, all routers in the same OSPF area

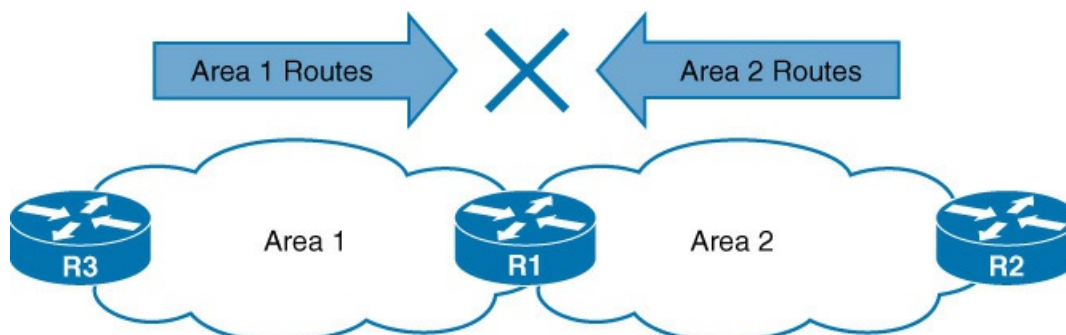
calculate the SPF tree again. Routers outside of that area do not calculate the full SPF tree again but will perform a partial SPF calculation if the metrics have changed or a prefix is removed.

In essence, an OSPF area hides the topology from another area but allows for the networks to be visible in other areas within the OSPF domain. Segmenting the OSPF domain into multiple areas reduces the size of the LSDB for each area, making SPF tree calculations faster and decreasing LSDB flooding between routers when a link flaps.

Area 0 is a special area called the *backbone*. By design, all areas must connect to Area 0 because OSPF expects all areas to inject routing information into it, and Area 0 will advertise the routes into other areas. The backbone design is crucial to preventing routing loops.

Area Border Routers (ABRs) are OSPF routers connected to Area 0 and another OSPF area per Cisco definition according to RFC 3509. ABRs are responsible for advertising routes from one area and injecting them into a different OSPF area. Every ABR needs to participate in Area 0; otherwise, routes will not advertise into another area. ABRs compute a SPF tree for every area that they participate in.

Just because a router connects to multiple OSPF areas does not mean it will function as an ABR. Figure 7-1 shows router R1 connected to Area 1 and Area 2. R1 does not have a connection to Area 0, so routes from Area 1 will not advertise into Area 2 and vice versa.



**Figure 7-1** Failed Route Advertisement Between Areas

Figure 7-2 shows that R1 is connected to Area 0, Area 1, and Area 2. R1 is a proper ABR router because it now participates in Area 0. Routes from Area 0 advertise into Area 1 and Area 2, routes from Area 1 advertise into Area 0 and Area 2, and routes from Area 2 advertise into Area 0 and Area 1.

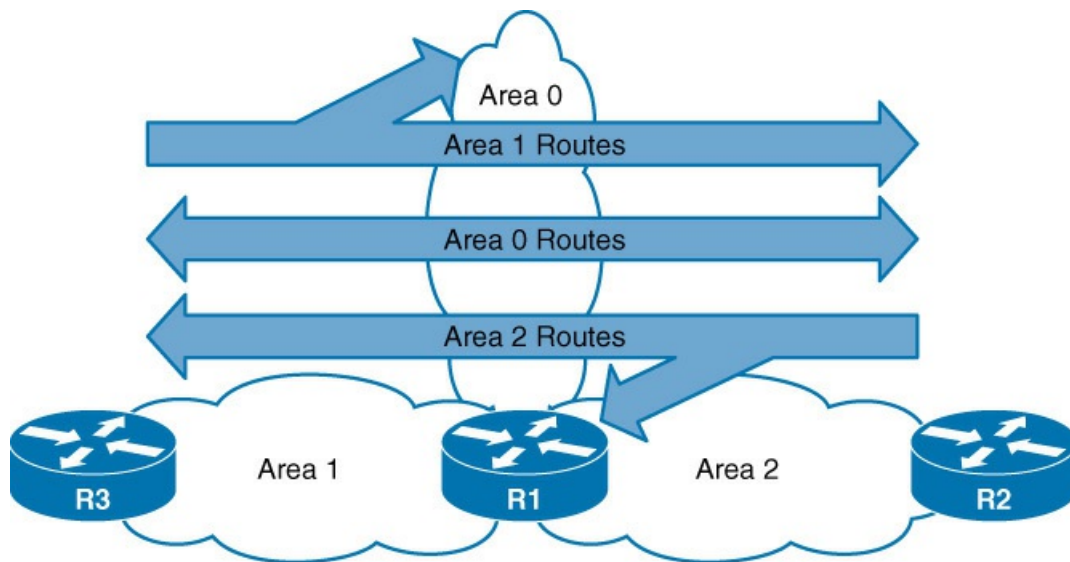


Figure 7-2 Successful Route Advertisement Between Areas

Figure 7-3 shows a larger-scale OSPF multi-area topology that is used throughout this chapter to describe OSPF concepts.

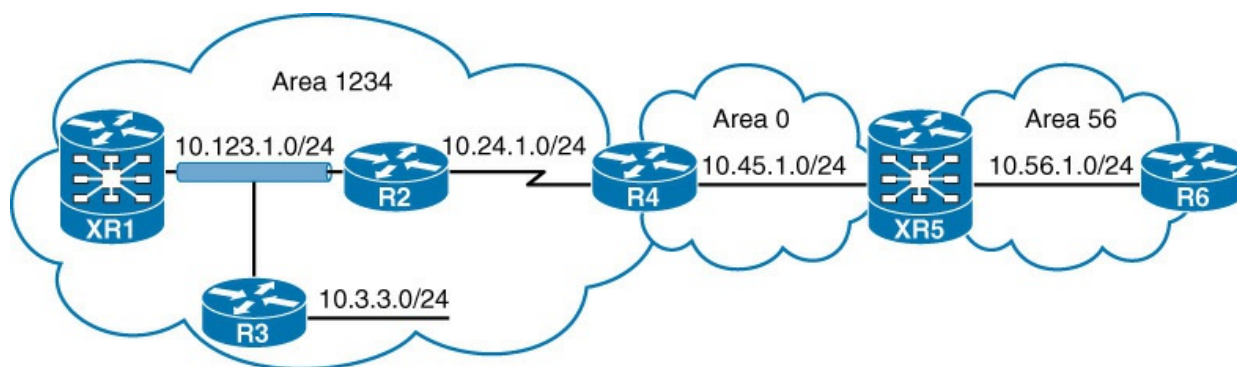


Figure 7-3 OSPF Multi-Area Topology

In the topology

- XR1, R2, R3, and R4 belong to Area 1234.
- R4 and XR5 belong to Area 0.
- XR5 and R6 belong to Area 56.
- R4 and XR5 are ABRs.
- Area 1234 connects to Area 0, and Area 56 connects to Area 0.
- Routers in Area 1234 can see routes from routers in Area 0 and Area 56, and vice versa.

Example 7-1 demonstrates the OSPF configuration for the ABRs R4 and XR5. Notice that there are multiple areas in the configuration.

**Example 7-1** Sample Multi-Area OSPF Configuration

[Click here to view code image](#)

#### R4

```
router ospf 1
router-id 192.168.4.4
network 10.24.1.0 0.0.0.255 area 1234
network 10.45.1.0 0.0.0.255 area 0
```

#### XR5

```
router ospf 1
router-id 192.168.5.5
area 0
interface GigabitEthernet0/0/0/2
!
area 56
interface GigabitEthernet0/0/0/3
```

**Example 7-2** verifies that interfaces on R4 belong to Area 1234 and Area 0 and that interfaces on XR5 belong to Area 0 and Area 56.

### **Example 7-2** Verification of Interfaces for ABRs

[Click here to view code image](#)

```
R4#show ip ospf interface brief
```

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Gi0/0	1	0	10.45.1.4/24	1	DR	1/1	
Se1/0	1	1234	10.24.1.4/29	64	P2P	1/1	

```
RP/0/0/CPU0:XR5#show ospf interface brief
```

Indicates MADJ interface, (P) Indicates fast detect hold down state

```
Interfaces for OSPF 1
```

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Gi0/0/0/2	1	0	10.45.1.5/24	1	BDR	1/1	
Gi0/0/0/3	1	56	10.56.1.5/24	1	DR	0/0	

### **Area ID**

The area is a 32-bit field and can take the format in simple decimal (0–4294967295) or dotted-decimal (0.0.0.0 – 255.255.255.255). During router configuration, the area can use decimal format on one router and dotted-decimal format on a different router, and the routers will still form an adjacency. The OSPF advertises the area ID in dotted-decimal format in the OSPF Hello packet, as shown in [Figure 7-4](#).

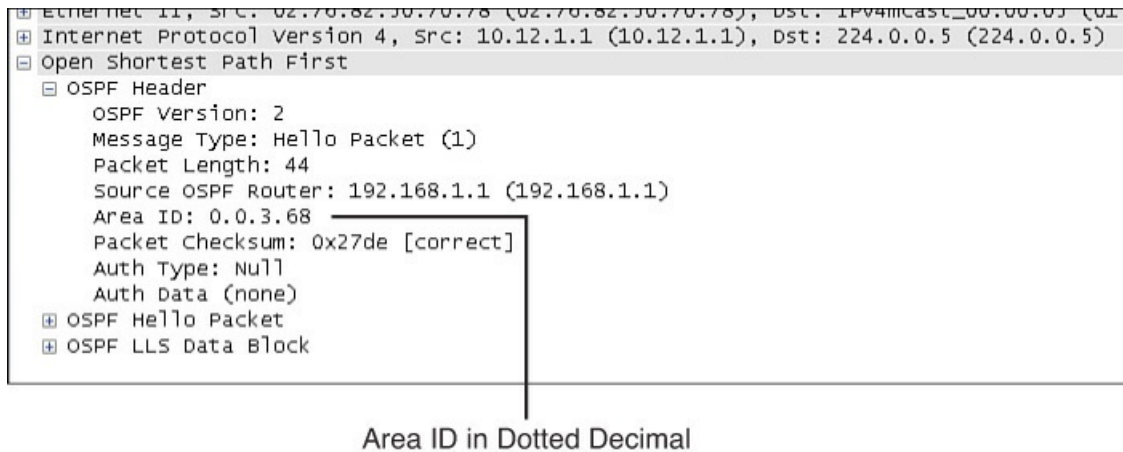


Figure 7-4 OSPF Area ID in OSPF Hello

Most organizations use the decimal format for the area ID, but this section describes the process to convert the decimal version to dotted-decimal format. The example uses the decimal area ID of 836.

**Step 1. Convert the number to binary.**

Convert the number to binary format using the technique shown in Chapter 2, “IPv4 Addressing.” Figure 7-5 demonstrates the technique. Notice that the ninth digit is  $2^8$  (that is, 256), and the tenth digit is  $2^9$  (that is, 512).

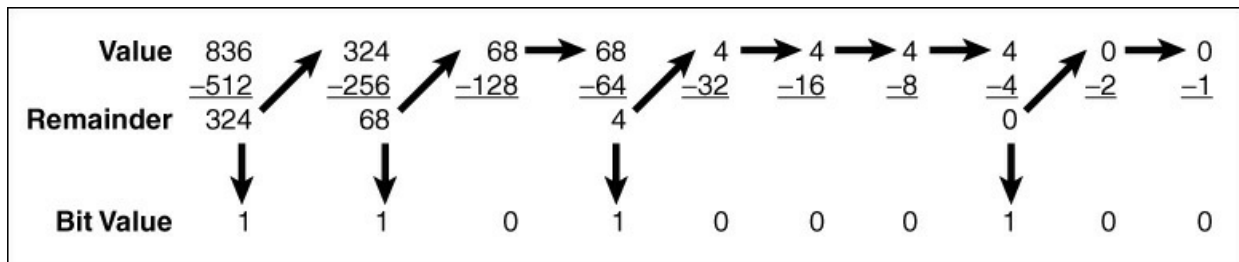


Figure 7-5 Decimal to Binary Conversion

**Step 2. Separate the binary into four octets.**

Split the binary into four octets starting with the furthest right number. Add 0s as required to complete each octet:

00000000.00000000.00000011.01000100

**Step 3. Convert each octet to dotted-decimal format.**

Convert each octet to dotted-decimal format, as shown in Chapter 2. 0.0.3.68 is the same as 836. Notice that our calculations provide the same area ID as the packet capture in Figure 7-4.

Note

Cisco will accept either format during configuration.

## OSPF Route Types

Network routes that are learned from other OSPF routers within the same area are known as *intra-area routes*. In [Figure 7-3](#), the network link between R2 and R4 (10.24.1.0/24) is an intra-area route to XR1. The IP routing table displays OSPF intra-area routes with an *O*.

Network routes that are learned from other OSPF routers from a different area via an ABR are known as *interarea routes*. In [Figure 7-3](#), the network link between R4 and XR5 (10.45.1.0/24) is an interarea route to XR1. The IP routing table displays OSPF interarea routers with *O IA*.

[Example 7-3](#) provides the routing table for XR1 from [Figure 7-3](#). Notice XR1's OSPF route table shows routes from within Area 1234 as intra-area (O routes) and routes from Area 0 and Area 56 as interarea (O IA routes).

### Example 7-3 OSPF Routing Tables for Sample Multi-Area OSPF Topology

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route ospf

O    10.3.3.0/24 [110/2] via 10.123.1.3, 00:03:06, GigabitEthernet0/0/0/2
O    10.24.1.0/29 [110/65] via 10.123.1.2, 00:08:07, GigabitEthernet0/0/0/2
O IA 10.45.1.0/24 [110/66] via 10.123.1.2, 00:08:07, GigabitEthernet0/0/0/2
O IA 10.56.1.0/24 [110/67] via 10.123.1.2, 00:04:12, GigabitEthernet0/0/0/2
```

[Example 7-4](#) provides the routing table for R4 from [Figure 7-3](#). Notice that R4's OSPF route table shows the routes from within Area 1234 and Area 0 as intra-area and routes from Area 56 as interarea because R4 does not connect to Area 56.

### Example 7-4 OSPF Routing Tables for Sample Multi-Area OSPF Topology

[Click here to view code image](#)

```
R4#show ip route ospf | begin Gateway
Gateway of last resort is not set

    10.0.0.0/8 is variably subnetted, 7 subnets, 3 masks
O    10.3.3.0/24 [110/66] via 10.24.1.2, 00:03:45, Serial1/0
O IA 10.56.1.0/24 [110/2] via 10.45.1.5, 00:04:56, GigabitEthernet0/0
O    10.123.1.0/24 [110/65] via 10.24.1.2, 00:13:19, Serial1/0
```

[Example 7-5](#) provides the routing table for XR5 and R6 from [Figure 7-3](#). XR5 and R6 contain only interarea routes in the OSPF routing table because intra-area routes are directly connected.

### Example 7-5 OSPF Routing Tables for Sample Multi-Area OSPF Topology

[Click here to view code image](#)

```
RP/0/0/CPU0:XR5#show route ospf
```

```
○ IA 10.3.3.0/24 [110/67] via 10.45.1.4, 00:06:16, GigabitEthernet0/0/0/2
○ IA 10.24.1.0/29 [110/65] via 10.45.1.4, 00:07:30, GigabitEthernet0/0/0/2
○ IA 10.123.1.0/24 [110/66] via 10.45.1.4, 00:07:30, GigabitEthernet0/0/0/2
```

```
R6#show ip route ospf | begin Gateway
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 6 subnets, 3 masks
```

```
○ IA 10.3.3.0/24 [110/68] via 10.56.1.5, 00:07:04, GigabitEthernet0/0
○ IA 10.24.1.0/29 [110/66] via 10.56.1.5, 00:08:19, GigabitEthernet0/0
○ IA 10.45.1.0/24 [110/2] via 10.56.1.5, 00:08:18, GigabitEthernet0/0
○ IA 10.123.1.0/24 [110/67] via 10.56.1.5, 00:08:19, GigabitEthernet0/0
```

### External OSPF Routes

External routes are routes learned from outside of the OSPF domain but are injected into an OSPF domain through redistribution. External OSPF routes can come from a different OSPF domain or from a different routing protocol. Route redistribution is explained in detail in [Chapter 13, “Route Redistribution.”](#)

When a router redistributes routes into an OSPF domain, the router is called an *Autonomous System Boundary Router (ASBR)*. An ASBR can be any OSPF router, and the ASBR function is independent of the ABR function. An OSPF domain can have an ASBR without having an ABR. An OSPF router can be an ASBR and ABR at the same time.

External routes are classified as Type 1 or Type 2. The main differences between Type 1 and Type 2 external OSPF routes are as follows:

- Type 1 routes are preferred over Type 2.
- The Type 1 metric equals the Redistribution Metric + Total Path Metric to the ASBR. In other words, as the LSA propagates away from the originating ASBR, the metric increases.
- The Type 2 metric equals only the redistribution metric. The metric is the same for the router next to the ASBR as the router 30 hops away from the originating ASBR. This is the default external metric type used by OSPF.

[Figure 7-6](#) shows a multi-area OSPF topology with R6 redistributing external routes into the OSPF domain. R6 is an ASBR, and R4 and XR5 are ABRs for this OSPF domain.



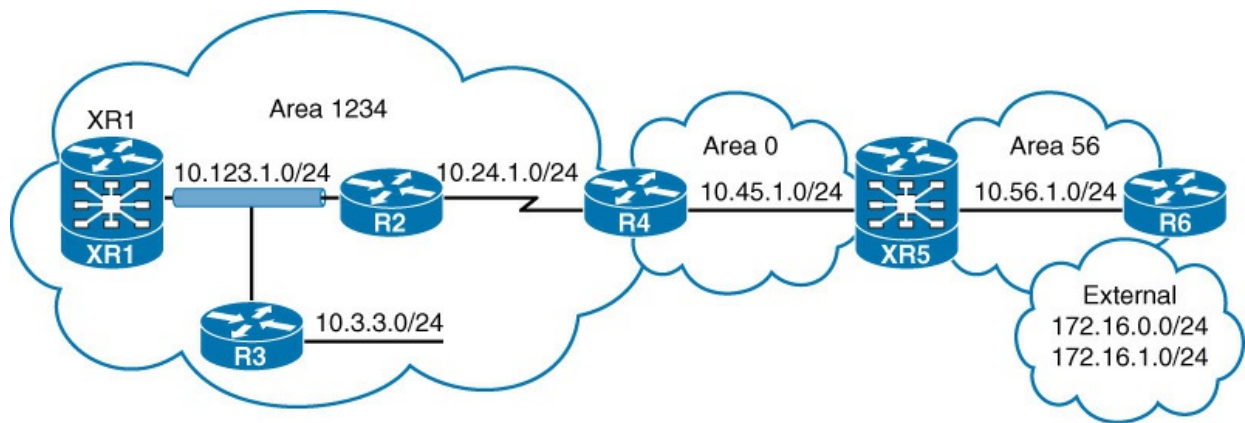


Figure 7-6 OSPF Multi-Area Domain with External Routes

Example 7-6 provides the route table of R2 and XR5. The 172.16.0.0/24 network is redistributed as a Type 2 route, and the 172.16.1.0/24 network is redistributed as a Type 1 route.

External OSPF network routes are marked as O E1 and O E2 in the route table and correlate with OSPF Type 1 and Type 2 external routes accordingly. Notice that the metric for the 172.16.0.0/24 network is the same on R2 as it is on XR5, but the metric for the 172.16.1.0/24 differs on the two routers because Type 1 external metrics include the path metric to the ASBR.

### Example 7-6 Route Table on R2 and XR5

[Click here to view code image](#)

```
R2#show ip route ospf | begin Gateway
Gateway of last resort is not set

    10.0.0.0/8 is variably subnetted, 7 subnets, 3 masks
O       10.3.3.0/24 [110/2] via 10.123.1.3, 01:23:14, GigabitEthernet0/0
O IA    10.45.1.0/24 [110/65] via 10.24.1.4, 01:33:21, Serial1/0
O IA    10.56.1.0/24 [110/66] via 10.24.1.4, 01:24:25, Serial1/0
    172.16.0.0/24 is subnetted, 2 subnets
O E2    172.16.0.0 [110/20] via 10.24.1.4, 00:02:27, Serial1/0
O E1    172.16.1.0 [110/86] via 10.24.1.4, 00:00:33, Serial1/0

RP/0/0/CPU0:XR5#show route ospf

O IA 10.3.3.0/24 [110/67] via 10.45.1.4, 01:24:51, GigabitEthernet0/0/0/2
O IA 10.24.1.0/29 [110/65] via 10.45.1.4, 01:26:06, GigabitEthernet0/0/0/2
O IA 10.123.1.0/24 [110/66] via 10.45.1.4, 01:26:06, GigabitEthernet0/0/0/2
O E2 172.16.0.0/24 [110/20] via 10.56.1.6, 00:04:04, GigabitEthernet0/0/0/3
O E1 172.16.1.0/24 [110/21] via 10.56.1.6, 00:02:10, GigabitEthernet0/0/0/3
```

## LINK-STATE ANNOUNCEMENTS

When OSPF neighbors become adjacent, the LSDBs synchronize between the OSPF routers. As an OSPF router adds or removes a directly connected network link to or from its database, the router floods the link-state advertisement (LSA) out all its active OSPF interfaces. The OSPF LSA contains a complete list networks advertised from that router.

Figure 7-7 revisits the sample OSPF topology from Chapter 6 (Figure 6-11), but shows R2 with a second interface (10.22.22.0/24).

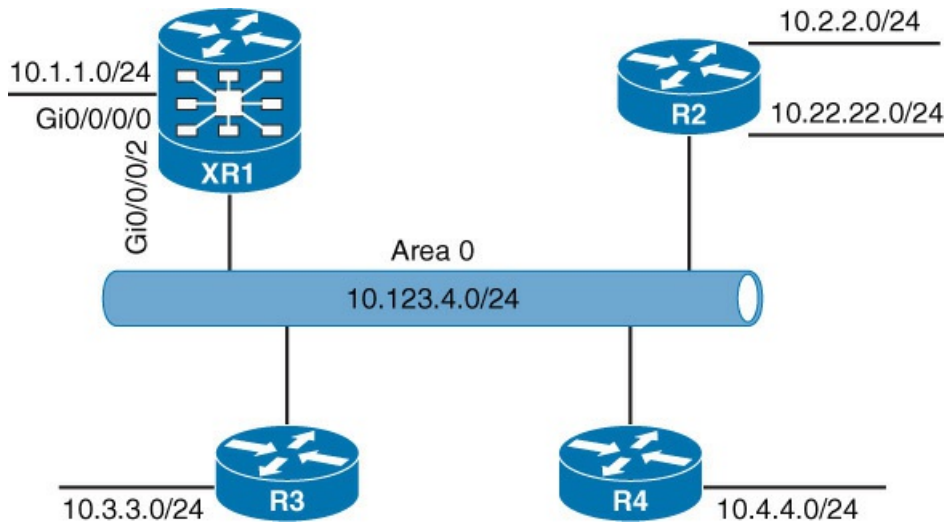


Figure 7-7 Topology Example with Second Interface on R2

Figure 7-8 shows the packet capture of R2's update LSA when the Gi0/2 (10.22.22.0/24) interface is OSPF-enabled. Notice that all of R2's interfaces correspond with a link ID as part of the LSA.

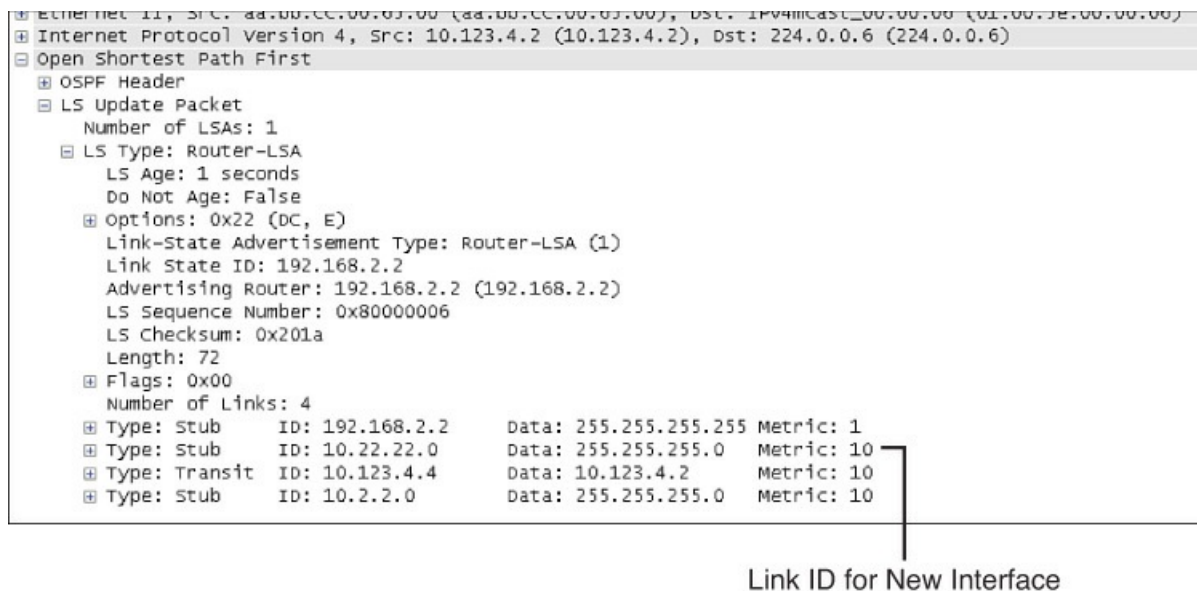


Figure 7-8 Packet Capture of LSA Update for Second Interface

If R2 shuts down or disables OSPF on interface Gigabit Ethernet 0/2 (10.22.22.0/24), R2 will send out a new LSA minus the entry for the Gigabit Ethernet 0/2 interface. The OSPF update LSA looks like the packet capture in Figure 7-9. Notice that the 10.22.22.0 link ID is missing.

```

Internet Protocol Version 4, Src: 10.123.4.2 (10.123.4.2), Dst: 224.0.0.6 (224.0.0.6)
Open Shortest Path First
  OSPF Header
  LS Update Packet
    Number of LSAs: 1
    LS Type: Router-LSA
      LS Age: 1 seconds
      Do Not Age: False
      Options: 0x22 (DC, E)
      Link-State Advertisement Type: Router-LSA (1)
      Link State ID: 192.168.2.2
      Advertising Router: 192.168.2.2 (192.168.2.2)
      LS Sequence Number: 0x80000007
      LS Checksum: 0x3653
      Length: 60
      Flags: 0x00
      Number of Links: 3
      Type: Stub      ID: 192.168.2.2      Data: 255.255.255.255 Metric: 1
      Type: Transit  ID: 10.123.4.4      Data: 10.123.4.2      Metric: 10
      Type: Stub      ID: 10.2.2.0        Data: 255.255.255.0   Metric: 10
  
```

Link ID for 10.22.22.0 Removed

Figure 7-9 Packet Capture of LSA Update for Second Interface Removal

When an OSPF router receives a Type 1 LSA, it checks all the link IDs in the LSA for an entry in the LSDB. If a link ID does not exist, the router processes the LSA by adding the link ID and then floods the LSA out all interfaces for that area, except the interface from which the original LSA was received.

Using Figure 7-10 as an example, if the 192.168.0.0/24 network is added, R4 will receive LSA from two paths, the serial path (R1-R3-R4) and the Ethernet path (R1-R2-R4). The LSA traveling across R1-R3-R4 may arrive later than the other path because of the serial links. R4 detects the LSA is already present in its LSDB and does not flood the LSA onto R2.

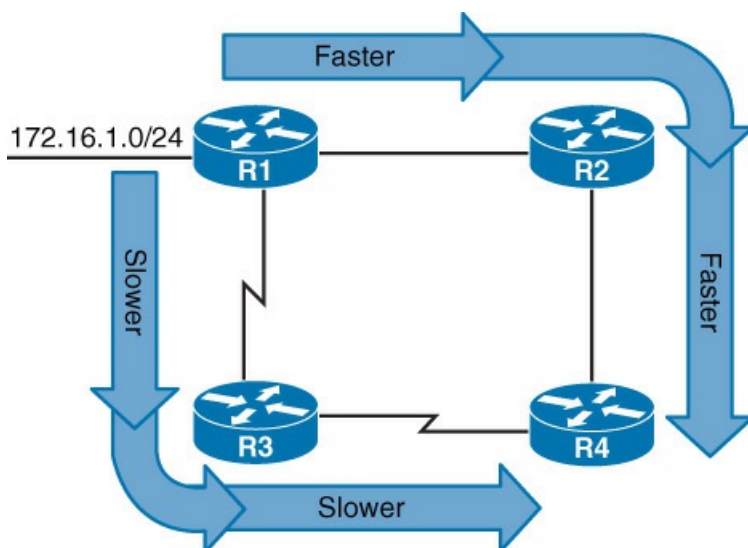


Figure 7-10 OSPF Topology with Serial and Ethernet Interfaces

This might not be a problem when a route is added to the topology; however, if the 172.16.1.0/24 network were to go down and come back up again, it could cause a problem. It is possible that the LSAs for the withdrawals (link down) and re-advertisement (link up) travel across the faster Ethernet path, and reach R4 faster than the slower serial links. R4 would process the LSAs sent from the Ethernet path and from the serial path and flapping the route a second time on R2.

OSPF uses a sequence number to overcome problems caused by delays in LSA propagation in a network. The LSA sequence number is a 32-bit number to control versioning. When the originating router sends LSAs out, the LSA sequence number is incremented. If a router receives an LSA sequence that is greater than the one in the LSDB, it processes the LSA. If the LSA sequence number is lower than the one in the LSDB, the router deems the LSA as old and discards the LSA.

Note

An exception applies to this logic because at some time the sequence numbers become too large and must reset back to 0 and start all over again. Before the sequence number is reset, the current instance of the LSA must first be flushed by prematurely aging the LSA and reflooding it. As soon as this flood has been acknowledged by all the neighbors, a new instance can be originated with an initial sequence number.

The frequency of this happening is so low that if an LSA were to reset every second, it would take approximately 6 years to reset back to the initial sequence.

### LSA Age and Flooding

All OSPF LSAs include an age that is entered into the local LSDB that will increment by one every second. When a router's OSPF LSA age exceeds 1800 seconds (30 minutes) for its networks, the originating router will advertise a new LSA with the LSA age set to zero. As each router forwards the LSA, the LSA age is incremented with a calculated delay that reflects the link (it is minimal). If the LSA Age reaches 3600, then the LSA is deemed invalid and is purged from the LSDB. The repetitive flooding of LSAs is a secondary safety mechanism to ensure that all routers maintain a consistent LSDB within an area.

### LSA Types

All routers within an OSPF area have an identical set of LSAs for that area. The ABRs will maintain a separate set of LSAs for each OSPF area. Most LSAs in one area will be different from the LSAs in another area. The Generic Router LSA output is shown with the command **show ip ospf database** for IOS nodes and with the command **show ospf database** on IOS XR nodes.

### LSA Type 1: Router Link

Every OSPF router advertises a Type 1 LSA. Type 1 LSAs are the essential building blocks within the LSDB. A Type 1 LSA entry exists for all OSPF-enabled links (that is, interfaces). Figure 7-11 demonstrates that the Type 1 LSAs are not advertised outside of an area, thus making the topology in an area invisible to other areas.

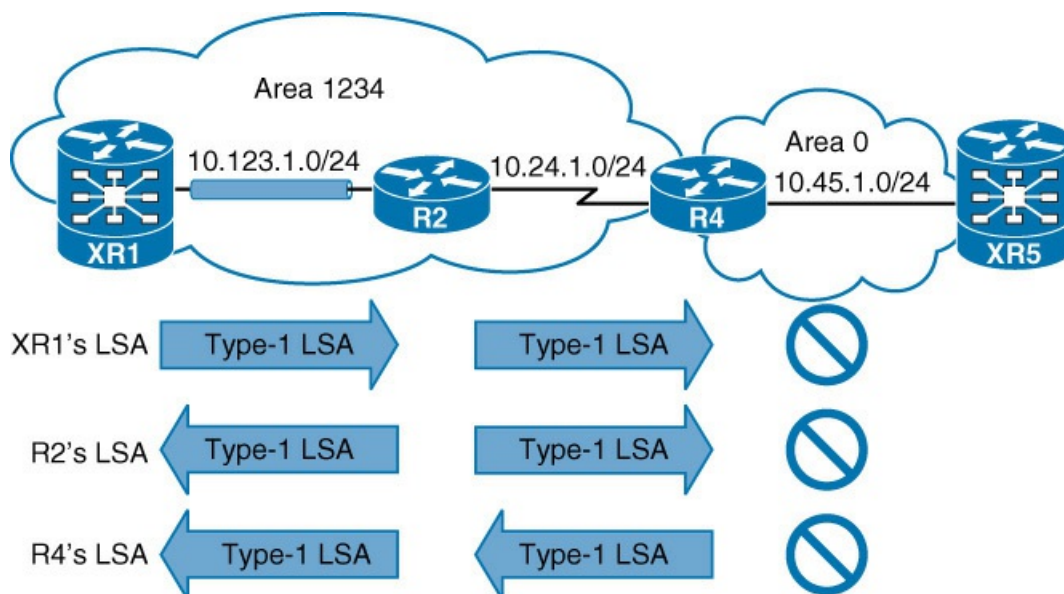


Figure 7-11 Type 1 LSA Flooding in an Area

A brief summary view of the Type 1 LSAs for an area are under the *Router Link States* column within the LSDB, as shown In [Example 7-7](#).

**Example 7-7** *Generic OSPF LSA Output for Type 1 LSAs*

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show ospf database

OSPF Router with ID (192.168.1.1) (Process ID 1)

Router Link States (Area 1234)

Link ID          ADV Router      Age             Seq#            Checksum Link count
192.168.1.1     192.168.1.1    539            0x80000003     0x00a4a4 1
192.168.2.2     192.168.2.2    551            0x80000004     0x005c95 3
192.168.3.3     192.168.3.3    563            0x80000004     0x0042da 2
192.168.4.4     192.168.4.4    590            0x80000003     0x00838a 2
```

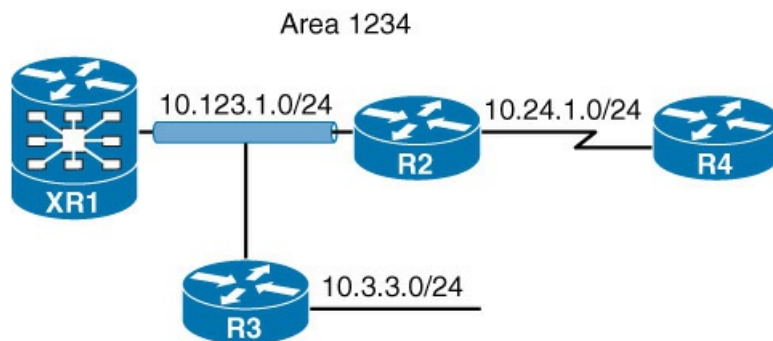
[Table 7-1](#) provides an overview of the fields used within the LSDB output.

Field	Description
Link ID	The OSPF router ID (RID) for this LSA.
ADV Router	The OSPF router ID for this LSA.
Age	Age of the LSA on the router the command is being run on. Values over 1800 are expected to refresh soon.
Seq #	Sequence number for the LSA to protect out of order LSAs.
Checksum	Checksum of the LSA to verify the integrity during flooding.
Link Count	Number of links on this router in the Type 1 LSA.

**Table 7-1** *OSPF LSDB Fields*

Type 1 LSAs for an area are shown with the command **show ip ospf database router** on IOS nodes or with the command **show ospf database router** on IOS XR nodes.

[Figure 7-12](#) is a reference subsection of Area 1234 taken from the original [Figure 7-3](#).



**Figure 7-12** *Type 1 LSA Flooding Within an Area*

[Example 7-8](#) provides output of all the Type 1 LSAs for Area 1234 in [Figure 7-12](#). Notice in the

output that entries exist for all four routers in the area.

### Example 7-8 OSPF Type 1 LSAs for Area 1234

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show ospf database router

OSPF Router with ID (192.168.1.1) (Process ID 1)

          Router Link States (Area 1234)

LS age: 691
Options: (No TOS-capability, DC)
LS Type: Router Links
Link State ID: 192.168.1.1
Advertising Router: 192.168.1.1
LS Seq Number: 80000003
Checksum: 0xa4a4
Length: 36
Number of Links: 1

Link connected to: a Transit Network
(Link ID) Designated Router address: 10.123.1.3
(Link Data) Router Interface address: 10.123.1.1
Number of TOS metrics: 0
TOS 0 Metrics: 1

Routing Bit Set on this LSA
LS age: 704
Options: (No TOS-capability, DC)
LS Type: Router Links
Link State ID: 192.168.2.2
Advertising Router: 192.168.2.2
LS Seq Number: 80000004
Checksum: 0x5c95
Length: 60
Number of Links: 3

Link connected to: a Transit Network
(Link ID) Designated Router address: 10.123.1.3
(Link Data) Router Interface address: 10.123.1.2
Number of TOS metrics: 0
TOS 0 Metrics: 1

Link connected to: another Router (point-to-point)
(Link ID) Neighboring Router ID: 192.168.4.4
(Link Data) Router Interface address: 10.24.1.2
Number of TOS metrics: 0
TOS 0 Metrics: 64

Link connected to: a Stub Network
(Link ID) Network/subnet number: 10.24.1.0
(Link Data) Network Mask: 255.255.255.248
Number of TOS metrics: 0
TOS 0 Metrics: 64
```

```

Routing Bit Set on this LSA
LS age: 716
Options: (No TOS-capability, DC)
LS Type: Router Links
Link State ID: 192.168.3.3
Advertising Router: 192.168.3.3
LS Seq Number: 80000004
Checksum: 0x42da
Length: 48
Number of Links: 2

Link connected to: a Transit Network
(Link ID) Designated Router address: 10.123.1.3
(Link Data) Router Interface address: 10.123.1.3
Number of TOS metrics: 0
TOS 0 Metrics: 1

Link connected to: a Stub Network
(Link ID) Network/subnet number: 10.3.3.0
(Link Data) Network Mask: 255.255.255.0
Number of TOS metrics: 0
TOS 0 Metrics: 1

Routing Bit Set on this LSA
LS age: 742
Options: (No TOS-capability, DC)
LS Type: Router Links
Link State ID: 192.168.4.4
Advertising Router: 192.168.4.4
LS Seq Number: 80000003
Checksum: 0x838a
Length: 48
Area Border Router
Number of Links: 2

Link connected to: another Router (point-to-point)
(Link ID) Neighboring Router ID: 192.168.2.2
(Link Data) Router Interface address: 10.24.1.4
Number of TOS metrics: 0
TOS 0 Metrics: 64

Link connected to: a Stub Network
(Link ID) Network/subnet number: 10.24.1.0
(Link Data) Network Mask: 255.255.255.248
Number of TOS metrics

```

The initial fields of each Type 1 LSA are the same as in [Table 7-1](#). If a router is functioning as an ABR, ASBR, or virtual-link endpoint, the function is listed between the *Length* and *Number of links* fields. In the preceding output, R4 (192.168.4.4) is an ABR.

Each OSPF-enabled interface is listed under the number of links for each router. Each network link on a router contains the following information in this order:



- Link type, as shown in [Table 7-2](#), after the *Link connected to*

Description	Link Type	Link-ID Value	Link Data
Point-to-point link (IP address assigned)	1	Neighbor RID	Interface IP address
Point-to-point link (using IP Unnumbered)	1	Neighbor RID	MIB II IfIndex value
Link to transit network	2	Interface Address of DR	Interface IP address
Link to stub network	3	Network address	Subnet mask
Virtual link	4	Neighbor RID	Interface IP address

**Table 7-2** OSPF Neighbor States for Type 1 LSAs

- Link ID using the values based of the link type listed in [Table 7-2](#)
- Link data (when applicable)
- Metric for the interface

During the SPF tree calculation, network link types are one of the following:

- **Transit:** A transit network indicates that an adjacency has been formed and that a DR has been elected on that link.
- **Point-to-point:** Point-to-point links indicate an adjacency has been formed on a network type that does not use a DR.
- **Stub:** A stub network indicates that no neighbor adjacencies have been established on that link.

Point-to-point and transit link types that have not become adjacent with another OSPF router are classified as a stub network link type. When an OSPF adjacency forms then the link type changes to the appropriate type, point-to-point or transit. Interfaces using the OSPF point-to-point network type advertise two links. One link is the point-to-point link type identifying the OSPF neighbor RID for that segment, and the other link is a stub network link that provides the subnet mask for that network.

Note

Secondary connected networks are always advertised as stub link types because an OSPF adjacency can never form on them

The output from [Example 7-8](#) allows for the router to calculate the topology as shown in [Figure 7-13](#). Using only Type 1 LSAs, a connection is made between R2 and R4 because they point to each other's RID in the point-to-point LSA. Notice that the three transit networks on XR1, R2, and R3 (10.123.1.0) have not been directly connected yet.



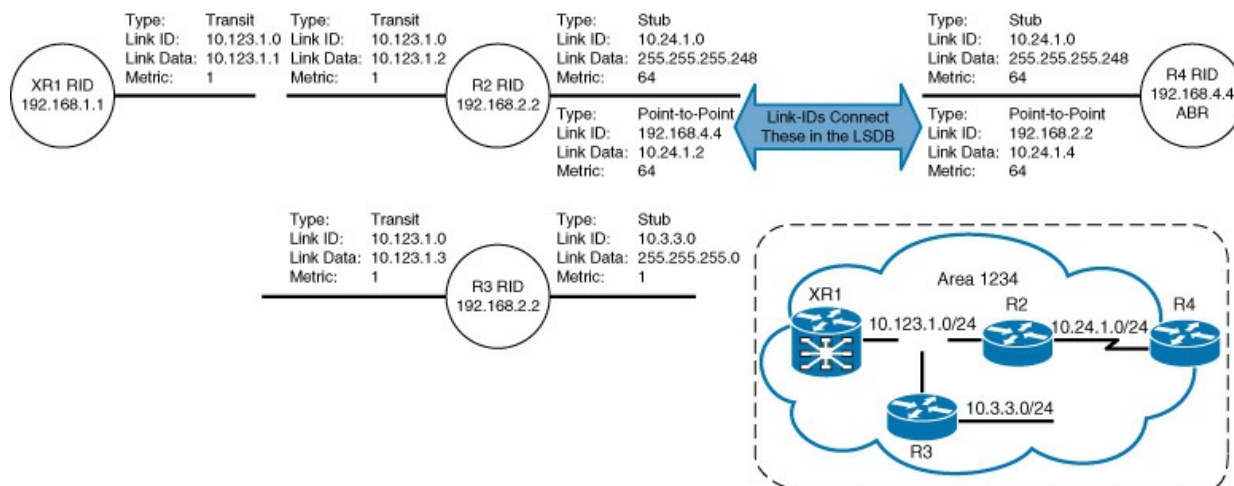


Figure 7-13 Visualization of Type 1 LSAs

## LSA Type 2: Network Link

Type 2 LSAs represent a multi-access network segments that use a DR. The DR always advertises the Type 2 LSA, and identifies all the routers attached to the network. If a DR has not been elected, a Type 2 LSA will not be present in the LSDB because the corresponding Type 1 transit link type LSA will be a stub. Type 2 LSAs are not flooded outside of the originating OSPF area in an identical fashion to Type 1 LSAs.

A brief summary view of the Type 2 LSAs is shown in the LSDB under *Net Link States*. [Example 7-9](#) provides the output for Type 2 LSAs in Area 1234 from the topology example.

### Example 7-9 Generic OSPF LSA Output for Type 2 LSAs

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show ospf database

          OSPF Router with ID (192.168.1.1) (Process ID 1)
! Output omitted for brevity
          Net Link States (Area 1234)

Link ID        ADV Router    Age         Seq#         Checksum
10.123.1.3     192.168.3.3   1100       0x80000005  0x0022ce
! Output omitted for brevity
```

Area 1234 has only 1 DR segment because R3 has not formed an OSPF adjacency on the 10.3.3.0/24 network segment. On the 10.123.1.0/24 network segment, R3 is elected as the DR, and R2 is elected as the BDR because of the order of the RIDs.

Detailed Type 2 LSA information is shown with the command **show ip ospf database network** for IOS nodes and with the command **show ospf database network** for IOS XR nodes.

[Example 7-10](#) confirms that the advertising router is R3 and that the link state ID 10.123.1.3 attaches to XR1, R2, R3 (by listing their RIDs at the bottom). The network mask for the subnet is included in the Type 2 LSA.

### Example 7-10 Detailed Output for OSPF Type 2 LSAs

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show ospf database network
```

```
OSPF Router with ID (192.168.1.1) (Process ID 1)
```

```
Net Link States (Area 1234)
```

```
Routing Bit Set on this LSA
```

```
LS age: 1234
```

```
Options: (No TOS-capability, DC)
```

```
LS Type: Network Links
```

```
Link State ID: 10.123.1.3 (address of Designated Router)
```

```
Advertising Router: 192.168.3.3
```

```
LS Seq Number: 80000005
```

```
Checksum: 0x22ce
```

```
Length: 36
```

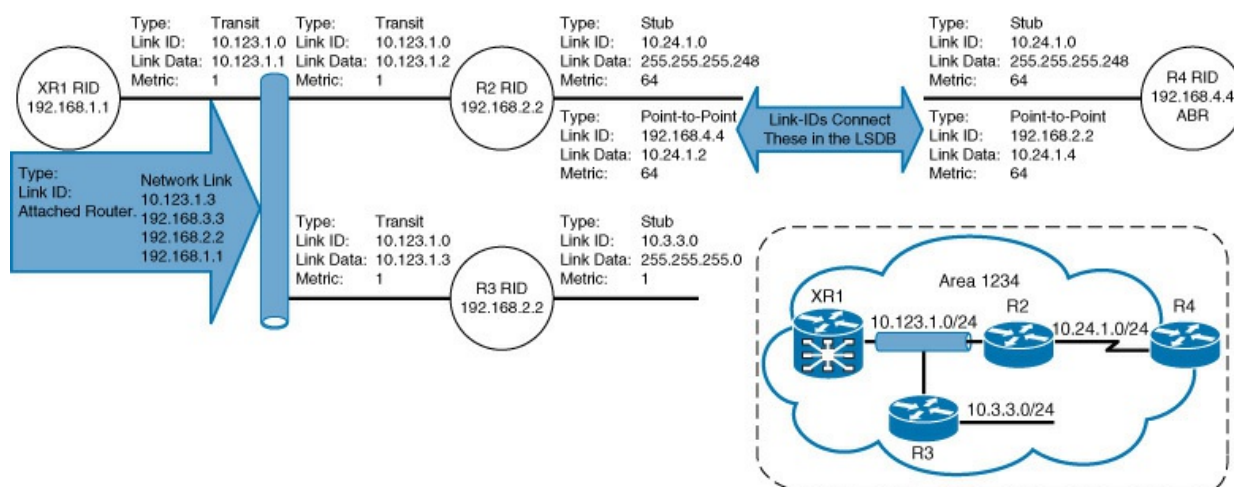
```
Network Mask: /24
```

```
Attached Router: 192.168.3.3
```

```
Attached Router: 192.168.1.1
```

```
Attached Router: 192.168.2.2
```

Now that we have the Type 2 LSA for Area 1234, all the network links are connected. [Figure 7-14](#) provides a visualization of the Type 1 and Type 2 LSAs, which corresponds with Area 1234 perfectly.



**Figure 7-14** Visualization of Area 1234 with Type 1 and Type 2 LSAs

Note

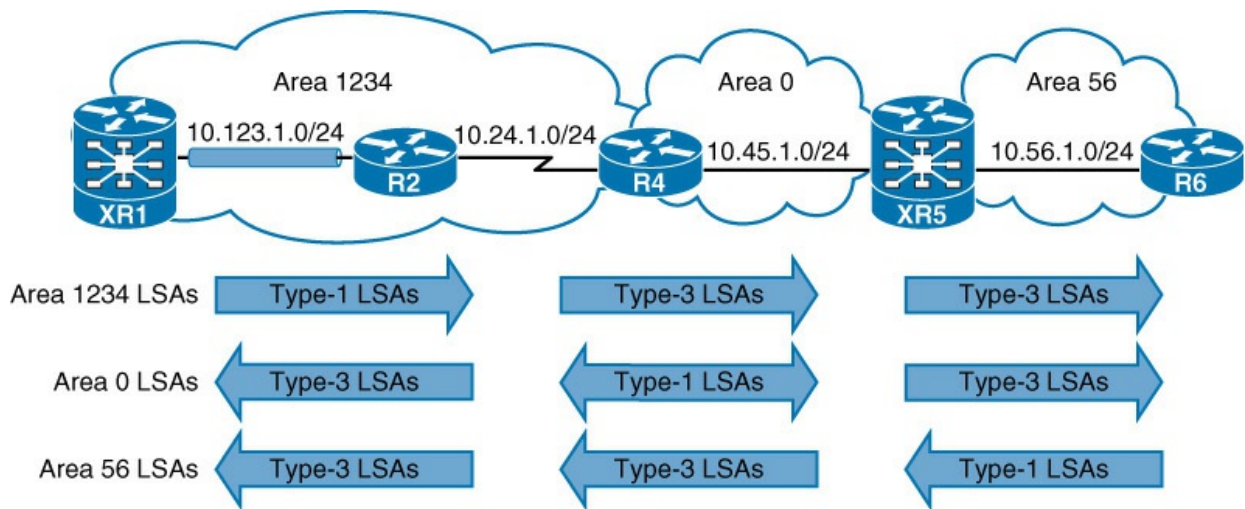
When the DR changes for a network segment, a new Type 2 LSA is created causing SPF to run again within the OSPF area.

### LSA Type 3: Summary Link

Type 3 LSAs represent networks from other areas. The role of the ABRs is to participate in multiple OSPF areas and ensure that the networks associated with Type 1 LSAs are reachable in the nonoriginating OSPF area. ABRs do not forward Type 1 or Type 2 LSAs into other areas. When an ABR receives a Type 1 LSA, it creates a Type 3 LSA referencing the network in the original Type 1 LSA. The ABR then advertises the Type 3 LSA into other areas.

If an ABR receives a Type 3 LSA from Area 0 (backbone), it regenerates a new Type 3 LSA for the

nonbackbone area, and lists itself as the advertising router. [Figure 7-15](#) demonstrates the concept of a Type 3 LSA interaction with Type 1 LSA.



**Figure 7-15** Type 3 LSA Conceptual

A brief summary view of the Type 3 LSAs are under *Summary Net Link*, as shown in [Example 7-11](#). The Type 3 LSAs show up under the appropriate area that they exist in the OSPF domain. For example, the 10.56.1.0 Type 3 LSA is in Area 0 and Area 1234 on R4, but on XR5 the Type 3 LSA exists only in Area 0 because the 10.56.1.0 network is a Type 1 LSA in Area 56.

**Example 7-11** Generic OSPF LSA Output for Type 3 LSAs

[Click here to view code image](#)

```
R4#show ip ospf database

                OSPF Router with ID (192.168.4.4) (Process ID 1)
! Output omitted for brevity
        Summary Net Link States (Area 0)

Link ID        ADV Router    Age           Seq#           Checksum
10.3.3.0       192.168.4.4  1129         0x8000000D   0x00FF6D
10.24.1.0      192.168.4.4  1129         0x8000000D   0x00DA88
10.56.1.0      192.168.5.5  1168         0x8000000F   0x00F97D
10.123.1.0     192.168.4.4  1129         0x8000000D   0x006691
! Output omitted for brevity
        Summary Net Link States (Area 1234)

Link ID        ADV Router    Age           Seq#           Checksum
10.45.1.0      192.168.4.4  1129         0x8000000D   0x008FF6
10.56.1.0      192.168.4.4  1129         0x8000000D   0x001565
```

```
RP/0/0/CPU0:XR5#show ospf database
```

```
OSPF Router with ID (192.168.5.5) (Process ID 1)
```

```
! Output omitted for brevity
```

```
Summary Net Link States (Area 0)
```

Link ID	ADV Router	Age	Seq#	Checksum
10.3.3.0	192.168.4.4	1158	0x8000000d	0x00ff6d
10.24.1.0	192.168.4.4	1158	0x8000000d	0x00da88
10.56.1.0	192.168.5.5	1195	0x8000000f	0x00f97d
10.123.1.0	192.168.4.4	1158	0x8000000d	0x006691

```
! Output omitted for brevity
```

```
Summary Net Link States (Area 56)
```

Link ID	ADV Router	Age	Seq#	Checksum
10.3.3.0	192.168.5.5	1195	0x8000000d	0x00fc6d
10.24.1.0	192.168.5.5	1195	0x8000000d	0x00d788
10.45.1.0	192.168.5.5	1195	0x8000000f	0x007e04
10.123.1.0	192.168.5.5	1195	0x8000000d	0x006391

Detailed Type 3 LSA information is shown with the command **show ip ospf database summary** on IOS nodes and the **show ospf database summary** command on IOS XR nodes. The output can be restricted to a specific LSA by adding the prefix to the end of the command.

The advertising router for Type 3 LSAs will be the last ABR that advertises the prefix. The metric within the Type 3 LSA uses the following logic:

- If the Type 3 LSA is created from a Type 1 LSA, it will be the total path metric to reach the originating router in the Type 1 LSA.
- If the Type 3 LSA is created from a Type 3 LSA from Area 0, it will be the total path metric to the ABR plus the metric in the original Type 3 LSA.

Example 7-12 shows the Type 3 LSA for the Area 56 prefix (10.56.1.0/24) on R4 for Area 0 and Area 1234. Notice that the advertising router and metric has changed depending on the area of the LSA.

### **Example 7-12** Detailed Output for OSPF Type 3 LSAs

[Click here to view code image](#)

```
R4#show ip ospf database summary 10.56.1.0
```

```
OSPF Router with ID (192.168.4.4) (Process ID 1)
```

```
Summary Net Link States (Area 0)
```

```
Routing Bit Set on this LSA in topology Base with MTID 0
```

```
LS age: 1335
```

```
Options: (No TOS-capability, DC, Upward)
```

```
LS Type: Summary Links(Network)
```

```
Link State ID: 10.56.1.0 (summary Network Number)
```

```
Advertising Router: 192.168.5.5
```

```
LS Seq Number: 8000000F
```

```
Checksum: 0xF97D
```

```
Length: 28
```

```
Network Mask: /24
```

```
MTID: 0
```

```
Metric: 1
```

```
Summary Net Link States (Area 1234)
```

```
LS age: 1296
```

```
Options: (No TOS-capability, DC, Upward)
```

```
LS Type: Summary Links(Network)
```

```
Link State ID: 10.56.1.0 (summary Network Number)
```

```
Advertising Router: 192.168.4.4
```

```
LS Seq Number: 8000000D
```

```
Checksum: 0x1565
```

```
Length: 28
```

```
Network Mask: /24
```

```
MTID: 0
```

```
Metric: 2
```

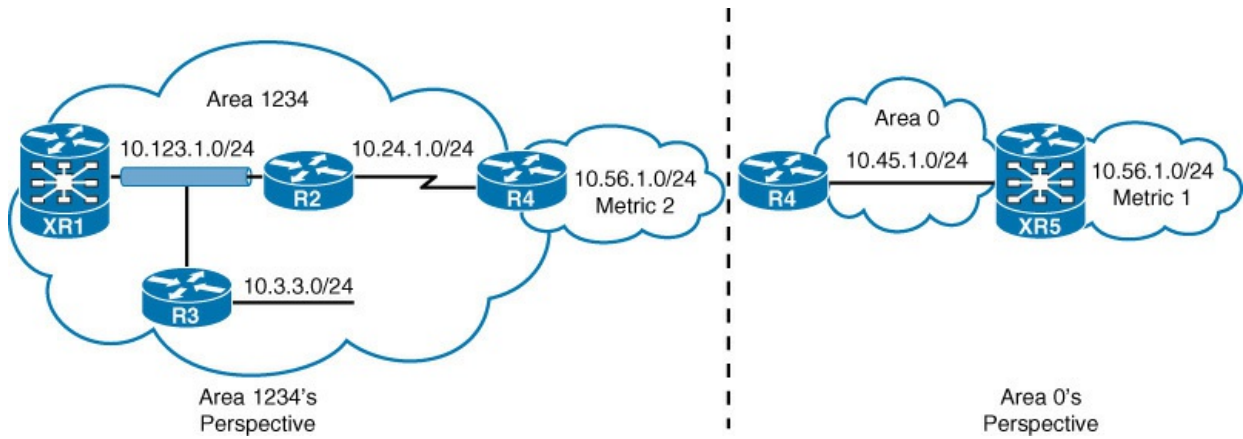
Table 7-3 provides an explanation of the fields within a Type 3 LSA.

Field	Description
Link ID	Network number
Advertising Router	RID of the router advertising the route (ABR)
Network Mask	Prefix length for the advertised network
Metric	Metric for the LSA as calculated above

Table 7-3 Type 3 LSA Fields

The Type 3 LSA contains the link-state ID (network number), subnet mask, IP address of the advertising ABR, and metric for the network prefix.

Figure 7-16 provides the perspective of a Type 3 LSA within an OSPF area. Routers within the area know how to reach the network prefix through the ABR and do not have visibility past the advertising ABR. Notice that the metric has changed on the Type 3 LSA advertised to Area 1234 when compared to the LSA advertised to Area 0.



**Figure 7-16** Visualization of Type 3 LSA from Area 0 and Area 1234

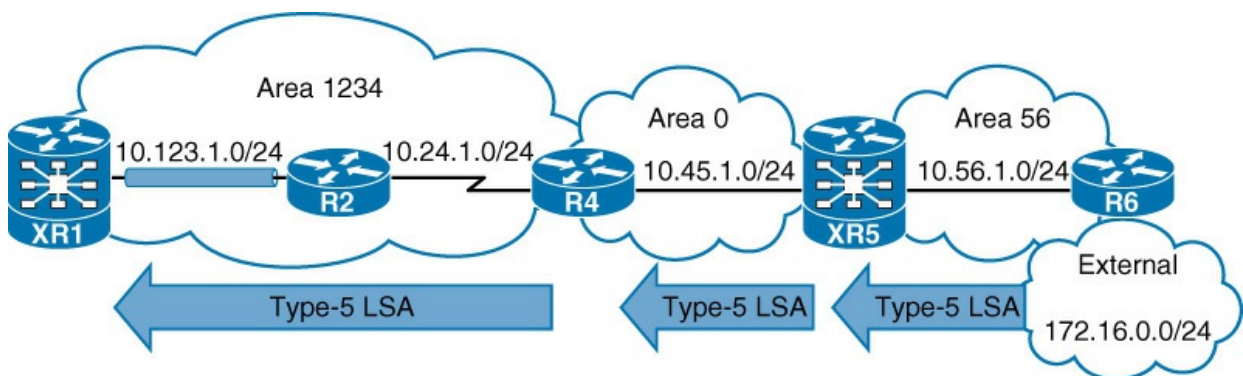
**Note**

An ABR will advertise only one Type 3 LSA for a prefix, even if it is aware of multiple paths from within its area (Type 1 LSAs) or from outside of its area (Type 3 LSAs). The metric for the best path will be used when the LSA is advertised.

**LSA Type 5: External Routes**

When a route is redistributed into OSPF on the ASBR, the external route is flooded throughout the entire OSPF domain as a Type 5 LSA. Type 5 LSAs are not associated to a specific area and are flooded throughout the OSPF domain. Only the LSA age is modified during flooding.

Figure 7-17 shows R6 redistributing the 172.16.0.0/24 static route into the OSPF domain.



**Figure 7-17** Concept of OSPF Type 5 LSA Flooding

A brief summary view of the Type 5 LSAs is shown as *Type 5 AS External*, as shown in Example 7-13. The link ID will be the external network number, and the advertising router is the RID for the router originating the Type 5 LSA. Notice that the Type 5 LSA is not associated with a specific OSPF area. This is because Type 5 LSAs are flooded throughout the OSPF routing domain by default.

**Example 7-13** Generic OSPF LSA Output for Type 5 LSAs

[Click here to view code image](#)

```
R6#show ip ospf database
```

```
! Output omitted for brevity
```

```
      Type-5 AS External Link States
```

Link ID	ADV Router	Age	Seq#	Checksum Tag
172.16.0.0	192.168.6.6	11	0x80000001	0x000866 0

Type 5 LSAs are shown in detail with the command **show ip ospf database external** for IOS nodes and with the command **show ospf database external** for IOS XR nodes. ABR routers modify only the LSA Age as the Type 5 LSA propagates through the OSPF domain.

Example 7-14 provides detailed output for the external OSPF LSAs in the OSPF domain. Notice that only the LS age is modified between the routers.

### **Example 7-14** *Detailed Output for OSPF Type 5 LSAs*

[Click here to view code image](#)

```
R6#show ip ospf database external
```

```
      OSPF Router with ID (192.168.6.6) (Process ID 1)
```

```
      Type-5 AS External Link States
```

```
LS age: 37
```

```
Options: (No TOS-capability, DC)
```

```
LS Type: AS External Link
```

```
Link State ID: 172.16.0.0 (External Network Number )
```

```
Advertising Router: 192.168.6.6
```

```
LS Seq Number: 80000001
```

```
Checksum: 0x866
```

```
Length: 36
```

```
Network Mask: /24
```

```
  Metric Type: 2 (Larger than any link state path)
```

```
  MTID: 0
```

```
  Metric: 20
```

```
  Forward Address: 0.0.0.0
```

```
  External Route Tag: 0
```

```
RP/0/0/CPU0:XR1#show ospf database external
```

```
OSPF Router with ID (192.168.1.1) (Process ID 1)
```

```
Type-5 AS External Link States
```

```
Routing Bit Set on this LSA
```

```
LS age: 65
```

```
Options: (No TOS-capability, DC)
```

```
LS Type: AS External Link
```

```
Link State ID: 172.16.0.0 (External Network Number)
```

```
Advertising Router: 192.168.6.6
```

```
LS Seq Number: 80000001
```

```
Checksum: 0x866
```

```
Length: 36
```

```
Network Mask: /24
```

```
  Metric Type: 2 (Larger than any link state path)
```

```
  TOS: 0
```

```
  Metric: 20
```

```
  Forward Address: 0.0.0.0
```

```
  External Route Tag: 0
```

Table 7-4 provides an explanation of the fields within a Type 5 LSA.

Field	Description
Link ID	External network number.
Network Mask	Subnet mask for the external network.
Advertising Router	RID of the router advertising the route (ASBR).
Metric Type	OSPF external metric type (an be a Type 1 O E1 or Type 2 O E2).
Metric	Metric upon redistribution.
External Route Tag	32-bit field that included with an external route. It is not used by OSPF itself; and can be used to communicate autonomous system boundaries or other relevant information to prevent routing loops.

Table 7-4 Type 5 LSA Fields

#### LSA Type 4: ASBR Summary

Type 4 LSAs locate the ASBR for Type 5 LSAs. A Type 5 LSA is flooded through the OSPF domain unmodified, and the only mechanism to identify the ASBR is the RID. Routers examine the Type 5 LSA, check to see whether the RID is in the local area, and if it is not, they require a mechanism to locate the ASBR.

Remember that the RID does not have to match an IP address on any OSPF router (including ASBRs). Only Type 1 or Type 2 LSAs provide a method to locate the RID within an area, as shown in Figure 7-14.

The Type 4 LSA provides a way for routers to locate the ASBR when the router is in a different area from the ASBR. A Type 4 LSA is a summary route strictly for the ASBR of a Type 5 LSA. The metric for a Type 4 LSA uses the following logic:



- When the Type 5 LSA crosses the first ABR, the ABR creates a Type 4 LSA with a metric set to the total path metric to the ASBR.
- When an ABR receives a Type 4 LSA from Area 0, the ABR creates a new Type 4 LSA with a metric set to the total path metric of the first ABR plus the metric in the original Type 4 LSA.

Figure 7-18 shows how the ABRs XR5 and R4 create Type 4 LSAs for the ASBR R6.

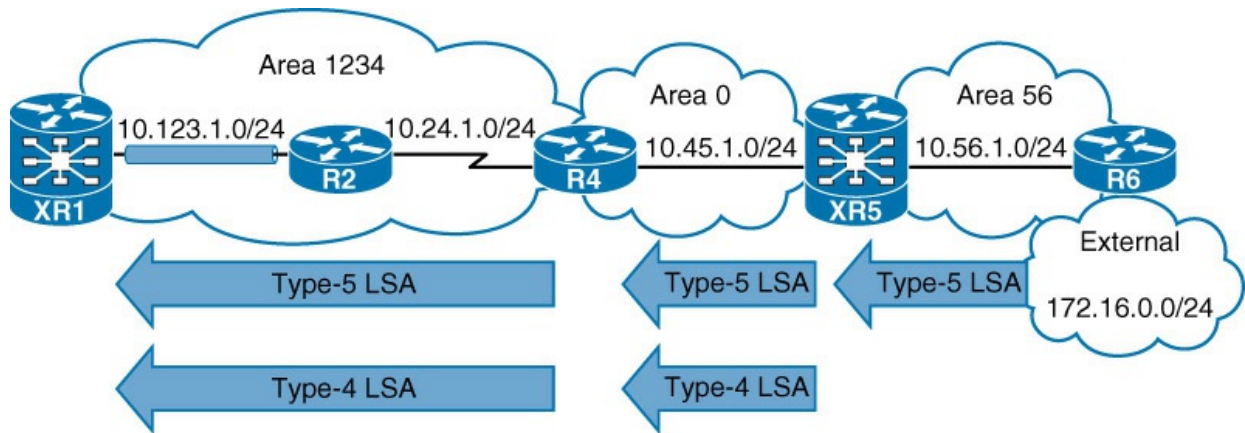


Figure 7-18 OSPF Type 4 and Type 5 LSA Flooding Within an OSPF Domain

A brief summary view of the Type 4 LSAs is shown in the LSDB under *Summary ASB Link States*, as shown in Example 7-15 for R4. Notice how the advertising router changes as the LSA crosses from ABR to ABR.

### Example 7-15 Generic OSPF LSA Output for Type 4 LSAs

[Click here to view code image](#)

```
R4#show ip ospf database
      OSPF Router with ID (192.168.4.4) (Process ID 1)
! Output omitted for brevity

      Summary ASB Link States (Area 0)

Link ID      ADV Router   Age         Seq#         Checksum
192.168.6.6  192.168.5.5  254        0x80000001  0x00084A

! Output omitted for brevity

      Summary ASB Link States (Area 1234)

Link ID      ADV Router   Age         Seq#         Checksum
192.168.6.6  192.168.4.4  253        0x80000001  0x001F34
```

To view the details of the Type 4 LSAs, the command **show ip ospf database asbr-summary** is used for IOS nodes, and the command **show ospf database asbr-summary** is used for IOS XR nodes.

Example 7-16 provides detailed output of the Type 4 LSA on R4. Notice that the metric and advertising router changes between the OSPF areas.

## Example 7-16 Detailed Output for Type 4 LSAs

[Click here to view code image](#)

```
R4#show ip ospf database asbr-summary
      OSPF Router with ID (192.168.4.4) (Process ID 1)

      Summary ASB Link States (Area 0)

Routing Bit Set on this LSA in topology Base with MTID 0
LS age: 321
Options: (No TOS-capability, DC, Upward)
LS Type: Summary Links(AS Boundary Router)
Link State ID: 192.168.6.6 (AS Boundary Router address)
Advertising Router: 192.168.5.5
LS Seq Number: 80000001
Checksum: 0x84A
Length: 28
Network Mask: /0
      MTID: 0      Metric: 1

      Summary ASB Link States (Area 1234)

LS age: 320
Options: (No TOS-capability, DC, Upward)
LS Type: Summary Links(AS Boundary Router)
Link State ID: 192.168.6.6 (AS Boundary Router address)
Advertising Router: 192.168.4.4
LS Seq Number: 80000001
Checksum: 0x1F34
Length: 28
Network Mask: /0
      MTID: 0      Metric: 2
```

Table 7-5 provides an explanation of the fields within a Type 4 LSA.

Field	Description
Link ID	ASBR RID
Advertising Router	RID of the router advertising the route (ABR)
Network Mask	This field always 0
Metric	Metric for the LSA as calculated above

**Table 7-5** Type 4 LSA Fields

### Note

An ABR advertises only one Type 4 LSA for every ASBR, even if the ASBR advertises thousands of Type 5 LSAs.

## LSA Type 7: NSSA External Summary

Later in this chapter, not-so-stubby areas (NSSAs) are explained as a method to reduce the LSDB within an area. A Type 7 LSA exists only in NSSA areas where the route redistribution is occurring.

An ASBR injects external routes as Type 7 LSAs in an NSSA area. The ABR does not advertise Type 7 LSAs outside of the originating NSSA area but converts the Type 7 LSA into Type 5 LSA for the other OSPF areas. If the Type 5 LSA crosses Area 0, the second ABR creates a Type 4 LSA for the Type 5 LSA.

Figure 7-19 shows Area 56 as an NSSA area and R6 redistributing the 172.16.0.0/24 prefix. The Type 7 LSA exists only in Area 56, XR5 creates a Type 5 LSA in Area 0, and R4 creates the Type 4 LSA for Area 1234.

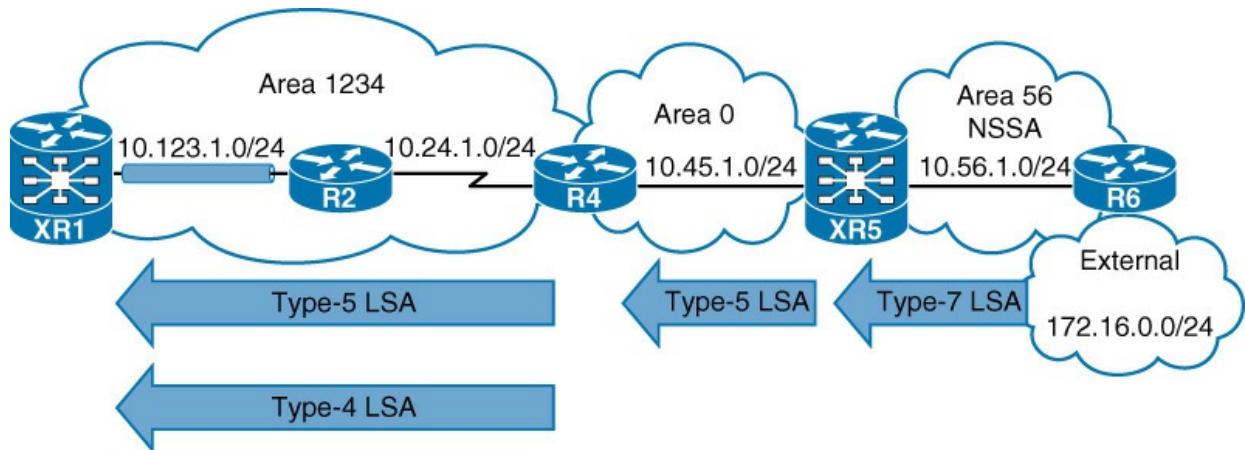


Figure 7-19 Concept of OSPF Type 5 LSA Flooding

A brief summary view of the Type 7 LSAs is under *Type 7 AS External*, as shown in Example 7-17. Type 7 LSAs are present only in the OSPF NSSA area where redistribution is occurring.

### Example 7-17 Generic OSPF LSA Output for Type 7 LSAs

[Click here to view code image](#)

```
RP/0/0/CPU0:XR5#show ospf database
      OSPF Router with ID (192.168.5.5) (Process ID 1)

! Output omitted for brevity
      Type-7 AS External Link States (Area 56)

Link ID          ADV Router      Age           Seq#           Checksum Tag
172.16.0.0      192.168.6.6    46           0x80000001    0x00e535 0

! Notice that no Type-4 LSA has been generated. Only the Type-7 LSA for Area 56
! and the Type-5 LSA for the other areas. R5 advertises the Type-5 LSA

      Type-5 AS External Link States

Link ID          ADV Router      Age           Seq#           Checksum Tag
172.16.0.0      192.168.5.5    28           0x80000001    0x00879f 0
```

To see the specific Type 7 LSA details, the command **show ip ospf database nssa-external** is used for IOS nodes, and the command **show ospf database nssa-external** is used for IOS XR nodes. Example 7-18 shows sample output from XR5.

## Example 7-18 Detailed Output for OSPF Type 7 LSAs

[Click here to view code image](#)

```
RP/0/0/CPU0:XR5#show ospf database nssa-external

      OSPF Router with ID (192.168.5.5) (Process ID 1)

      Type-7 AS External Link States (Area 56)

Routing Bit Set on this LSA
LS age: 69
Options: (No TOS-capability, Type 7/5 translation, DC)
LS Type: AS External Link
Link State ID: 172.16.0.0 (External Network Number)
Advertising Router: 192.168.6.6
LS Seq Number: 80000001
Checksum: 0xe535
Length: 36
Network Mask: /24
    Metric Type: 2 (Larger than any link state path)
    TOS: 0
    Metric: 20
    Forward Address: 10.56.1.6
    External Route Tag: 0
```

Table 7-6 provides an explanation of the fields within a Type 7 LSA.

Field	Description
Link ID	External network number.
Network Mask	Subnet mask for the external network.
Advertising Router	RID of the router advertising the route (ASBR).
Metric Type	OSPF external metric type (can be a Type 1 O N1 or Type 2 O N2).
Metric	Metric upon redistribution.
External Route Tag	32-bit field that included with an external route. It is not used by OSPF itself and can be used to communicate autonomous system boundaries or other relevant information to prevent routing loops.

Table 7-6 Type 7 LSA Fields

### LSA Type Summary

The OSPF LSA types can be difficult to understand but are very important when troubleshooting a router's behavior for a specific prefix. Table 7-7 provides a summary of the OSPF LSAs discussed.

LSA Type	Description
1	Router link
2	Network link
3	Summary link
4	ASBR summary
5	AS external
6	Multicast group membership (deprecated)
7	NSSA external
9–11	Opaque LSAs (extensions in OSPF like MPLS Traffic Engineering) (Opaque LSAs are beyond the scope of this book.)

Table 7-7 OSPF LSA Types

## OSPF PATH SELECTION

OSPF executes Dijkstra's Shortest Path First (SPF) algorithm to create a loop-free topology of shortest paths. All routers use the same logic to calculate the shortest-path for each network. Path selection prioritizes paths by using the following logic:

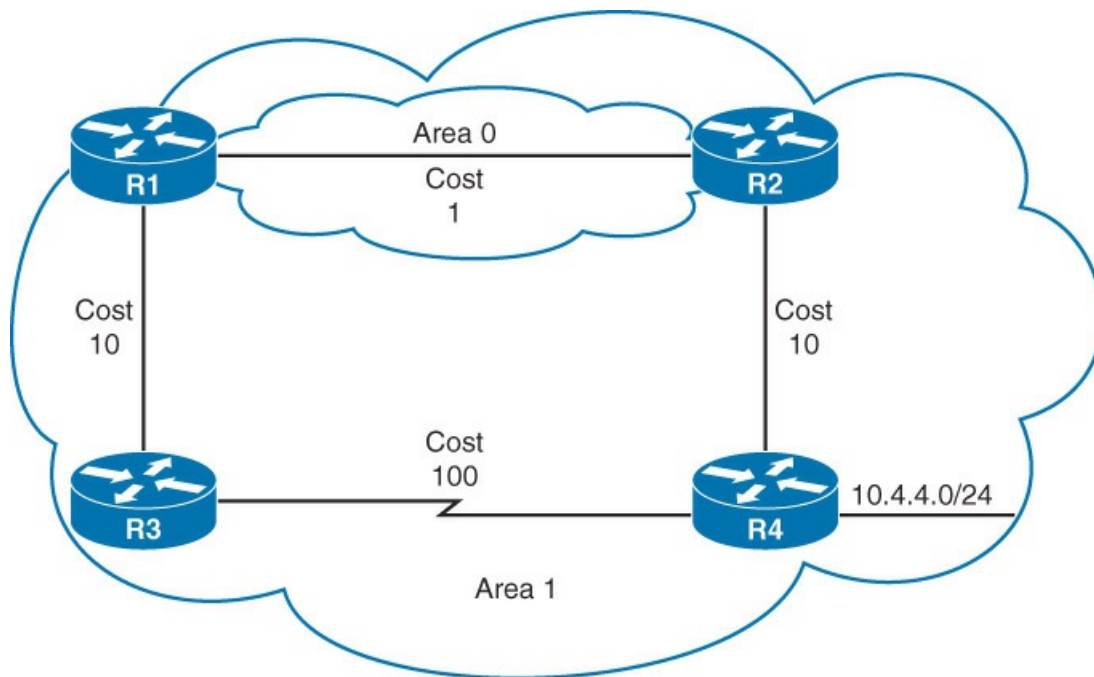
1. Intra-area
2. Interarea
3. External Type 1
4. External Type 2

The following sections explain each component in detail.

### Intra-Area Routes

Routes advertised via a Type 1 LSA for an area are always preferred over Type 3 and Type 5 LSAs. If multiple intra-area routes exist, the path with the lowest total path metric is installed in the Routing Information Base (RIB). If there is a tie in metric, both routes install into the RIB.

In [Figure 7-20](#), R1 is computing the route to 10.4.4.0/24. Instead of taking the faster Ethernet connection (R1-R2-R4), R1 will take the path across the slower serial link (R1-R3-R4) to R4 because that is the intra-area path.



**Figure 7-20** Intra-Area Routes over Interarea Routes

**Example 7-19** displays R1's routing table entry for the 10.4.4.0/24 network. Notice that the metric is 111 and that the intra-area path was selected over the interarea path with the lower total path metric.

**Example 7-19** R1's Route Table for the 10.4.4.0/24 Network

[Click here to view code image](#)

```
R1#show ip route 10.4.4.0
Routing entry for 10.4.4.0/24
  Known via "ospf 1", distance 110, metric 111, type intra area
  Last update from 10.13.1.3 on GigabitEthernet0/1, 00:00:42 ago
  Routing Descriptor Blocks:
    * 10.13.1.3, from 10.34.1.4, 00:00:42 ago, via GigabitEthernet0/1
      Route metric is 111, traffic share count is 1
```

**Interarea Routes**

Interarea routes take the lowest total path metric to the destination. If there is a tie in metric, both routes install into the RIB. All interarea paths for a route must go through Area 0.

In **Figure 7-21**, R1 is computing the path to R6. R1 uses the path R1-R3-R5-R6 because its total path metric is 35 versus the R1-R2-R4-R6 path with a metric of 40.

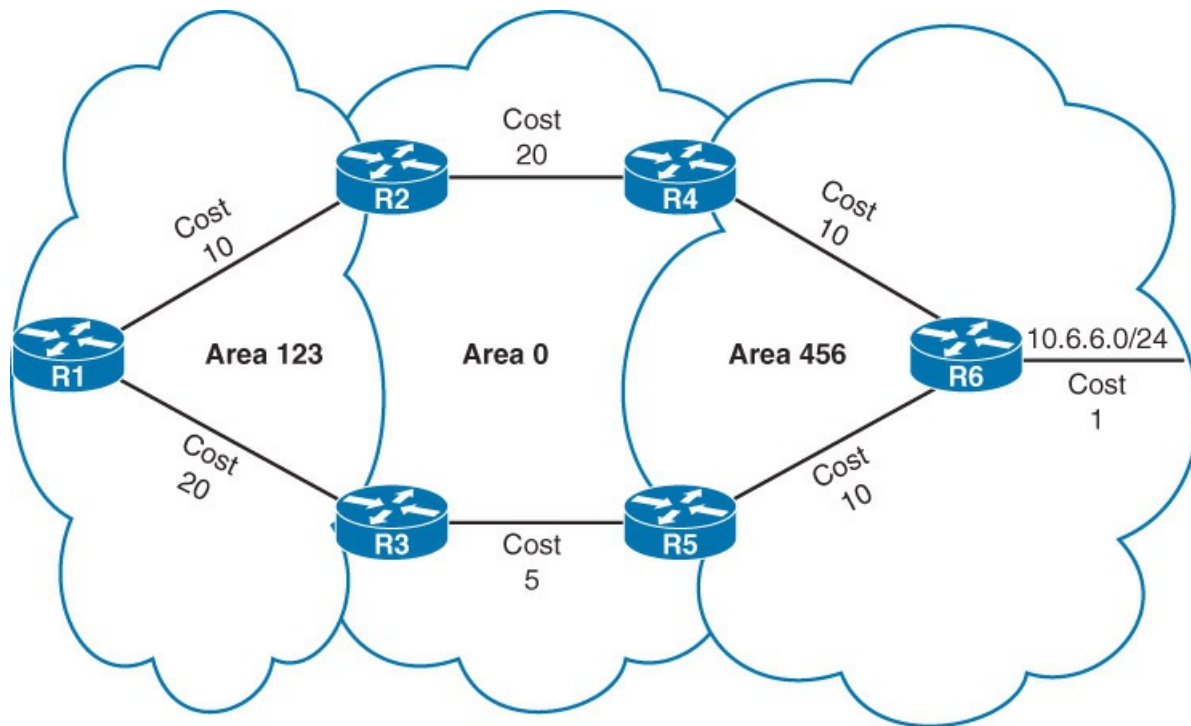


Figure 7-21 Interarea Route Selection

### External Route Selection

Earlier in this chapter, external routes were briefly explained as Type 1 or Type 2. The main differences between Type 1 and Type 2 external OSPF routes are as follows:

- Type 1 routes are preferred over Type 2.
- The Type 1 path metric equals the Redistribution Metric + Total Path Metric to reach the ASBR.
- The Type 2 path metrics equals only the redistribution metric.

### E1 and N1 External Routes

External OSPF Type 1 route calculation uses the Redistribution Metric + The Lowest Path Metric to reach the ASBR that advertised the network. Type 1 path metrics will be lower for routers closer to the originating ASBR; whereas the path metric will be higher for a router ten hops away from the ASBR.

If there is a tie in the path metric, both routes install into the RIB. If the ASBR is in a different area, then the path of the traffic must go through Area 0. An ASBR router will not install an O E1 and O N1 route into the RIB at the same time. O N1 is always given preference and will prevent the O E1 from installing on the ASBR.

### E2 and N2 External Routes

External OSPF Type 2 routes do not increment in metric irrespective of the path metric to the ASBR. If there is a tie in the redistribution metric, then the router compares the forwarding cost. The forwarding cost is the metric to the ASBR that advertised the network, and the lower forwarding cost is preferred. If there is a tie in forwarding cost, then both routes install into the routing table. An ASBR router will not install an O E2 and O N2 route into the RIB at the same time. O N2 is always given preference and will prevent the O E2 from installing on the ASBR.

Figure 7-22 shows the topology for R1 computing a path to the external network (172.16.0.0/24) that is being redistributed.



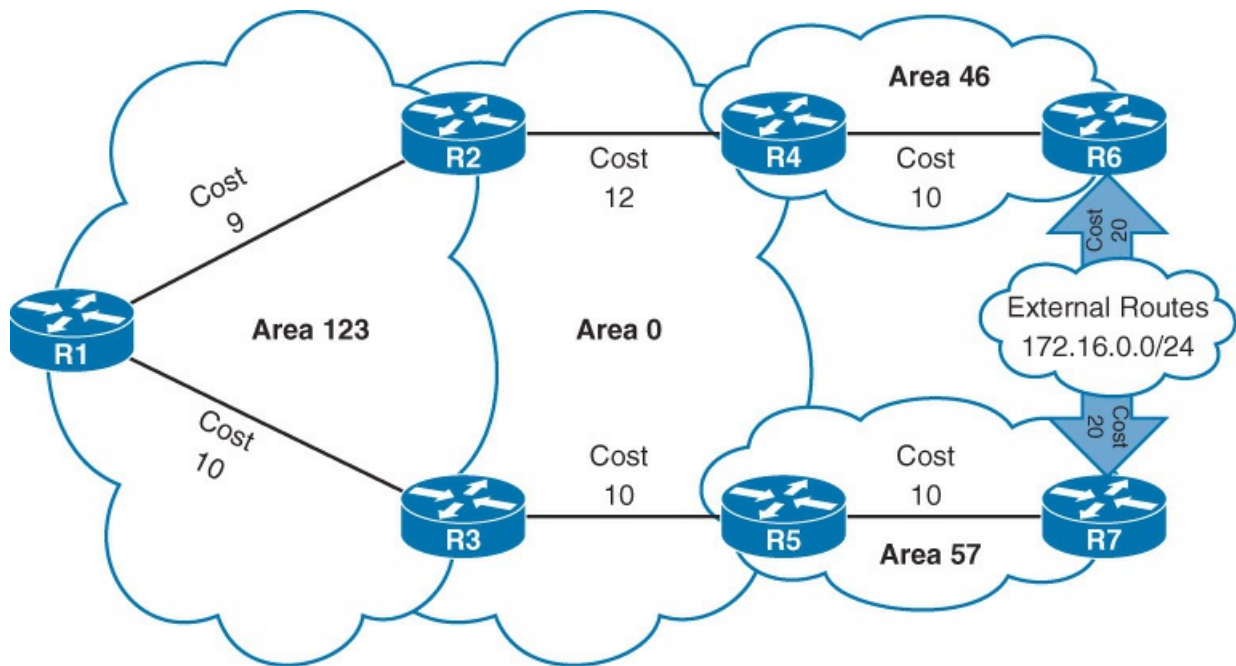


Figure 7-22 External Type 2 Route Selection Topology

The path R1-R2-R4-R6 has a metric of 20, which ties with the path R1-R3-R5-R7. The forwarding metric of the R1-R2-R4-R6 path is 31, and the forwarding metric of the R1-R3-R5-R7 path is 30. R1 installs the R1-R3-R5-R7 path into the routing table.

Example 7-20 shows R1's metric and forwarding metric to the 172.16.0.0/24 network.

### Example 7-20 OSPF Forwarding Metric

[Click here to view code image](#)

```
R1#show ip route 172.16.0.0
Routing entry for 172.16.0.0/24
  Known via "ospf 1", distance 110, metric 20, type extern 2, forward metric 30
  Last update from 10.13.1.3 on GigabitEthernet0/1, 00:12:40 ago
  Routing Descriptor Blocks:
    * 10.13.1.3, from 192.168.7.7, 00:12:40 ago, via GigabitEthernet0/1
      Route metric is 20, traffic share count is 1
```

### Equal Cost Multi-Path

If OSPF identifies multiple paths in the algorithms from above, then those routes will be installed into the routing table as Equal Cost Multi-Path (ECMP) routing. The default maximum ECMP paths for IOS nodes are 4 routes and 16 routes for IOS XR nodes.

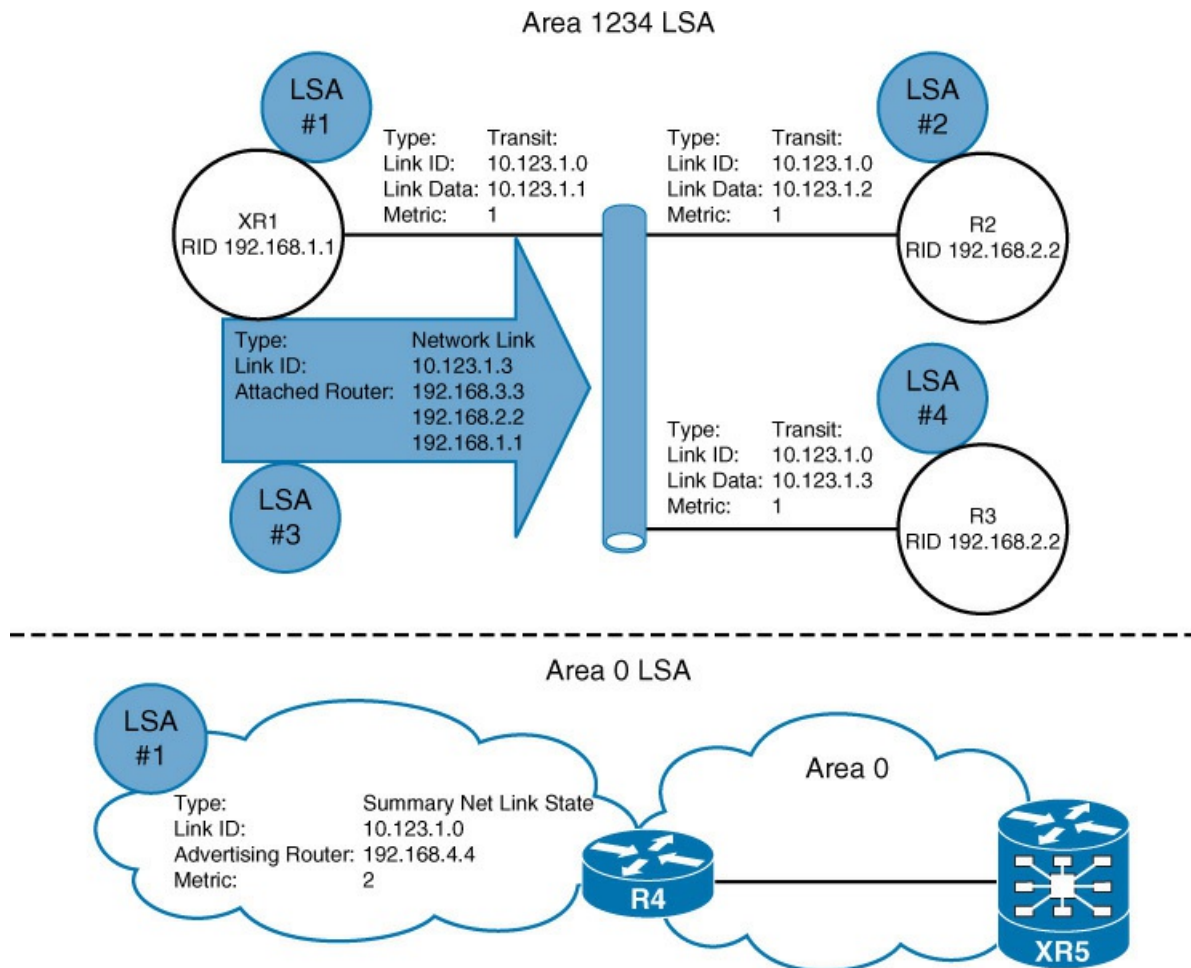
The default ECMP setting can be over-written with the command `maximum-paths` under the OSPF process to increase the defaults to 16 for IOS and 32 for IOS XR.

## SUMMARIZATION OF ROUTES

Route scalability is a large factor for the IGP routing protocols used by service providers because there can be thousands of routers running in the network. Splitting up an OSPF routing domain into multiple areas reduces the size of the LSDB per area. While the number of routers and networks remains the same within the OSPF routing domain, the number of detailed Type 1 and Type 2 LSAs are exchanged for simpler Type 3 LSAs.



For example, in [Figure 7-13](#) for Area 1234, there are three Type 1 LSAs and one Type 2 LSA for the 10.123.1.0/24 network. Those four LSAs become one Type 3 LSA outside of Area 1234. [Figure 7-23](#) illustrates the reduction of LSAs through area segmentation for the 10.123.1.0/24 network.



**Figure 7-23** LSA Reduction Through Area Segmentation

Another method of shrinking the LSDB involves summarizing network prefixes. Newer routers have more memory and faster processors than those in the past, but because all routers have an identical copy of the LSDB, the OSPF area needs to accommodate the smallest and slowest router in that area.

Summarization of routes also helps SPF calculations run faster. A router that has 10,000 network entries will take longer to run the SPF calculation than a router with 500 network entries. Because all routers within an area must maintain an identical copy of the LSDB, summarization occurs between areas on the ABRs.

Summarization can eliminate the SPF calculation outside the area for the summarized prefixes because it hides them outside the area. [Figure 7-24](#) provides a simple network topology; if the 10.1.12.0/24 link fails, all routers in Area 1 will have to run SPF calculations. The Type 3 LSA metrics will change for the 10.1.13.0/24, and 10.1.34.0/24 network prefixes and routers outside of Area 1 will need to run SPF calculations for just those prefixes.

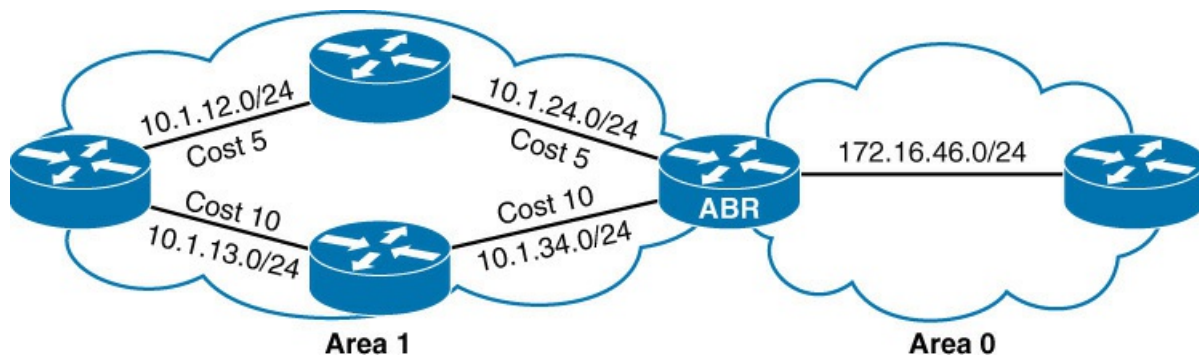


Figure 7-24 Topology Example

Figure 7-25 shows the networks in Area 1 being summarized at the ABR for the 10.1.0.0/18 network. If the 10.1.12.0/24 link fails, all the routers in Area 1 would still run the SPF calculation, but routers in Area 0 would not be affected because the link 10.1.13.0/24 and 10.1.34.0/24 networks are not know outside of Area 1.

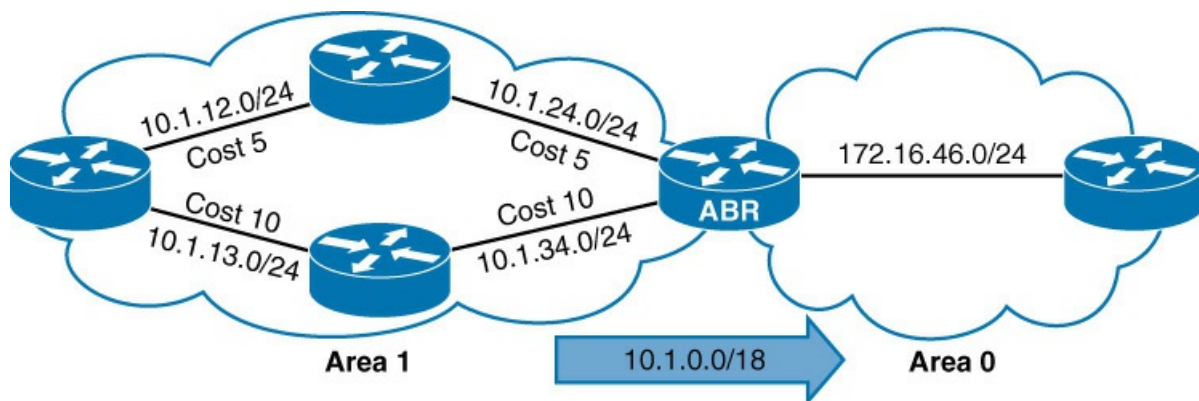


Figure 7-25 Topology Example with Summarization

This concept applies to networks of varying sizes but is beneficial for networks with a carefully developed IP addressing scheme and proper summarization. The following sections will explain summarization in more detail.

### Interarea Summarization

Interarea summarization reduces the number of Type 3 LSAs that an ABR advertises into an area when it receives Type 1 LSAs. The network summarization range is associated with a specific source area for Type 1 LSAs.

When a Type 1 LSA within the summarization range reaches the ABR from the source area, the ABR creates a Type 3 LSA for the summarized network range. The ABR suppresses the more specific Type 3 LSAs and reducing multiple Type 3 LSAs from being generated. Interarea summarization does not impact the Type 1 LSAs within the source area.

Figure 7-26 illustrates the concept as the 172.16.1.0/24 – 172.16.15.0/24 Type 1 LSAs summarize into one Type 3 LSA as the 172.16.0.0/24 network.

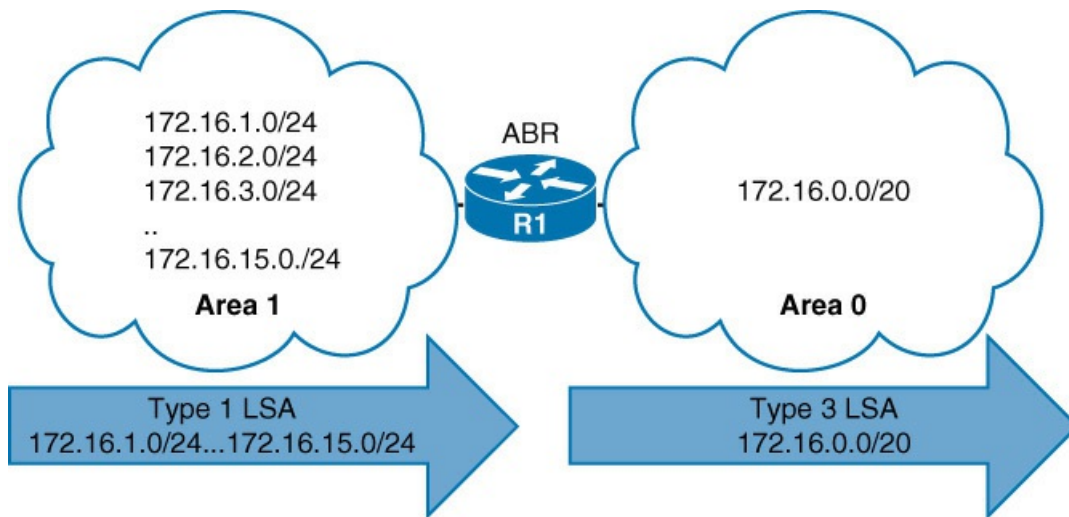


Figure 7-26 OSPF Interarea Summarization Concept

Summarization works only on Type 1 LSAs and normally occurs on nonbackbone areas so that the summarized routes advertise into the backbone.

In Figure 7-27, a junior network engineer is trying to reduce the number of network prefixes in Area 2 and has placed a 172.16.0.0/15 summary route on R2 for Area 0. Notice that R2 is not summarizing Area 1's prefixes as they enter into Area 2. Interarea summarization did not occur because the LSAs in Area 0 are Type 3 and not Type 1 LSAs. Placing interarea summarization on R1 for Area 1 will provide the intended results.

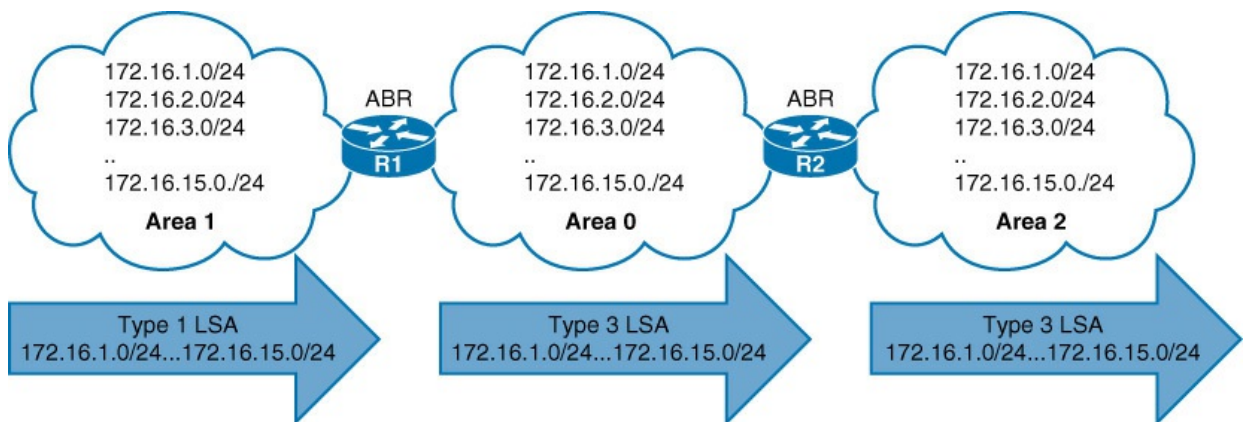


Figure 7-27 Interarea Summarization Failure

The default metric for a summary LSA is the smallest metric associated with a LSA; however, it can be set as part of the configuration. In Figure 7-25, R1 summarizes three prefixes with various path costs. The 172.16.3.0/24 prefix has the lowest metric so that metric will be used for the summarized route. Figure 7-28 demonstrates the metric selection for a summary prefix.

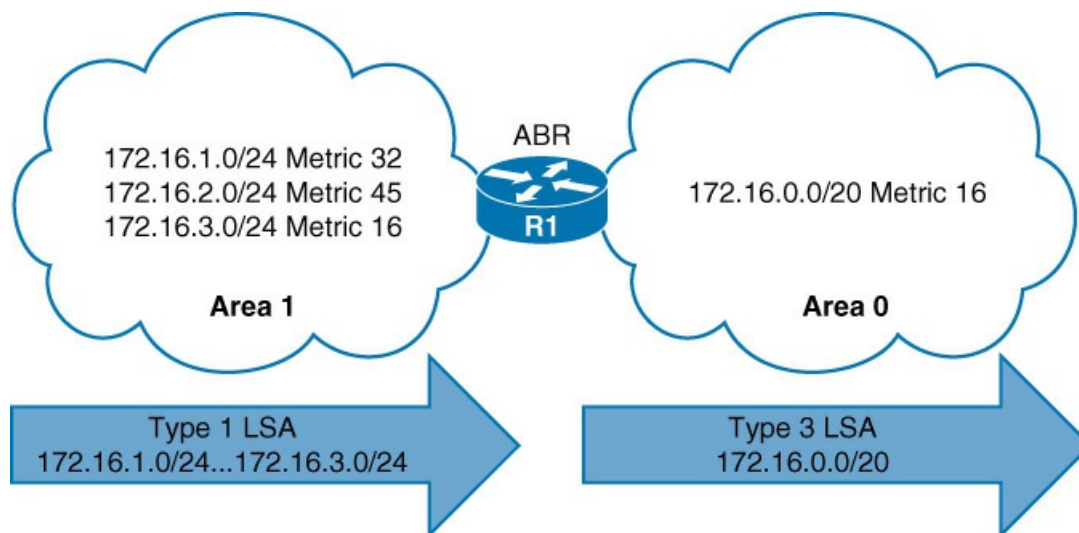


Figure 7-28 Interarea Summarization Metric

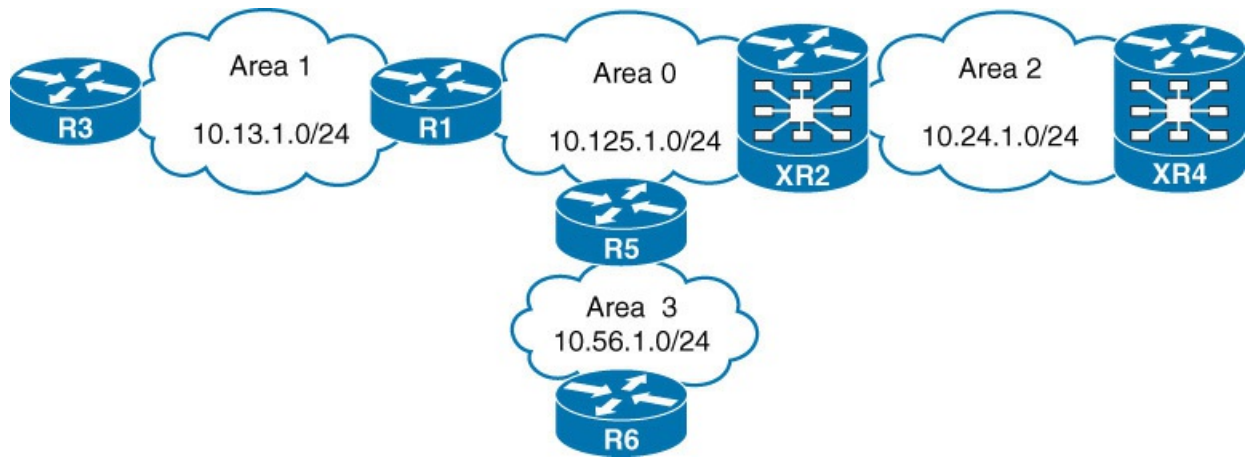
OSPF behaves identically to Enhanced Interior Gateway Routing Protocol (EIGRP) and will check every prefix within the summarization range when a matching Type 1 LSA is added or removed. If a lower metric is available, the summary LSA is advertised with the new metric; and if the lowest metric is removed, a new metric is identified, and a new summary LSA is advertised.

Defining the summarization range and associated area is performed on IOS nodes with the command **area** *area-id ip-address subnet-mask* [**advertise** | **not-advertise**] [**cost** *metric*]. IOS XR nodes use the command **range** *ip-address subnet-mask* {**advertise** | **not-advertise**} under the source area to define the summarization range nodes.

On IOS, the default behavior is to advertise the summary prefix so the keyword **advertise** is not necessary. The option **not-advertise** can be used in place of **advertise** on either IOS or IOS XR, which will not create any Type 3 LSAs for any networks in that range, thus making the route visible only within the area it originates.

Appending the **cost** *metric* keyword to the command statically sets the metric on the summary route. IOS XR does not support statically setting the metric for a summary route.

Figure 7-29 provides a topology example where R1 will summarize the 10.13.1.0/24 to a 10.0.0.0/12 network and where XR2 will summarize the 10.24.1.0/24 network to a 10.24.0.0/13 network.



**Figure 7-29** OSPF Interarea Summarization Example

**Example 7-21** displays the routing table on R6 before summarization. Notice that the 10.13.1.0/24 network from Area 1 is present, and that the 10.24.1.0/24 network from Area 2 is present.

**Example 7-21** *Route Table Before OSPF Interarea Route Summarization*

[Click here to view code image](#)

```
R6#show ip route ospf
! Output omitted for brevity

10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
O IA 10.13.1.0/24 [110/3] via 10.56.1.5, 00:00:33, GigabitEthernet0/1
O IA 10.24.1.0/24 [110/3] via 10.56.1.5, 00:00:27, GigabitEthernet0/1
O IA 10.125.1.0/24 [110/2] via 10.56.1.5, 00:00:33, GigabitEthernet0/1
```

**Example 7-22** provides R1's and XR's configuration for interarea summarization.

**Example 7-22** *Interarea Route Summarization Configuration*

[Click here to view code image](#)

```
R1
router ospf 1
router-id 192.168.1.1
area 1 range 10.0.0.0 255.240.0.0
network 10.13.1.1 0.0.0.0 area 1
network 10.125.1.1 0.0.0.0 area 0
```

**XR2**

```
router ospf 1
  area 0
    interface GigabitEthernet0/0/0/2
    !
  !
  area 2
    range 10.24.0.0/13 advertise
    interface GigabitEthernet0/0/0/3
```

[Example 7-23](#) displays R6's routing table with interarea route summarization on R1 and XR2. Notice the 10.0.0.0/12 and the 10.24.0.0/13 networks.

**Example 7-23** *Route Table After OSPF Interarea Route Summarization*

[Click here to view code image](#)

```
R6#show ip route ospf
! Output omitted for brevity

10.0.0.0/8 is variably subnetted, 5 subnets, 4 masks
O IA 10.0.0.0/12 [110/3] via 10.56.1.5, 00:04:43, GigabitEthernet0/1
O IA 10.24.0.0/13 [110/3] via 10.56.1.5, 00:01:27, GigabitEthernet0/1
O IA 10.125.1.0/24 [110/2] via 10.56.1.5, 00:08:22, GigabitEthernet0/1
```

The ABR performing interarea summarization will install discard routes, a route to the Null0 interface that matches the summarized network range. Discard routes prevent routing loops where portions of the summarized network range do not have a more specific route in the RIB. This is similar to the static route to Null0 in [Chapter 4, "Static Routing."](#)

[Example 7-24](#) displays the route to Null0 on R1 and XR2.

**Example 7-24** *R1 and XR2 Loop-Prevention Routes*

[Click here to view code image](#)

```
R1#show ip route ospf
! Output omitted for brevity

10.0.0.0/8 is variably subnetted, 7 subnets, 4 masks
O 10.0.0.0/12 is a summary, 00:02:55, Null0
O IA 10.24.0.0/13 [110/11] via 10.125.1.2, 00:02:45, GigabitEthernet0/0
O IA 10.56.1.0/24 [110/20] via 10.125.1.5, 00:02:45, GigabitEthernet0/0
RP/0/0/CPU0:
```

**XR2#show route ospf**

```
O IA 10.0.0.0/12 [110/11] via 10.125.1.1, 00:03:12, GigabitEthernet0/0/0/2
O IA 10.24.0.0/13 [254/0] via 0.0.0.0, 00:03:12, Null0
O IA 10.56.1.0/24 [110/11] via 10.125.1.5, 00:03:12, GigabitEthernet0/0/0/2
```



Note

IOS sets the AD to 110 for internal discard routes and 254 for external discard routes. IOS XR sets the AD to 254 for all discard routes, allowing another protocol to preempt the network range if present.

## External Summarization

During OSPF redistribution, external routes advertise into the OSPF domain as Type 5 or Type 7 LSAs (NSSA). External summarization is a feature that reduces the number of external LSAs in an OSPF domain. An external network summarization range is configured on the ASBR router and network prefixes that match the network range do not generate a Type 5/Type 7 LSA for the specific prefix. Instead, a Type 5/Type 7 LSA with the external network summarization range is created.

Note

ABRs for NSSA areas act as an ASBR when the Type 7 LSAs is converted to a Type 5 LSA. External summarization can be performed only on ABRs when they match this scenario.

Figure 7-30 demonstrates the concept with the external network summarization range 172.16.0.0/20 configured on the ASBR. The ABR creates only one Type 5/Type 7 LSA in Area 1 when EIGRP redistributes routes into OSPF.

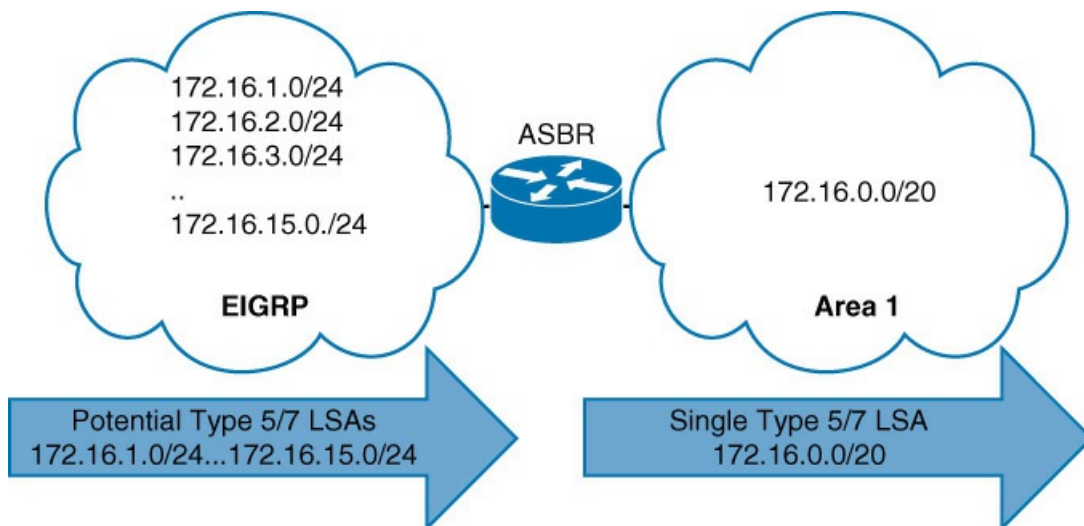


Figure 7-30 External Summarization Concept

Configuring external summarization uses the command **summary-address** *ip-address subnet-mask* for on IOS nodes and the command **summary-prefix** *ip-address subnet-mask* for IOS XR nodes.

Figure 7-31 provides a topology with external routes redistributed on XR4 and R6.

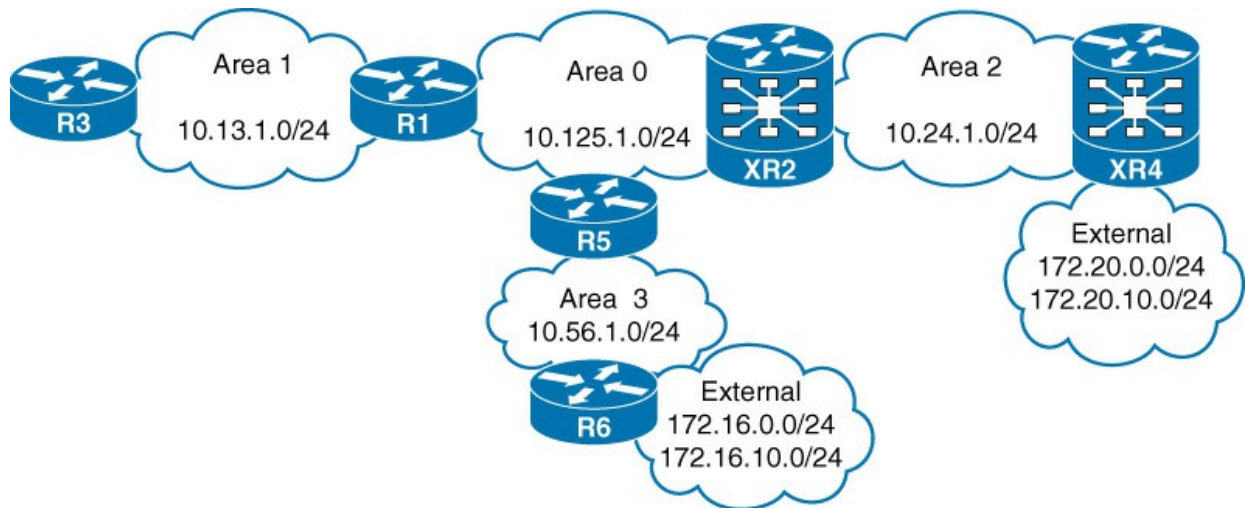


Figure 7-31 OSPF External Summarization

Example 7-25 provides the route table on R1 before external route summarization.

### Example 7-25 Route Table Before External Summarization

[Click here to view code image](#)

```
R1#show ip route ospf
! Output omitted for brevity

10.0.0.0/8 is variably subnetted, 7 subnets, 3 masks
O    10.0.0.0/12 is a summary, 00:03:44, Null0
O IA  10.24.1.0/24 [110/11] via 10.125.1.2, 00:03:44, GigabitEthernet0/0
O IA  10.56.1.0/24 [110/20] via 10.125.1.5, 00:03:44, GigabitEthernet0/0
172.16.0.0/24 is subnetted, 2 subnets
O E2  172.16.0.0/24 [110/20] via 10.125.1.5, 00:01:19, GigabitEthernet0/0
O E2  172.16.10.0/24 [110/20] via 10.125.1.5, 00:01:19, GigabitEthernet0/0
172.20.0.0/24 is subnetted, 2 subnets
O E2  172.20.0.0/24 [110/20] via 10.125.1.2, 00:00:12, GigabitEthernet0/0
O E2  172.20.10.0/24 [110/20] via 10.125.1.2, 00:00:12, GigabitEthernet0/0
```

XR4 will redistribute and summarize multiple routes within the 172.20.0.0 address space to a 172.20.0.0/14 network range. R6 will redistribute and summarize multiple routes within the 172.16.0.0 address space to a 172.16.0.0/14 network range. Example 7-26 provides the relevant configuration.

### Example 7-26 OSPF External Summarization Configuration

[Click here to view code image](#)

```
XR4
router ospf 1
router-id 192.168.4.4
summary-prefix 172.20.0.0/14
redistribute static
area 2
interface GigabitEthernet0/0/0/2
```



## R6

```
router ospf 1
router-id 192.168.6.6
summary-address 172.16.0.0 255.252.0.0
redistribute static subnets
network 0.0.0.0 255.255.255.255 area 3
```

[Example 7-27](#) provides the route table of R1 verifying that XR4 and R6 summarized the external networks to the 172.16.0.0/14 and 172.20.0.0/14 ranges.

### Example 7-27 Route Table After External Summarization

[Click here to view code image](#)

```
R1#show ip route ospf
! Output omitted for brevity

10.0.0.0/8 is variably subnetted, 7 subnets, 3 masks
O      10.0.0.0/12 is a summary, 00:07:18, Null0
O IA   10.24.1.0/24 [110/11] via 10.125.1.2, 00:07:18, GigabitEthernet0/0
O IA   10.56.1.0/24 [110/20] via 10.125.1.5, 00:07:18, GigabitEthernet0/0
O E2   172.20.0.0/14 [110/20] via 10.125.1.2, 00:01:33, GigabitEthernet0/0
O E2   172.16.0.0/14 [110/20] via 10.125.1.5, 00:02:44, GigabitEthernet0/0
```

The summarizing ASBR installs a route to Null0 that matches the summarized network range as part of a loop- prevention mechanism. [Example 7-28](#) provides the routing table of XR4 and R6 with the Null0 route.

### Example 7-28 XR4 and R6 Loop-Prevention Routes

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show route ospf

O IA 10.0.0.0/12 [110/12] via 10.24.1.2, 00:08:01, GigabitEthernet0/0/0/0
O IA 10.56.1.0/24 [110/12] via 10.24.1.2, 00:08:01, GigabitEthernet0/0/0/0
O IA 10.125.1.0/24 [110/2] via 10.24.1.2, 00:08:01, GigabitEthernet0/0/0/0
O E2 172.16.0.0/14 [110/20] via 10.24.1.2, 00:01:20, GigabitEthernet0/0/0/0
O E2 172.20.0.0/14 [254/0] via 0.0.0.0, 00:02:12, Null0
```

## R6#show ip route ospf

```
! Output omitted for brevity

10.0.0.0/8 is variably subnetted, 5 subnets, 3 masks
O IA   10.0.0.0/12 [110/30] via 10.56.1.5, 00:36:38, GigabitEthernet0/1
O IA   10.24.1.0/24 [110/21] via 10.56.1.5, 00:07:14, GigabitEthernet0/1
O IA   10.125.1.0/24 [110/20] via 10.56.1.5, 00:36:38, GigabitEthernet0/1
O      172.16.0.0/14 is a summary, 00:01:47, Null0
O E2   172.20.0.0/14 [110/20] via 10.56.1.5, 00:02:38, GigabitEthernet0/1
```

## Default Route

OSPF supports advertising the default route into the OSPF domain. The advertising router must have a default route in its routing table for the default route to be advertised. IOS and IOS XR nodes use the following command syntax under the global OSPF process:

[Click here to view code image](#)

```
default-information originate [always] [metric metric-value]
[metric-type type-value]
```

Figure 7-32 provides a simple topology of two OSPF domains. XR1 will advertise a default route into Area 12, and R3 will advertise a default route into Area 34.

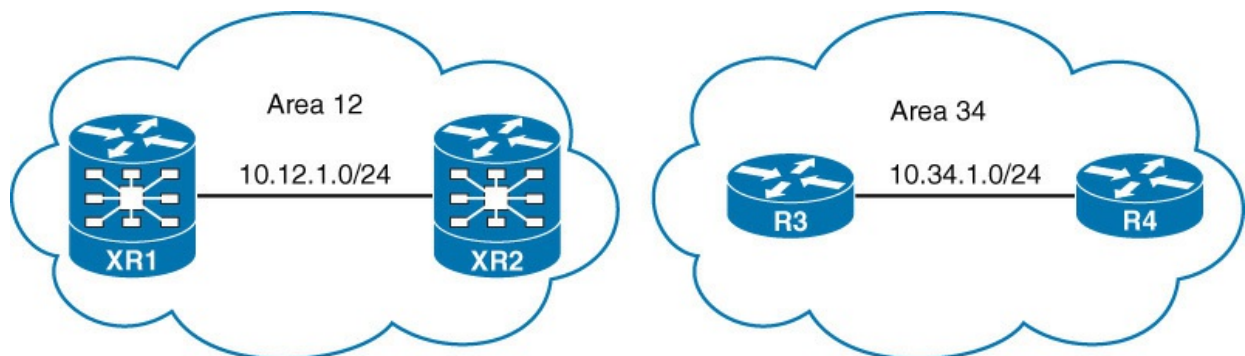


Figure 7-32 Two OSPF Domains

Example 7-29 provides the relevant configuration. Notice that XR1 and R3 use a default static route to Null0 to satisfy the requirement of having the default route in the RIB.

### Example 7-29 OSPF Default Information Originate Configuration

[Click here to view code image](#)

```
XR1
router static
  address-family ipv4 unicast
    0.0.0.0/0 Null0
  !
!
router ospf 1
  default-information originate
  area 12
  interface GigabitEthernet0/0/0/0
```

```
R3
ip route 0.0.0.0 0.0.0.0 null 0

router ospf 1
  default-information originate
  Network 0.0.0.0 255.255.255.255 area 34
```

Example 7-30 provides the routing table of XR2 and R4. Notice that OSPF advertises the default

route as an external OSPF route.

### Example 7-30 XR2 and R4's Routing Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route
! Output omitted for brevity
Gateway of last resort is 10.12.1.1 to network 0.0.0.0

O*E2 0.0.0.0/0 [110/1] via 10.12.1.1, 00:03:35, GigabitEthernet0/0/0/0
C    10.12.1.0/24 is directly connected, 00:14:14, GigabitEthernet0/0/0/0
```

```
R4#show ip route
! Output omitted for brevity
Gateway of last resort is 10.34.1.3 to network 0.0.0.0

O*2 E 0.0.0.0/0 [110/1] via 10.34.1.3, 00:03:25, GigabitEthernet0/0
      10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C    10.34.1.0/24 is directly connected, GigabitEthernet0/0
```

The **always** option will advertise a default route even if a default route does not exist in the RIB. In addition, the route metric can be changed with the **metric** *metric-value* option, and the metric type can be changed with the **metric-type** *type-value* options.

Example 7-31 provides the configuration of XR1 and R3 with the **always** keyword, so a static default route is no longer needed. XR1 sets the metric to 60 with a metric-type of 2, and R3 sets the metric to 40 with a metric-type of 1.

### Example 7-31 OSPF Default Information Originate Configuration

[Click here to view code image](#)

```
XR1
router ospf 1
router-id 192.168.1.1
default-information originate always metric 60 metric-type 2
area 12
interface GigabitEthernet0/0/0/0
```

```
R3
router ospf 1
router-id 192.168.3.3
network 10.34.1.0 0.0.0.255 area 34
default-information originate always metric 40 metric-type 1
```

Example 7-32 provides the routing table of XR2 and R4. Notice that XR2 sets the default route as an O E2 with a metric of 60, and R4 sets the default route as an O E1 with a metric of 41.

## Example 7-32 OSPF Default Information Originate Configuration

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route
! Output omitted for brevity
Gateway of last resort is 10.12.1.1 to network 0.0.0.0

O*E2 0.0.0.0/0 [110/60] via 10.12.1.1, 00:02:24, GigabitEthernet0/0/0/0
C    10.12.1.0/24 is directly connected, 01:29:16, GigabitEthernet0/0/0/0
```

```
R4#show ip route
! Output omitted for brevity
Gateway of last resort is 10.34.1.3 to network 0.0.0.0

O*E1 0.0.0.0/0 [110/41] via 10.34.1.3, 00:01:22, GigabitEthernet0/0
     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C    10.34.1.0/24 is directly connected, GigabitEthernet0/0
```

### Note

OSPF advertises the default route as an external route with this technique, so the advertising OSPF router will function as an ASBR.

## OSPF STUBBY AREAS

The previous section focused on summarizing routes as they left an area. OSPF stubby areas provide a method to filter out external and the option to block interarea routes.

OSPF stubby areas are identified by the area flag in the OSPF Hello packet. All routers within an OSPF stubby area need to be configured as a stub so that the routers can establish/maintain an OSPF adjacency. The following section explains the four types of OSPF stubby areas in more detail:

- Stub areas
- Totally stubby areas
- Not-so-stubby areas (NSSAs)
- Totally NSSA areas

### Note

Totally stubby areas and totally NSSA areas are not defined in RFC 2328 but are compliant and commonly implemented by Cisco and other network vendors.

### Stub Areas

OSPF stub areas prohibit Type 5 LSAs (external routes) and Type 4 LSAs (ASBR summary LSAs) from entering the area at the ABR. When a Type 5 LSA reaches the ABR of a stub area, the ABR generates a default route for the stub area via a Type 3 LSA. [Figure 7-33](#) demonstrates the concept.

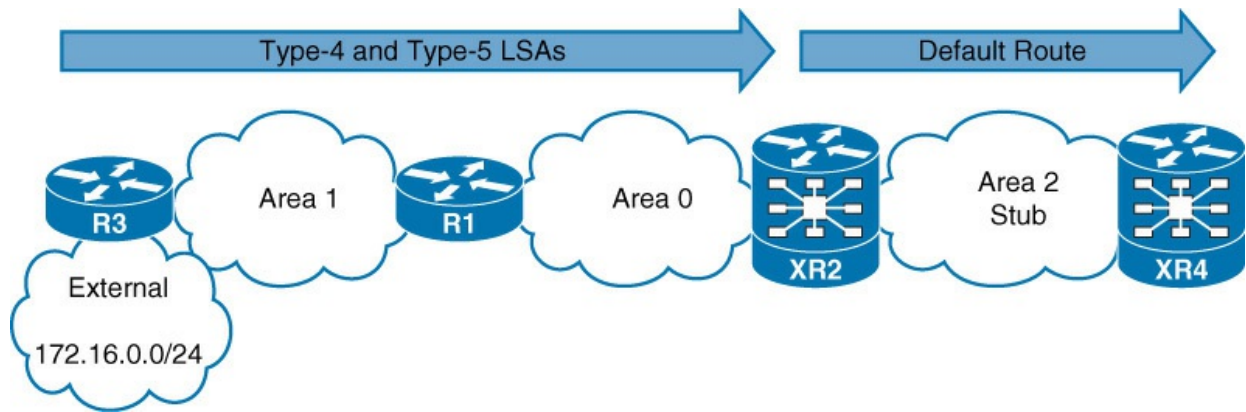


Figure 7-33 OSPF Stub Area Concept

All routers in the stub area must be configured as a stub, or an adjacency will not form because the *area type flags* in the Hello packets will not match. IOS nodes use the command **area area-id stub**, and IOS XR nodes use the command **stub** under the area configuration.

Figure 7-34 demonstrates the stub area with R6 advertising external routes and Area 1 and Area 2 configured as OSPF stub areas.

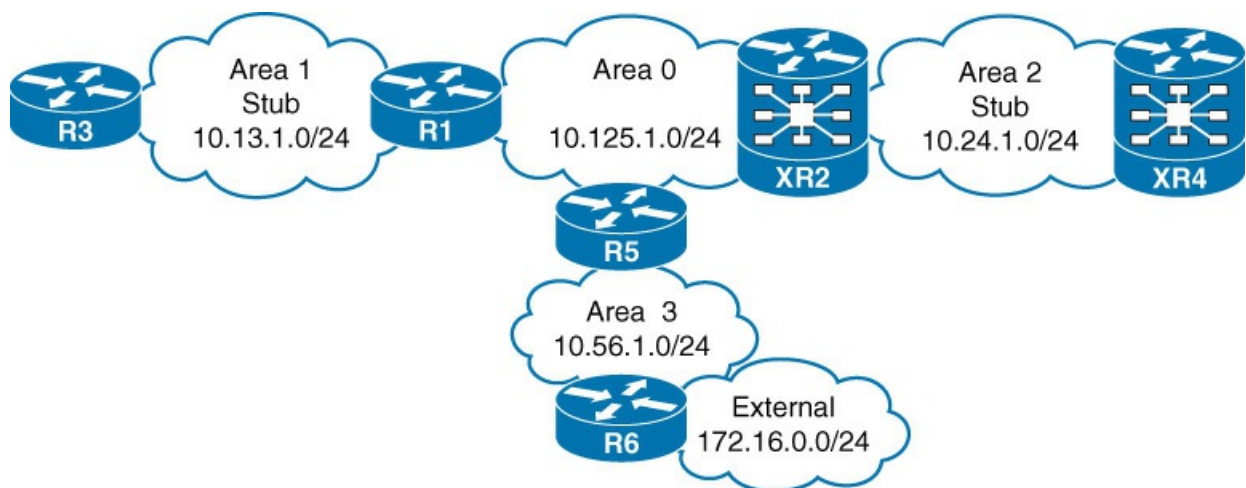


Figure 7-34 OSPF Stub Examples

Example 7-33 displays the route tables for R3 and XR4 with Area 1 and Area 2 as normal OSPF area (nonstub). Notice the external OSPF network in the routing table.

### Example 7-33 Route Table in Area 1 and Area 2 Without Stub

[Click here to view code image](#)

```
R3#show ip route ospf
! Output omitted for brevity

10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
O IA 10.24.1.0/24 [110/21] via 10.13.1.1, 00:00:18, GigabitEthernet0/1
O IA 10.56.1.0/24 [110/30] via 10.13.1.1, 00:00:18, GigabitEthernet0/1
O IA 10.125.1.0/24 [110/20] via 10.13.1.1, 00:00:18, GigabitEthernet0/1
O E2 172.16.0.0/24 [110/20] via 10.13.1.1, 00:00:08, GigabitEthernet0/1
```

```
RP/0/0/CPU0:XR4#show route ospf
```

```
O IA 10.13.1.0/24 [110/12] via 10.24.1.2, 00:00:06, GigabitEthernet0/0/0/2
O IA 10.56.1.0/24 [110/12] via 10.24.1.2, 00:00:06, GigabitEthernet0/0/0/2
O IA 10.125.1.0/24 [110/2] via 10.24.1.2, 00:00:06, GigabitEthernet0/0/0/2
O E2 172.16.0.0/24 [110/20] via 10.24.1.2, 00:00:06, GigabitEthernet0/0/0/2
```

**Example 7-34** provides the OSPF configuration for the routers. Notice that Area 1 and Area 2 are set as an OSPF stub area.

### **Example 7-34** *OSPF Stub Configuration for Area 1 and Area 2*

[Click here to view code image](#)

#### **R1**

```
router ospf 1
  router-id 192.168.1.1
  area 1 stub
  network 10.13.1.1 0.0.0.0 area 1
  network 10.125.1.1 0.0.0.0 area 0
```

#### **XR2**

```
router ospf 1
  area 0
  interface GigabitEthernet0/0/0/2
  !
  !
  area 2
  stub
  interface GigabitEthernet0/0/0/3
```

#### **R3**

```
router ospf 1
  router-id 192.168.3.3
  area 1 stub
  network 0.0.0.0 255.255.255.255 area 1
```

#### **XR4**

```
router ospf 1
  router-id 192.168.4.4
  area 2
  stub
  interface GigabitEthernet0/0/0/2
```

**Example 7-35** displays the route table for R3 and XR4 with Area 1 and Area 2 as OSPF stub areas. When the Type 5 LSA reaches the ABRs (R1 and XR2), the ABRs generate a default route. While R3 and XR4 do not see the route 172.16.0.0/24 in their routing table, they still have connectivity to the network via the default route. Notice the interarea routes are allowed in Area 1 and Area 2.

Default routes are Interarea to prevent advertisement into other areas.

### Example 7-35 R3 and XR4 Routing Table

[Click here to view code image](#)

```
R3#show ip route ospf
! Output omitted for brevity

O*IA 0.0.0.0/0 [110/11] via 10.13.1.1, 00:02:32, GigabitEthernet0/1
    10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
O IA   10.24.1.0/24 [110/21] via 10.13.1.1, 00:02:32, GigabitEthernet0/1
O IA   10.56.1.0/24 [110/30] via 10.13.1.1, 00:02:32, GigabitEthernet0/1
O IA   10.125.1.0/24 [110/20] via 10.13.1.1, 00:02:32, GigabitEthernet0/1
```

```
RP/0/0/CPU0:XR4#show route ospf

O*IA 0.0.0.0/0 [110/2] via 10.24.1.2, 00:14:14, GigabitEthernet0/0/0/2
O IA 10.13.1.0/24 [110/12] via 10.24.1.2, 00:01:32, GigabitEthernet0/0/0/2
O IA 10.56.1.0/24 [110/12] via 10.24.1.2, 00:14:14, GigabitEthernet0/0/0/2
O IA 10.125.1.0/24 [110/2] via 10.24.1.2, 00:14:14, GigabitEthernet0/0/0/2
```

### Totally Stubby Areas

Totally stubby areas prohibit Type 3 LSAs (interarea), Type 4 LSAs (ASBR summary LSAs), and Type 5 LSAs (external routes) from entering the area at the ABR. When an ABR of a totally stubby area receives a Type 3 or Type 5 LSA, the ABR generates a default route for the totally stubby area.

In fact, ABRs for totally stubby area will advertise the default route into the totally stubby areas the instant an interface is assigned to an area outside of the totally stubby area. Assigning the interface acts as the trigger for the Type 3 LSA that leads to the default route generation. Within a totally stubby area only intra-area and default routes should exist.

Figure 7-35 illustrates the totally stubby area concept.

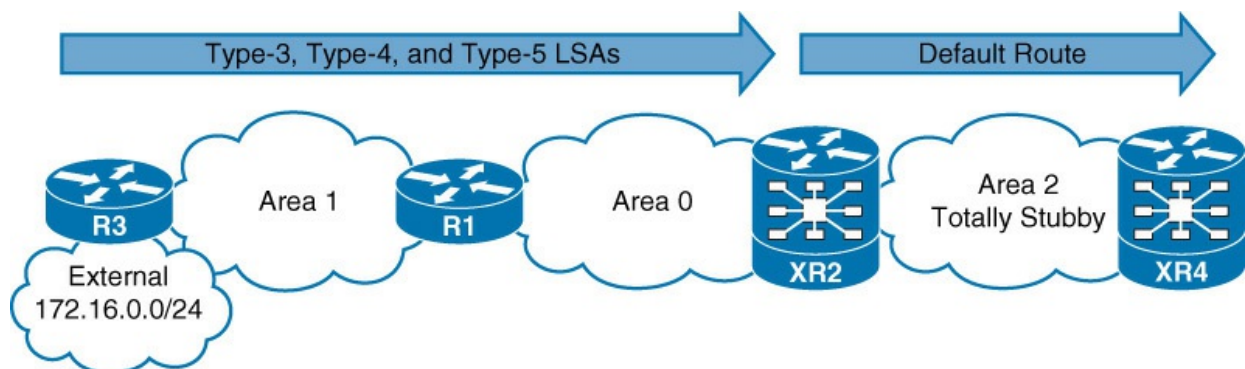


Figure 7-35 Totally Stubby Area Concept

Member routers (Non-ABR routers) of a totally stubby area are configured the same as a stub area. ABRs of a totally stubby area have the **no-summary** appended to the configuration. The command **area area-id stub no-summary** is used on IOS nodes, and the command **stub no-summary** is placed under the area configuration on IOS XR nodes.

Note

The keyword **no-summary** does exactly what it states, and blocks all Type 3 (summary) LSAs going into the stub area making it a totally stubby area.

Figure 7-36 provides a topology example where Area 1 and Area 2 will become totally stubby areas.

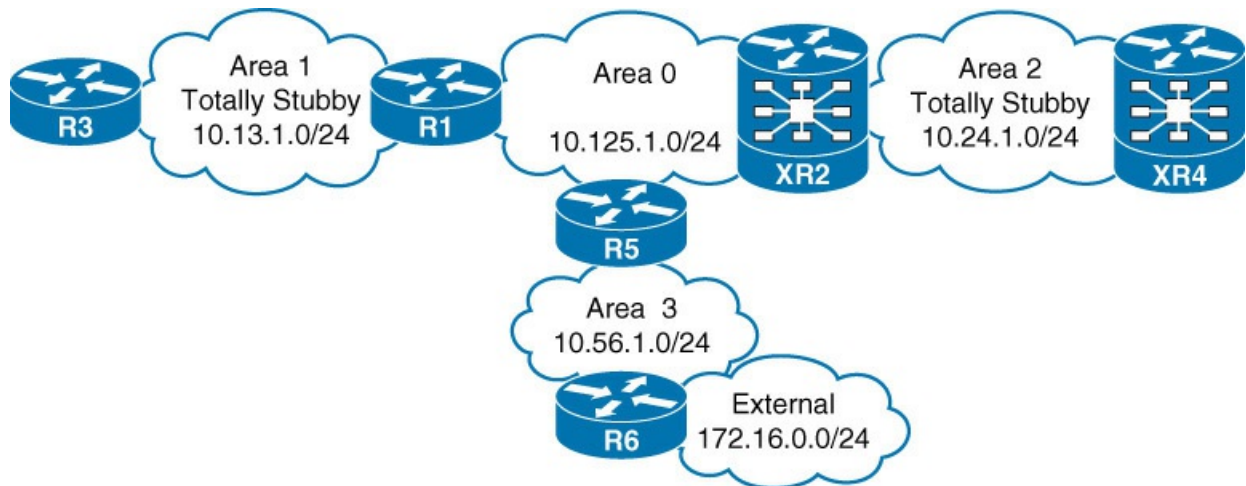


Figure 7-36 Totally Stubby Topology Example

Example 7-36 provides the routing table of R3 and XR4 before Area 1 and Area 2 become totally stubby areas. R1 and XR2 have advertised their Loopback 0 addresses into Area 1 and Area 2 accordingly.

### Example 7-36 Route Table of R3 and XR4 Before Totally Stubby Areas

[Click here to view code image](#)

```
R3#show ip route ospf
! Output omitted for brevity

10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
O IA 10.24.1.0/24 [110/21] via 10.13.1.1, 00:00:18, GigabitEthernet0/1
O IA 10.56.1.0/24 [110/30] via 10.13.1.1, 00:00:18, GigabitEthernet0/1
O IA 10.125.1.0/24 [110/20] via 10.13.1.1, 00:00:18, GigabitEthernet0/1
O E2 172.16.0.0/24 [110/20] via 10.13.1.1, 00:00:08, GigabitEthernet0/1
192.168.1.1/32 is subnetted, 1 subnets
O 192.168.1.1 [110/11] via 10.13.1.1, 00:01:47, GigabitEthernet0/1
```

```
RP/0/0/CPU0:XR4#show route ospf

O IA 10.13.1.0/24 [110/12] via 10.24.1.2, 00:00:06, GigabitEthernet0/0/0/2
O IA 10.56.1.0/24 [110/12] via 10.24.1.2, 00:00:06, GigabitEthernet0/0/0/2
O IA 10.125.1.0/24 [110/2] via 10.24.1.2, 00:00:06, GigabitEthernet0/0/0/2
O E2 172.16.0.0/24 [110/20] via 10.24.1.2, 00:00:06, GigabitEthernet0/0/0/2
O 192.168.2.2/32 [110/2] via 10.24.1.2, 00:00:36, GigabitEthernet0/0/0/2
```

Example 7-37 demonstrates the configuration for totally stubby areas for the routers.



Configuration on R3 and XR4 is identical to a stub area.

### **Example 7-37** *Totally Stubby Area Configurations*

[Click here to view code image](#)

```
R1
router ospf 1
  router-id 192.168.1.1
  area 1 stub no-summary
  network 192.168.1.1 0.0.0.0 area 1
  network 10.13.1.1 0.0.0.0 area 1
  network 10.125.1.1 0.0.0.0 area 0
```

```
XR2
router ospf 1
  area 0
    interface GigabitEthernet0/0/0/2
      !
    !
  area 2
    stub no-summary
    interface Loopback0
      !
    interface GigabitEthernet0/0/0/3
```

```
R3
router ospf 1
  router-id 192.168.3.3
  area 1 stub
  network 0.0.0.0 255.255.255.255 area 1
```

```
XR4
router ospf 1
  router-id 192.168.4.4
  area 2
    stub
    interface GigabitEthernet0/0/0/2
```

[Example 7-38](#) displays the routing table for R3 and XR4 after Area 1 and Area 2 became totally stubby areas. Notice that there are not any interarea OSPF routes, but the intra-area routes (Type 1 LSA) still exists.

### **Example 7-38** *Route Table After Area 1 and 2 Are Made Totally Stubby*

[Click here to view code image](#)

```

R3#show ip route ospf
! Output omitted for brevity

Gateway of last resort is 10.13.1.1 to network 0.0.0.0

O*IA 0.0.0.0/0 [110/11] via 10.13.1.1, 00:05:51, GigabitEthernet0/1
    1.0.0.0/32 is subnetted, 1 subnets
O      192.168.1.1 [110/11] via 10.13.1.1, 00:02:33, GigabitEthernet0/1

RP/0/0/CPU0:XR4#show route ospf

O*IA 0.0.0.0/0 [110/2] via 10.24.1.2, 00:05:27, GigabitEthernet0/0/0/2
O  192.168.2.2/32 [110/2] via 10.24.1.2, 00:02:30, GigabitEthernet0/0/0/2

```

### Not-So-Stubby Areas

OSPF stub areas prohibit Type 5 LSAs (external routes) and Type 4 LSAs (ASBR summary LSAs) from entering the area at the ABR, and they prohibit redistribution of external routes in to the stub area, too. The NSSA prohibits Type 5 LSAs from entering at the ABR, but allows for redistribution of external routes into the NSSA.

As the ASBR redistributes the network into OSPF in the NSSA, the ASBR advertises the network with a Type 7 LSA instead of a Type 5 LSA. When the Type 7 LSA reaches the ABR the ABR converts the Type 7 LSA to a Type 5 LSA.

The ABR does not advertise a default route automatically when a Type 5 or Type 7 LSA is blocked. During configuration an option exists to advertise a default route to provide connectivity to the blocked LSAs, or other techniques will need to be used to ensure bidirectional connectivity.

Figure 7-37 demonstrates the concept of the LSAs being processed on the ABR for the NSSA area. Notice that the default route is optional and depends on the configuration.

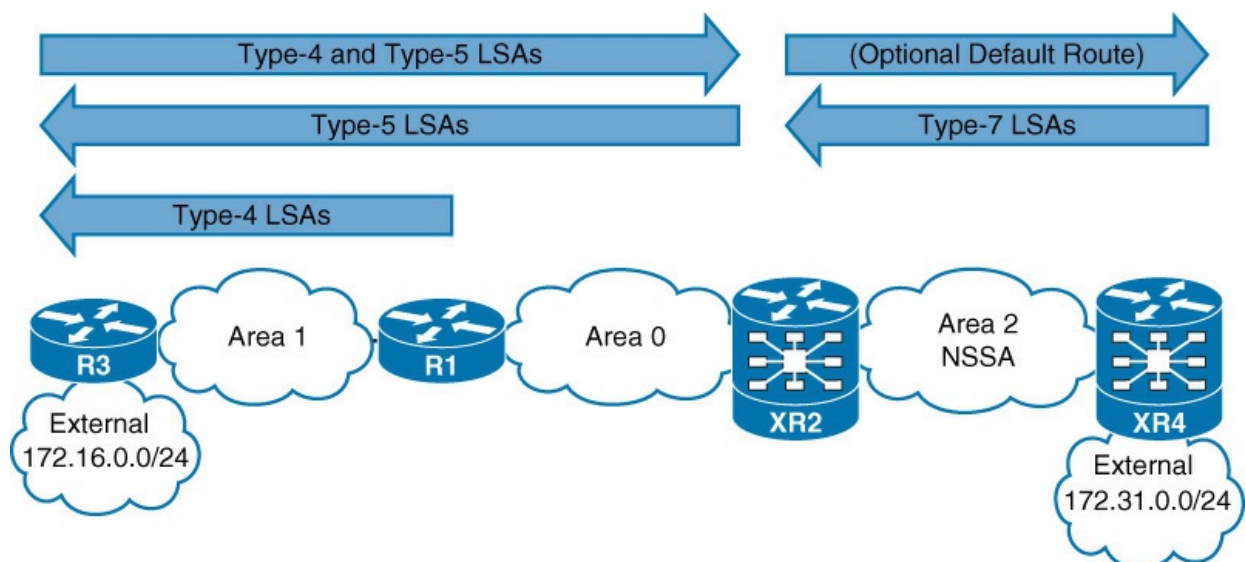


Figure 7-37 NSSA Concept

The command **area area-id nssa [default-information-originate]** is used on IOS nodes, and the command **nssa [default-information-originate]** is placed under the area configuration on IOS XR nodes. All routers in an NSSA area must be configured with the NSSA option, or they will not become adjacent because the area type flags must match in the OSPF Hello protocol to

become adjacent.

A default route is not injected on the ABRs automatically for NSSA areas, but the optional command **default-information-originate** can be appended to the configuration if a default route is needed in the NSSA.

Figure 7-38 provides a topology example to demonstrate an OSPF NSSA. Area 1 and Area 2 are an NSSA with default route origination, whereas R3 and XR4 redistribute a prefix into the OSPF domain.

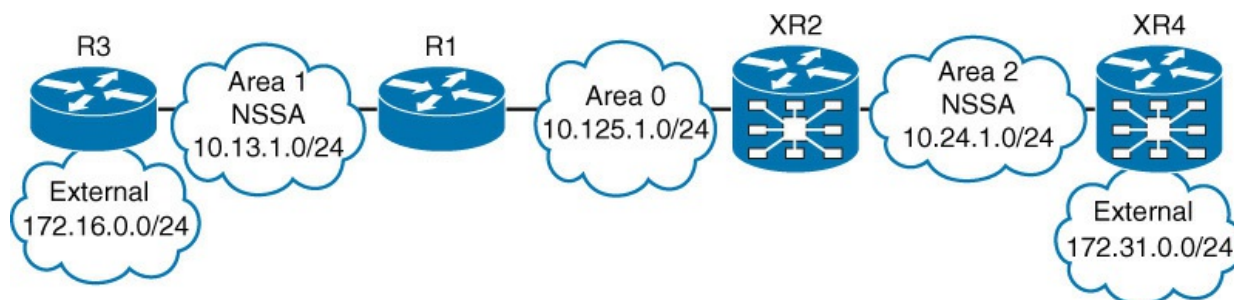


Figure 7-38 OSPF Autonomous Systems

Example 7-39 provides the routing table of R3 and XR4 before Area 1 and Area 2 become an NSSA. Notice the external routes received from the other area.

### Example 7-39 Route Table of R3 and R4 Before Area 1 and Area 2 Are NSSA

[Click here to view code image](#)

```
R3#show ip route ospf
! Output omitted for brevity

10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
O IA 10.24.1.0/24 [110/21] via 10.13.1.1, 00:43:32, GigabitEthernet0/1
O IA 10.125.1.0/24 [110/20] via 10.13.1.1, 00:45:01, GigabitEthernet0/1
172.31.0.0/24 is subnetted, 1 subnets
O E2 172.31.0.0 [110/20] via 10.13.1.1, 00:00:49, GigabitEthernet0/1
```

```
RP/0/0/CPU0:XR4#show route ospf

O IA 10.13.1.0/24 [110/12] via 10.24.1.2, 00:17:00, GigabitEthernet0/0/0/2
O IA 10.125.1.0/24 [110/2] via 10.24.1.2, 00:17:00, GigabitEthernet0/0/0/2
O E2 172.16.0.0/24 [110/20] via 10.24.1.2, 00:03:12, GigabitEthernet0/0/0/2
```

Example 7-40 provides the OSPF configuration. Notice that the ABRs also include the **default-information-originate** option to advertise the default route into the NSSA.

### Example 7-40 NSSA Configuration for Area 1 and Area 2 Routers

[Click here to view code image](#)

**R1**

```
router ospf 1
router-id 192.168.1.1
area 1 nssa default-information-originate
network 10.13.1.1 0.0.0.0 area 1
network 10.125.1.1 0.0.0.0 area 0
```

**XR2**

```
router ospf 1
area 0
interface GigabitEthernet0/0/0/2
!
!
area 2
nssa default-information-originate
interface GigabitEthernet0/0/0/3
```

**R3**

```
router ospf 1
router-id 192.168.3.3
area 1 nssa
redistribute static subnets
network 0.0.0.0 255.255.255.255 area 1
```

**XR4**

```
router ospf 1
router-id 192.168.4.4
redistribute static
area 2
nssa
interface GigabitEthernet0/0/0/2
```

**Example 7-41** provides the routing table of R3 and XR4 after they are NSSA areas. The previous external route from the other area is no longer in the routing table. A default route is injected, but notice that the route shows up as *O N2*. Routes with *O N1* or *O N2* are representatives of Type 7 LSAs in an NSSA.

**Example 7-41** R4 and XR4 OSPF NSSA Route Tables

[Click here to view code image](#)

```

R3#show ip route ospf
! Output omitted for brevity

Gateway of last resort is 10.13.1.1 to network 0.0.0.0

O*N2 0.0.0.0/0 [110/1] via 10.13.1.1, 00:07:50, GigabitEthernet0/1
    10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
O IA   10.24.1.0/24 [110/21] via 10.13.1.1, 00:08:29, GigabitEthernet0/1
O IA   10.125.1.0/24 [110/20] via 10.13.1.1, 00:08:29, GigabitEthernet0/1

RP/0/0/CPU0:XR4#show route ospf

O*N2 0.0.0.0/0 [110/1] via 10.24.1.2, 00:05:50, GigabitEthernet0/0/0/2
O IA 10.13.1.0/24 [110/12] via 10.24.1.2, 00:05:50, GigabitEthernet0/0/0/2
O IA 10.125.1.0/24 [110/2] via 10.24.1.2, 00:05:50, GigabitEthernet0/0/0/2

```

### Totally NSSA Areas

Totally stubby areas prohibit Type 3 LSAs (interarea), Type 4 LSAs (ASBR summary LSAs), and Type 5 LSAs (external routes) from entering the area at the ABR, and they prohibit routes from being redistributed within that area. OSPF areas that need to block Type 3 and Type 5 LSAs, and still provide the capability of redistributing external networks into OSPF, should use the totally NSSA.

When the ASBR redistributes the network into OSPF, the ASBR advertises the network with a Type 7 LSA. As the Type 7 LSA reaches the ABR, the ABR converts the Type 7 LSA to a Type 5 LSA. When an ABR for a Totally NSSA area receives a Type 3 LSA from the backbone, the ABR generates a default route for the totally NSSA. When an interface on the ABR is assigned to Area 0, it acts as the trigger for the Type 3 LSA that leads to the default route generation within the Totally NSSA.

Figure 7-39 demonstrates how the LSAs are processed on the ABR for a totally NSSA.

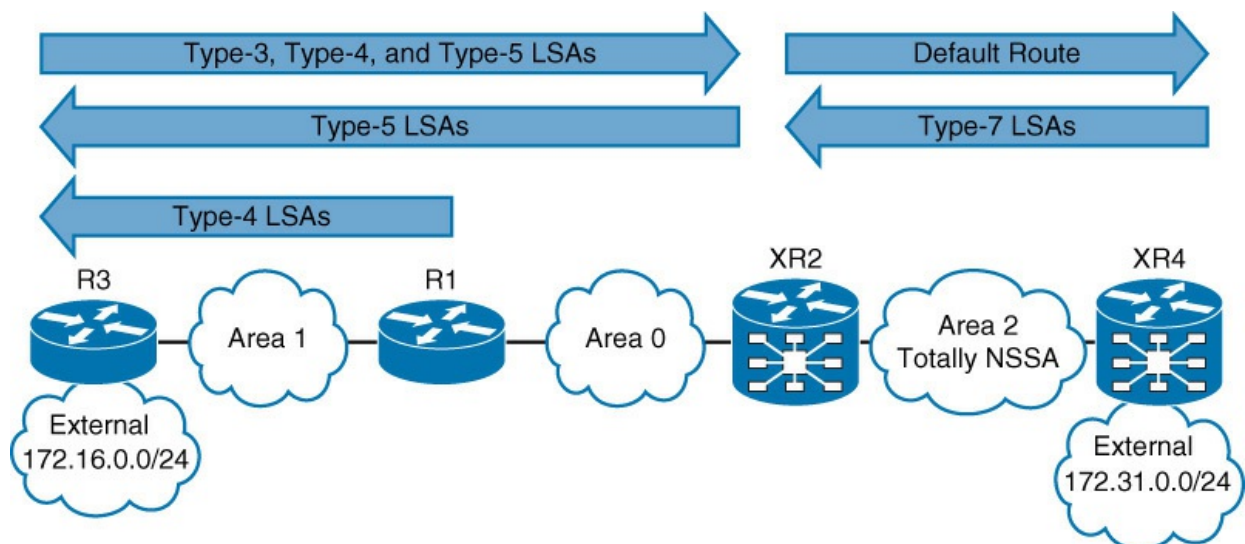


Figure 7-39 Totally NSSA Concept

Member routers of a totally NSSA use the same configuration as members of an NSSA. ABRs of a totally stubby area have the **no-summary** appended to the configuration. The command **area area-id nssa no-summary** is used on IOS nodes, and the command **nssa no-summary** is

placed under the area configuration on IOS XR nodes.

Figure 7-40 provides a topology example to illustrate this concept. Area 1 and Area 2 will become totally NSSAs, whereas R3 and XR4 will redistribute routes into the OSPF domain.

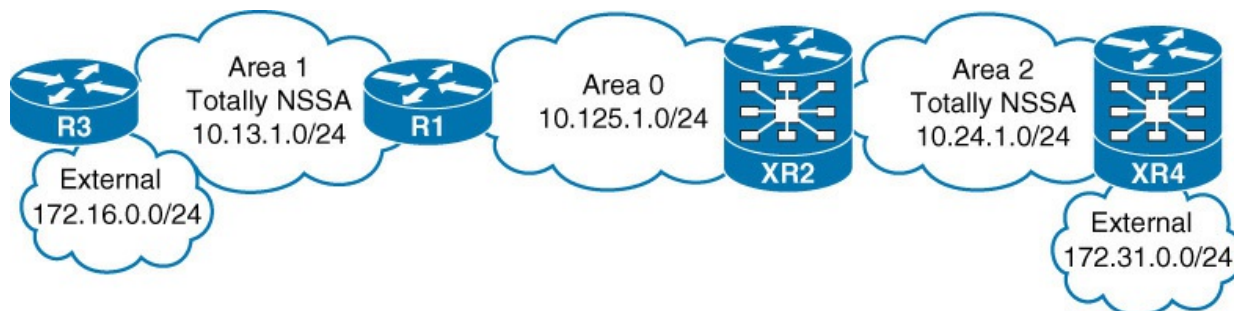


Figure 7-40 Totally NSSA Topology Example

Example 7-42 displays the route table of R3 and R4 before Area 1 and Area 2 become totally NSSA areas. R1 and XR2 advertise their loopback addresses into Area 1 and Area 2 accordingly.

### Example 7-42 Route Table of R3 and R4

[Click here to view code image](#)

```
R3#show ip route ospf
! Output omitted for brevity

10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
O IA 10.24.1.0/24 [110/21] via 10.13.1.1, 00:00:02, GigabitEthernet0/1
O IA 10.125.1.0/24 [110/20] via 10.13.1.1, 00:00:02, GigabitEthernet0/1
172.32.0.0/24 is subnetted, 1 subnets
O E2 172.31.0.0 [110/20] via 10.13.1.1, 00:00:49, GigabitEthernet0/1
192.168.1.0/32 is subnetted, 1 subnets
O 192.168.1.1 [110/11] via 10.13.1.1, 00:00:12, GigabitEthernet0/1
192.168.2.0/32 is subnetted, 1 subnets
O IA 192.168.2.2 [110/21] via 10.13.1.1, 00:00:02, GigabitEthernet0/1
```

```
RP/0/0/CPU0:XR4#show route ospf

O IA 10.13.1.0/24 [110/12] via 10.24.1.2, 00:00:43, GigabitEthernet0/0/0/2
O IA 10.125.1.0/24 [110/2] via 10.24.1.2, 00:00:43, GigabitEthernet0/0/0/2
O E2 172.16.0.0/24 [110/20] via 10.24.1.2, 00:00:43, GigabitEthernet0/0/0/2
O IA 192.168.1.1/32 [110/3] via 10.24.1.2, 00:00:43, GigabitEthernet0/0/0/2
O 192.168.2.2/32 [110/2] via 10.24.1.2, 00:00:43, GigabitEthernet0/0/0/2
```

Example 7-43 provides the relevant configuration on R1 and XR2. The configuration for R3 and XR4 is the same as a NSSA. Notice the **no-summary** command on the **nssa** command.

### Example 7-43 Totally NSSA Configuration

[Click here to view code image](#)

**R1**

```
router ospf 1
router-id 192.168.1.1
area 1 nssa no-summary
network 192.168.1.1 0.0.0.0 area 1
network 10.13.1.1 0.0.0.0 area 1
network 10.125.1.1 0.0.0.0 area 0
```

**XR2**

```
router ospf 1
area 0
interface GigabitEthernet0/0/0/2
!
!
area 2
nssa no-summary
interface Loopback0
!
interface GigabitEthernet0/0/0/3
```

**R3**

```
router ospf 1
router-id 192.168.3.3
area 1 nssa
redistribute static subnets
network 0.0.0.0 255.255.255.255 area 1
```

**XR4**

```
router ospf 1
router-id 192.168.4.4
redistribute static
area 2
nssa
interface GigabitEthernet0/0/0/2
```

**Example 7-44** provides the route table of R3 and XR4 after Area 1 and Area 2 became totally NSSA areas. Notice that the interarea routes are not in the routing table anymore and a default is installed.

**Example 7-44** R3 and R4 OSPF Totally NSSA Route Tables

[Click here to view code image](#)

```
R3#show ip route ospf
```

```
! Output omitted for brevity
```

```
Gateway of last resort is 10.13.1.1 to network 0.0.0.0
```

```
O*IA 0.0.0.0/0 [110/11] via 10.13.1.1, 00:01:40, GigabitEthernet0/1  
1.0.0.0/32 is subnetted, 1 subnets
```

```
O 192.168.1.1 [110/11] via 10.13.1.1, 00:16:18, GigabitEthernet0/1
```

```
RP/0/0/CPU0:XR4#show route ospf
```

```
O*IA 0.0.0.0/0 [110/2] via 10.24.1.2, 00:01:20, GigabitEthernet0/0/0/2
```

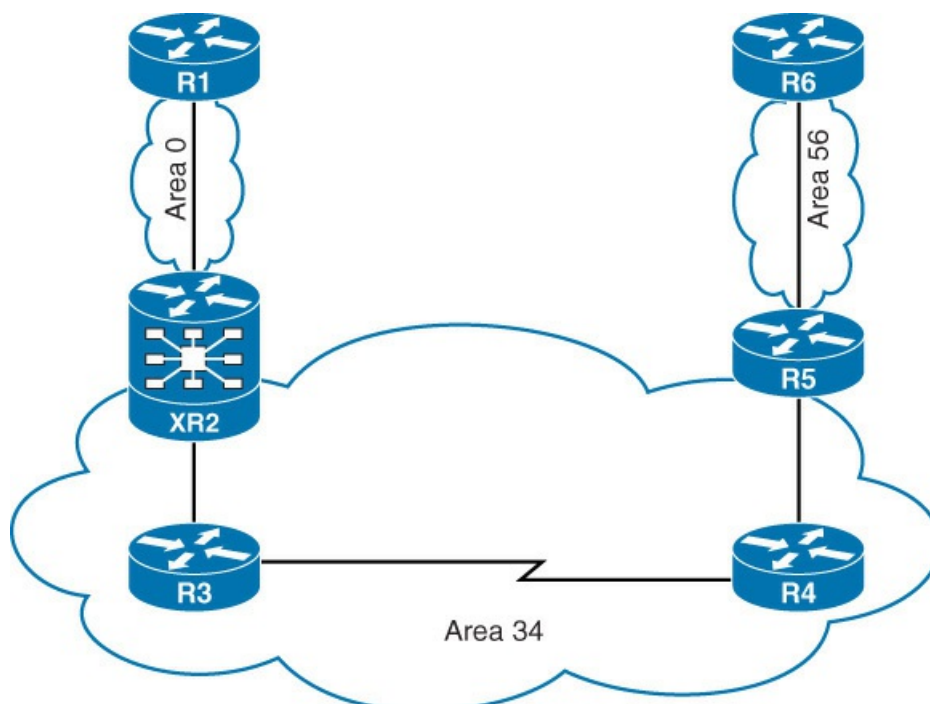
```
O 192.168.2.2/32 [110/2] via 10.24.1.2, 00:13:12, GigabitEthernet0/0/0/2
```

## VIRTUAL LINKS

LSA propagation across OSPF areas requires an OSPF ABR. ABRs require connectivity to Area 0. OSPF routers require connectivity to Area 0 to be an ABR.

*Virtual links* overcome the ABR limitations by extending Area 0 into a nonbackbone area. It is similar to running a virtual tunnel within OSPF between an ABR and another multi-area OSPF router. Virtual links are not supported on any OSPF stubby area.

In [Figure 7-41](#), company ABC just purchased company XYZ. Company XYZ's routers use Area 56 for OSPF. The only connectivity option between the two companies is to install a new circuit between R4 and R5. Enabling OSPF on the new circuit is not enough, and routes will not exchange between Area 56 and Area 34 because R5 is not an ABR.



**Figure 7-41** Bad OSPF Design

XR2 and R5 will form a virtual link across Area 34, thus extending Area 0 to R5 virtually so that it can function as an ABR. This allows routes to be exchanged within the OSPF domain. [Figure 7-42](#)



demonstrates how a virtual link on Area 34 fixes the design issue.

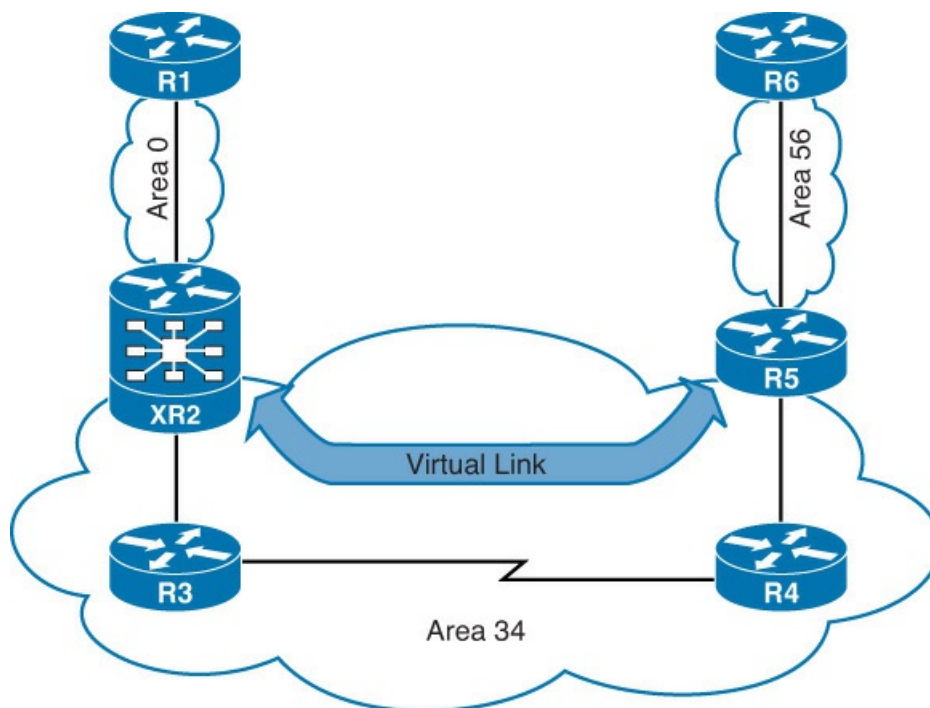


Figure 7-42 OSPF Virtual Links

The virtual link configuration is applied using the command **area area-id virtual-link endpoint-RID** for IOS nodes and the command **virtual-link endpoint-RID** under the specific area for IOS XR nodes. The configuration is performed on both endpoints of the virtual link. At least one endpoint virtual link router has to be a member of Area 0, and virtual links cannot be formed on any OSPF stubby areas. In Figure 7-42, Area 34 cannot be a stub area.

Example 7-45 demonstrates the virtual link configuration between XR2 and R5.

#### Example 7-45 OSPF Virtual Link Configuration

[Click here to view code image](#)

```
XR2
router ospf 1
router-id 192.168.2.2
area 0
interface GigabitEthernet0/0/0/2
!
!
area 34
virtual-link 192.168.5.5
!
interface GigabitEthernet0/0/0/3
```

## R5

```
router ospf 1
router-id 192.168.5.5
area 34 virtual-link 192.168.2.2
network 10.45.45.5 0.0.0.0 area 34
network 10.56.56.5 0.0.0.0 area 56
```

To verify the virtual link status, the command **show ip ospf virtual-links** is used on IOS nodes and the command **show ospf virtual-links** for IOS XR nodes. [Example 7-46](#) displays the output from these commands. Notice that the virtual link status, outbound interface to the endpoints, and the interface cost is in the output.

Interface costs for virtual links cannot be set and is the metric for the intra-area distance between the two virtual link endpoints.

### Example 7-46 OSPF Virtual Link Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show ospf virtual-links
```

```
Virtual Links for OSPF 1
```

```
Virtual Link OSPF_VL0 to router 192.168.5.5 is up
Run as demand circuit
DoNotAge LSA allowed.
Transit area 34, via interface GigabitEthernet0/0/0/3, Cost of using 75
Transmit Delay is 1 sec, State POINT_TO_POINT,
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
Hello due in 00:00:00
Adjacency State FULL (Hello suppressed)
Number of DBD retrans during last exchange 0
Index 2/3, retransmission queue length 0, number of retransmission 0
First 0(0)/0(0) Next 0(0)/0(0)
Last retransmission scan length is 0, maximum is 0
Last retransmission scan time is 0 msec, maximum is 0 msec
```

```
R5#show ip ospf virtual-links
```

```
Virtual Link OSPF_VL0 to router 192.168.2.2 is up
Run as demand circuit
DoNotAge LSA allowed.
Transit area 34, via interface GigabitEthernet0/1
Topology-MTID      Cost      Disabled      Shutdown      Topology Name
      0          75          no            no            Base
Transmit Delay is 1 sec, State POINT_TO_POINT,
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
Hello due in 00:00:00
Adjacency State FULL (Hello suppressed)
Index 1/3, retransmission queue length 0, number of retransmission 0
First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
Last retransmission scan length is 0, maximum is 0
Last retransmission scan time is 0 msec, maximum is 0 msec
```

Note

Virtual links are typically used on a temporary basis because they add complexity in troubleshooting.

## DISCONTIGUOUS NETWORK

Network engineers that do not fully understand OSPF design may create a topology as illustrated in Figure 7-43. While XR2 and R3 have OSPF interfaces in Area 0, traffic from Area 12 must cross Area 23 to reach Area 34. An OSPF network with this design is discontinuous because interarea traffic is trying to cross a nonbackbone area.

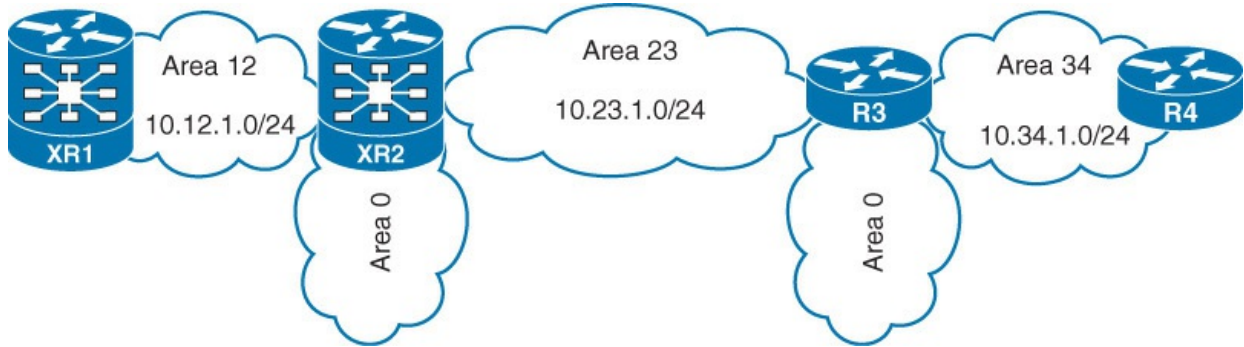


Figure 7-43 Discontiguous Network

**Example 7-47** demonstrates that XR2 and R3 have full connectivity to all networks in the OSPF domain. XR2 has connectivity to the 10.34.1.0/24 network and 192.168.4.4/32 network, and R3 has connectivity to the 10.12.1.0/24 network and 192.168.1.1/32 network.

### Example 7-47 Verification of Nonbroadcast OSPF Interfaces

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route ospf
```

```
O IA 10.34.1.0/24 [110/2] via 10.23.1.3, 00:01:23, GigabitEthernet0/0/0/2
O   192.168.1.1/32 [110/2] via 10.12.1.1, 00:00:07, GigabitEthernet0/0/0/0
O IA 192.168.3.3/32 [110/2] via 10.23.1.3, 00:01:23, GigabitEthernet0/0/0/2
O IA 192.168.4.4/32 [110/3] via 10.23.1.3, 00:01:23, GigabitEthernet0/0/0/2
```

```
R3#show ip route ospf
```

```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
O IA 10.12.1.0/24 [110/2] via 10.23.1.2, 00:01:44, GigabitEthernet0/0
100.0.0.0/32 is subnetted, 4 subnets
O IA 192.168.1.1 [110/3] via 10.23.1.2, 00:00:33, GigabitEthernet0/0
O IA 192.168.2.2 [110/2] via 10.23.1.2, 00:01:44, GigabitEthernet0/0
O 192.168.4.4 [110/2] via 10.34.1.4, 00:01:54, GigabitEthernet0/1
```

**Example 7-48** shows the route tables for XR1 and R4. XR1 is missing route entries for Area 34, and R4 is missing route entries for Area 12. When Area 12's Type 1 LSAs reach XR2, XR2 generates a Type 3 LSA into Area 0 and Area 23. R3 receives the Type 3 LSA and inserts it into

the LSDB for Area 23. R3 does not create a new Type 3 LSA for Area 0 or Area 34.

### Example 7-48 Verification of Nonbroadcast OSPF Interfaces

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route ospf
```

```
O IA 10.23.1.0/24 [110/2] via 10.12.1.2, 00:04:03, GigabitEthernet0/0/0/0
O IA 192.168.2.2/32 [110/2] via 10.12.1.2, 00:04:03, GigabitEthernet0/0/0/0
```

```
R4#show ip route ospf
```

```
! Output omitted for brevity
```

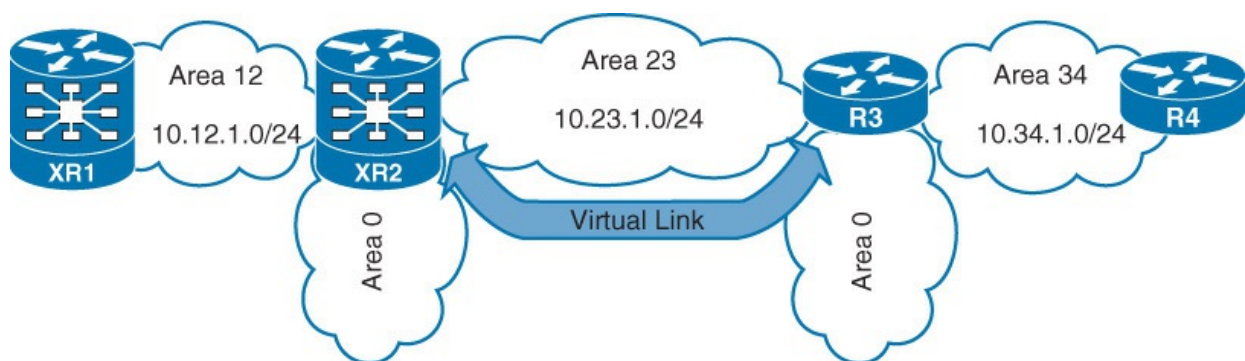
```
Gateway of last resort is not set
```

```
10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
O IA 10.23.1.0/24 [110/2] via 10.34.1.3, 00:01:29, Ethernet0/1
100.0.0.0/32 is subnetted, 2 subnets
O IA 192.168.3.3 [110/2] via 10.34.1.3, 00:07:04, Ethernet0/1
```

ABRs use the following logic for Type 3 LSAs when entering another OSPF area:

- Type 1 LSAs received from an area create Type 3 LSAs into backbone area and nonbackbone areas
- Type 3 LSAs received from Area 0 are created for the nonbackbone area.
- Type 3 LSAs received from a nonbackbone area insert only into the LSDB for the source area. ABRs will not create a Type 3 LSA for the other areas.

The simplest fix for a discontinuous network is to install a virtual link, as shown in [Figure 7-44](#). The virtual link extends Area 0 across Area 23, making Area 0 a contiguous OSPF area.



**Figure 7-44** Repair of a Discontinuous Network

#### Note

Real-life scenarios of discontinuous networks involve Area 0 becoming partitioned due to hardware failures. Another potential workaround is through the usage of generic routing encapsulation (GRE) tunnels between the ABRs, but GRE tunnel configuration is beyond the scope of this book.

## MULTI-AREA ADJACENCY

Figure 7-45 provides a topology example where XR3's and XR4's loopback interfaces are in Area 1. Connectivity between the loopbacks takes the path across the slower serial link (10.56.1.0/24). The serial path is an intra-area path, and the link across the gigabit links (10.34.1.0/24) is an interarea route, which forces a suboptimal path between XR3 and XR4.

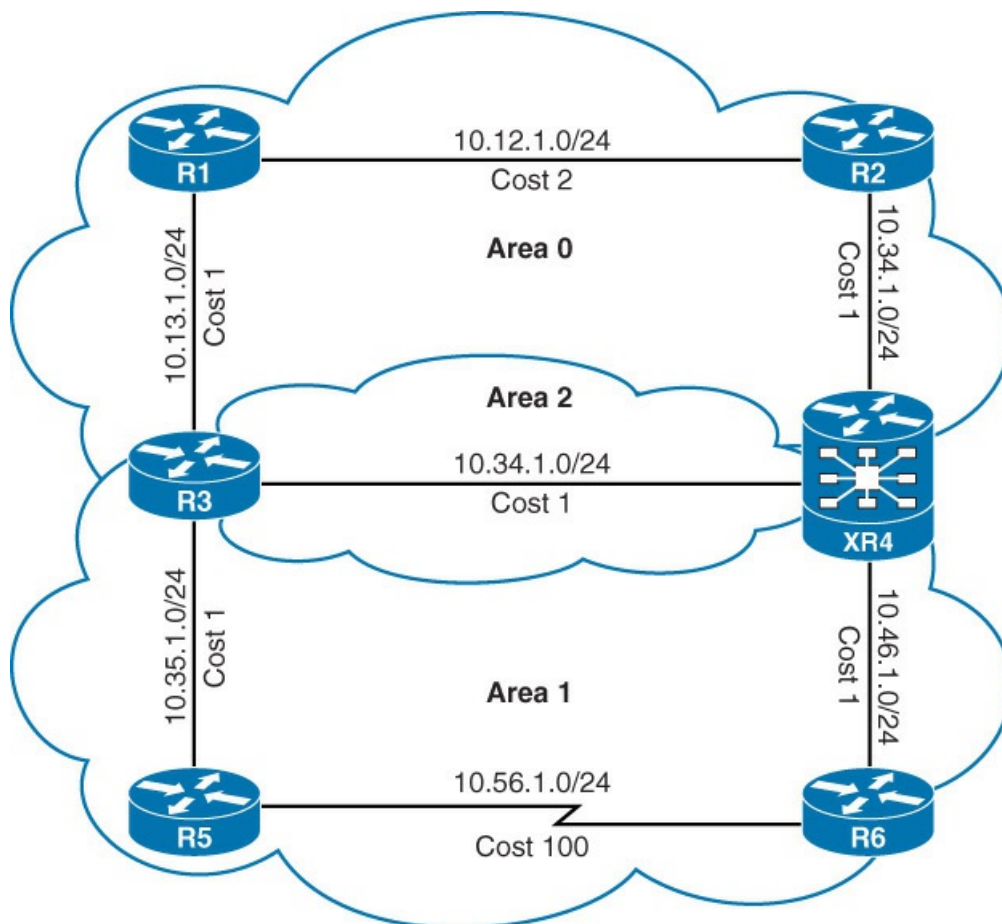


Figure 7-45 Inefficient Topologies

Example 7-49 displays the routing table of the topology. Notice that the metric is 103 across the serial path (10.56.1.0/24) between R3's and XR4's Loopback 0 interfaces.

### Example 7-49 Verification of Nonbroadcast OSPF Interfaces

[Click here to view code image](#)

```
R3#show ip route ospf
! Output omitted for brevity
Gateway of last resort is not set

O   10.12.1.0/24 [110/3] via 10.13.1.1, 00:21:55, GigabitEthernet0/2
O   10.24.1.0/24 [110/4] via 10.13.1.1, 00:21:55, GigabitEthernet0/2
O   10.46.1.0/24 [110/102] via 10.35.1.5, 00:14:03, GigabitEthernet0/3
O   10.56.1.0/24 [110/101] via 10.35.1.5, 00:14:03, GigabitEthernet0/3
O   192.168.1.1/32 [110/2] via 10.13.1.1, 00:23:41, GigabitEthernet0/2
O   192.168.2.2/32 [110/4] via 10.13.1.1, 00:21:55, GigabitEthernet0/2
O   192.168.4.4/32 [110/103] via 10.35.1.5, 00:01:31, GigabitEthernet0/3
O   192.168.5.5/32 [110/2] via 10.35.1.5, 00:25:07, GigabitEthernet0/3
O   192.168.6.6/32 [110/102] via 10.35.1.5, 00:14:03, GigabitEthernet0/3
```

```
RP/0/0/CPU0:XR4#show route ospf
```

```
O 10.12.1.0/24 [110/3] via 10.24.1.2, 00:23:04, GigabitEthernet0/0/0/2
O 10.13.1.0/24 [110/4] via 10.24.1.2, 00:23:04, GigabitEthernet0/0/0/2
O 10.35.1.0/24 [110/102] via 10.46.1.6, 00:14:46, GigabitEthernet0/0/0/3
O 10.56.1.0/24 [110/101] via 10.46.1.6, 00:14:46, GigabitEthernet0/0/0/3
O 192.168.1.1/32 [110/4] via 10.24.1.2, 00:23:04, GigabitEthernet0/0/0/2
O 192.168.2.2/32 [110/2] via 10.24.1.2, 00:23:35, GigabitEthernet0/0/0/2
O 192.168.3.3/32 [110/103] via 10.46.1.6, 00:02:40, GigabitEthernet0/0/0/3
O 192.168.5.5/32 [110/102] via 10.46.1.6, 00:14:46, GigabitEthernet0/0/0/3
O 192.168.6.6/32 [110/2] via 10.46.1.6, 00:25:46, GigabitEthernet0/0/0/3
```

In May 2008, RFC 5185 was released; it allows OSPF point-to-point network interfaces to be members of multiple areas. The interfaces can be any physical interface type, but they must be OSPF point-to-point networks.

IOS nodes use the interface parameter command **ip ospf multi-area area-id** for configuring the second area of adjacency. IOS XR configuration of the second area of adjacency is placed under the appropriate area with the command **multi-area-interface interface-type interface-number**. [Example 7-50](#) shows the configuration of R3's Gi0/0 and XR4's Gi0/0/0/0 interface in Area 1 and Area 2. Notice that the multi-area interfaces are configured as a prerequisite point-to-point OSPF network type.

### **Example 7-50** *OSPF Configuration for Multi-Area Adjacency*

[Click here to view code image](#)

```
R3
interface GigabitEthernet0/0
 ip address 10.34.1.3 255.255.255.255
 ip ospf network point-to-point
 ip ospf multi-area 2
!
router ospf 1
 router-id 192.168.3.3
 network 10.13.1.0 0.0.0.255 area 0
 network 10.35.1.0 0.0.0.255 area 1
 network 10.34.1.0 0.0.0.255 area 1
 network 192.168.3.3 0.0.0.0 area 1
```

```

XR2
router ospf 1
router-id 192.168.4.4
area 0
interface GigabitEthernet0/0/0/2
!
!
area 1
multi-area-interface GigabitEthernet0/0/0/0
!
interface Loopback0
!
interface GigabitEthernet0/0/0/3
!
!
area 2
interface GigabitEthernet0/0/0/0
network point-to-point

```

**Example 7-51** verifies that R3's Gi0/0 and XR4's Gi0/0/0/0 interface are in both Area 1 and Area 2. IOS lists the second interface as MAO, and IOS XR's multi-area interface contains the asterisk (\*) by it.

### **Example 7-51** Verification of Multi-Area Adjacency in Area 1 and Area 2

[Click here to view code image](#)

```

R3#show ip ospf interface brief

```

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Gi0/2	1	0	10.13.1.3/24	1	DR	1/1	
Lo0	1	1	192.168.3.3/32	1	LOOP	0/0	
Gi0/3	1	1	10.35.1.3/24	1	DR	1/1	
Gi0/0	1	2	10.34.1.3/24	1	P2P	1/1	
MAO	1	1	Unnumbered Gi0/0	1	P2P	1/1	

```

RP/0/0/CPU0:XR4#show ospf interface brief

`* Indicates MADJ interface, (P) Indicates fast detect hold down state

Interfaces for OSPF 1

```

Interface	PID	Area	IP Address/Mask	Cost	State	Nbrs	F/C
Gi0/0/0/2	1	0	10.24.1.4/24	1	DR	1/1	
Lo0	1	1	192.168.4.4/32	1	LOOP	0/0	
Gi0/0/0/0*	1	1	10.34.1.4/24	1	P2P	1/1	
Gi0/0/0/3	1	1	10.46.1.4/24	1	DR	1/1	
Gi0/0/0/0	1	2	10.34.1.4/24	1	P2P	1/1	

**Example 7-52** displays that two OSPF neighbor adjacencies have formed across the same interface.

### **Example 7-52** Multi-Area Interface OSPF Neighborhood

[Click here to view code image](#)

```
R2#show ip ospf neighbor
Neighbor ID      Pri   State           Dead Time   Address      Interface
192.168.1.1     1    FULL/BDR        00:00:32   10.13.1.1   GigabitEthernet0/2
192.168.4.4     0    FULL/ -         00:00:37   10.34.1.4   GigabitEthernet0/0
192.168.4.4     0    FULL/ -         00:00:38   10.34.1.4   OSPF_MA0
192.168.5.5     1    FULL/BDR        00:00:37   10.35.1.5   GigabitEthernet0/3
```

```
RP/0/0/CPU0:XR4#show ospf neighbor
```

```
* Indicates MADJ interface
```

```
Neighbors for OSPF 1
```

```
Neighbor ID  Pri   State           Dead Time   Address      Interface
192.168.2.2  1    FULL/BDR        00:00:39   10.24.1.2   GigabitEthernet0/0/0/2
Neighbor is up for 00:28:00
192.168.3.3  1    FULL/ -         00:00:31   10.34.1.3   GigabitEthernet0/0/0/0*
Neighbor is up for 00:02:58
192.168.6.6  1    FULL/BDR        00:00:34   10.46.1.6   GigabitEthernet0/0/0/3
Neighbor is up for 00:30:11
192.168.3.3  1    FULL/ -         00:00:33   10.34.1.3   GigabitEthernet0/0/0/0
Neighbor is up for 00:34:38
```

```
Total neighbor count: 4
```

**Example 7-53** displays R3's and XR4's routing tables with multi-area interfaces configured. Notice that the route to their loopbacks is using the 10.34.1.0/24 network link and has a metric of 1 and not 103 like before.

### **Example 7-53** Verification of Multi-Area OSPF Routes

[Click here to view code image](#)

```
R3#show ip route ospf
! Output omitted for brevity
Gateway of last resort is not set

O    10.12.1.0/24 [110/3] via 10.13.1.1, 00:27:34, GigabitEthernet0/2
O    10.24.1.0/24 [110/4] via 10.13.1.1, 00:27:34, GigabitEthernet0/2
O    10.46.1.0/24 [110/2] via 10.34.1.4, 00:03:16, GigabitEthernet0/0
O    10.56.1.0/24 [110/101] via 10.35.1.5, 00:19:42, GigabitEthernet0/3
O    192.168.1.1/32 [110/2] via 10.13.1.1, 00:29:21, GigabitEthernet0/2
O    192.168.2.2/32 [110/4] via 10.13.1.1, 00:27:34, GigabitEthernet0/2
O    192.168.4.4/32 [110/2] via 10.34.1.4, 00:03:16, GigabitEthernet0/0
O    192.168.5.5/32 [110/2] via 10.35.1.5, 00:30:46, GigabitEthernet0/3
O    192.168.6.6/32 [110/3] via 10.34.1.4, 00:03:16, GigabitEthernet0/0
```



```
RP/0/0/CPU0:XR4#show route ospf
```

```
O 10.12.1.0/24 [110/3] via 10.24.1.2, 00:28:02, GigabitEthernet0/0/0/2
O 10.13.1.0/24 [110/4] via 10.24.1.2, 00:28:02, GigabitEthernet0/0/0/2
O 10.35.1.0/24 [110/2] via 10.34.1.3, 00:03:31, GigabitEthernet0/0/0/0
O 10.56.1.0/24 [110/101] via 10.46.1.6, 00:19:44, GigabitEthernet0/0/0/3
O 192.168.1.1/32 [110/4] via 10.24.1.2, 00:28:02, GigabitEthernet0/0/0/2
O 192.168.2.2/32 [110/2] via 10.24.1.2, 00:28:32, GigabitEthernet0/0/0/2
O 192.168.3.3/32 [110/2] via 10.34.1.3, 00:03:31, GigabitEthernet0/0/0/0
O 192.168.5.5/32 [110/3] via 10.34.1.3, 00:03:31, GigabitEthernet0/0/0/0
O 192.168.6.6/32 [110/2] via 10.46.1.6, 00:30:44, GigabitEthernet0/0/0/3
```

#### Note

Multi-area OSPF adjacency is introduced in various Cisco IOS 15.3 versions of code.

## PREFIX SUPPRESSION

Service provider networks typically have 500,000+ routes in their network. Border Gateway Protocol (BGP) is commonly used for advertisement of network segments with a prefix length smaller than /30 allowing for the interior gateway protocol (IGP) to only include transit networks (/30 and /31) and loopback interfaces (/32). Unless connectivity to a transit network (/30 and /31) is required, leaving the transit network IP prefixes in the LSDB can cause SPF calculations to take longer and can consume more router memory than necessary. Removing the transit networks can reduce the number of prefixes significantly while leaving loopback addresses for establishing BGP sessions.

In Cisco IOS 12.4T, a method to suppress networks in the Type 1 and Type 2 LSAs was introduced. This allows the routing domain to shrink in prefix count while allowing more routers to participate in an OSPF routing domain. OSPF suppression works by filtering out specified networks in only Type 1 and Type 2 LSAs.

When OSPF prefix suppression is enabled, the Type 1 LSAs filter out the stub link (Type 3) on non-DR-enabled network links. This allows the topology to be built using the transit links (Type 2). Revisit [Figure 7-13](#) and notice that R2 and R4 connect to each other using the transit links and not the stub links. This ensures that connectivity is still established after filtering out the Stub Link type that contains the actual network prefix.

Type 2 LSAs notify area routers that the DR enabled network segment should be suppressed by changing the network mask to a special /32 network mask. If a router does not support prefix suppression, traffic destined to that network will still use the DR (advertising router) address to forward packets. OSPF Type 3, Type 4, Type 5, or Type 7 LSAs are not suppressed.

IOS routers can suppress an interface's IP address with the interface parameter command **ip ospf prefix-suppression [disable]**. If a router has multiple interfaces, it might be easier to use the OSPF router configuration command **prefix-suppression**, which will advertise only the passive interfaces, loopback interfaces, or secondary IP addresses.

IOS XR currently does not support this feature, but it is planned for future software versions.

[Figure 7-46](#) demonstrates the concept of OSPF prefix suppression. R1 connects to R2 via serial link, R2 connects to R3 via Ethernet, and R3 connects to R4 via serial link. Only connectivity to

the Loopback 0 interfaces is required.

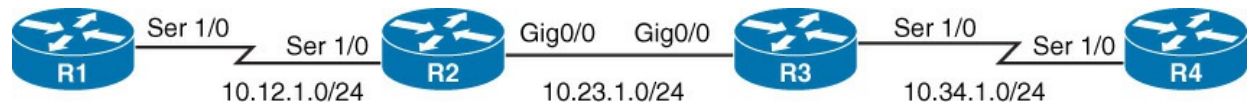


Figure 7-46 OSPF Prefix Suppression

Example 7-54 displays R1's routing table. Notice that the transit networks 10.23.1.0/24 and 10.34.1.0/24 are present.

### Example 7-54 R1 Route Table

[Click here to view code image](#)

```
R1#show ip route ospf
! Output omitted for brevity

O       10.23.1.0/24 [110/65] via 10.12.1.2, 00:01:55, Serial1/0
O       10.34.1.0/24 [110/129] via 10.12.1.2, 00:01:55, Serial1/0
O       192.168.2.2 [110/65] via 10.12.1.2, 00:03:10, Serial1/0
O       192.168.3.3 [110/66] via 10.12.1.2, 00:01:55, Serial1/0
O       192.168.4.4 [110/130] via 10.12.1.2, 00:00:03, Serial1/0
```

Example 7-55 displays the Type 1 LSAs for the topology. The Type 1 LSAs are for the Loopback 0 interfaces and the serial links (10.12.1.0/24 and 10.34.1.0/24). There are a total of eight stub links spread across four routers (two entries per router). The transit links that will be suppressed are highlighted.

### Example 7-55 Type 1 LSAs Before Suppression

[Click here to view code image](#)

```

R1#show ip ospf database router | i Advertising Router|Link conn|Network/subn
Advertising Router: 192.168.1.1
  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 192.168.1.1
  Link connected to: another Router (point-to-point)
  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 10.12.1.0
Advertising Router: 192.168.2.2
  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 192.168.2.2
  Link connected to: another Router (point-to-point)
  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 10.12.1.0
  Link connected to: a Transit Network
Advertising Router: 192.168.3.3
  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 192.168.3.3
  Link connected to: another Router (point-to-point)
  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 10.34.1.0
  Link connected to: a Transit Network
Advertising Router: 192.168.4.4
  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 192.168.4.4
  Link connected to: another Router (point-to-point)
  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 10.34.1.0

```

**Example 7-56** displays the OSPF Type 2 LSAs for this topology. Notice that the subnet mask is a /24.

### **Example 7-56** *Type 2 LSAs Before Suppression*

[Click here to view code image](#)

```

R1#show ip ospf database network | i Mask|Attached Router|State ID
Link State ID: 10.23.1.3 (address of Designated Router)
Network Mask: /24
  Attached Router: 192.168.3.3
  Attached Router: 192.168.2.2

```

**Example 7-57** demonstrates the configuration of the OSPF prefix suppression. R1, R3, and R4 enable OSPF prefix suppression for the entire router; and R2 enables it specifically for the Gi0/0 and Serial 1/0 interfaces.

### **Example 7-57** *Prefix Suppression Configuration*

[Click here to view code image](#)

**R1**

```
router ospf 1
  prefix-suppression
  network 0.0.0.0 255.255.255.255 area 0
```

**R2**

```
interface GigabitEthernet0/0
  ip address 10.23.1.2 255.255.255.0
  ip ospf prefix-suppression
  ip ospf cost 1
!
interface Serial11/0
  ip address 10.12.1.2 255.255.255.0
  ip ospf prefix-suppression
  serial restart-delay 0
!
router ospf 1
  prefix-suppression
  network 0.0.0.0 255.255.255.255 area 0
```

**R3**

```
router ospf 1
  prefix-suppression
  network 0.0.0.0 255.255.255.255 area 0
```

**R4**

```
router ospf 1
  prefix-suppression
  network 0.0.0.0 255.255.255.255 area 0
```

**Example 7-58** verifies that R1 no longer contains the transit networks (10.23.1.0/24 and 10.34.1.0/24) in the RIB.

**Example 7-58** *VR1 Route Table After Prefix Suppression*

[Click here to view code image](#)

```
R1#show ip route ospf
! Output omitted for brevity

O          192.168.2.2 [110/65] via 10.12.1.2, 00:03:10, Serial11/0
O          192.168.3.3 [110/66] via 10.12.1.2, 00:01:55, Serial11/0
O          192.168.4.4 [110/130] via 10.12.1.2, 00:00:03, Serial11/0
```

**Example 7-59** displays the Type 1 LSAs after the OSPF prefix suppression. Notice how the stub network entries are not present for the transit links.

**Example 7-59** *Type 1 LSAs After Prefix Suppression*

[Click here to view code image](#)

```
R1#show ip ospf database router | i Advertising Router|Link conn|Network/subn
Advertising Router: 192.168.1.1
  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 192.168.1.1
  Link connected to: another Router (point-to-point)
Advertising Router: 192.168.2.2
  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 192.168.2.2
  Link connected to: another Router (point-to-point)
  Link connected to: a Transit Network
Advertising Router: 192.168.3.3
  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 192.168.3.3
  Link connected to: another Router (point-to-point)
  Link connected to: a Transit Network
Advertising Router: 192.168.4.4
  Link connected to: a Stub Network
    (Link ID) Network/subnet number: 192.168.4.4
  Link connected to: another Router (point-to-point)
```

[Example 7-60](#) displays the LSDB's Type 2 LSA after prefix suppression was enabled. Notice that the subnet mask (network mask) has been changed to a /32.

### **Example 7-60** *Type 2 LSAs After Prefix Suppression*

[Click here to view code image](#)

```
R1#show ip ospf database network | i Mask|Attached Router|State ID
Link State ID: 10.23.1.3 (address of Designated Router)
Network Mask: /32
  Attached Router: 192.168.3.3
  Attached Router: 192.168.2.2
```

[Example 7-61](#) demonstrates that R4 can connect to R1. Notice that the packet is sourced from R4's Loopback 0 interface to ensure that R1 will be able to reply.

### **Example 7-61** *Verification of Connectivity with Ping*

[Click here to view code image](#)

```
R4#ping 192.168.1.1 source loopback 0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.4.4
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 5/5/6 ms
```

[Example 7-62](#) demonstrates that the transit networks are still visible in the traceroute while they are not present in the OSPF LSDB.

## Example 7-62 Verification of Connectivity with Traceroute

[Click here to view code image](#)

```
R4#traceroute 192.168.1.1 source loopback 0
Type escape sequence to abort.
Tracing the route to 192.168.1.1
VRF info: (vrf in name/id, vrf out name/id)
 1 10.34.1.3 0 msec 1 msec 1 msec
 2 10.23.1.2 6 msec 3 msec 3 msec
 3 10.12.1.1 9 msec * 7 msec
```

### SUMMARY

OSPF scales very well with proper design supporting thousands of network prefixes. This chapter explained key concepts around the design and scalability of an OSPF domain.

OSPF LSAs use sequence numbers and aging to ensure that all routers within the OSPF domain maintain correct and current information in the LSDB. There are seven important LSAs essential to a traditional OSPF topology:

- **Type 1, router:** LSAs within an area
- **Type 2, network:** Advertise number of routers attached to broadcast segment
- **Type 3, summary:** Advertises network prefixes in nonoriginating areas
- **Type 4, ASBR summary:** Summary LSA for a specific ASBR
- **Type 5, AS external:** LSAs for routes that have been redistributed
- **Type 7, NSSA external:** Redistributed routes in NSSAs

Route summarization occurs between areas because the LSDB is identical between all routers in an area. Summarization reduces the memory consumption for the routing table and provides fault isolation because visibility is removed for more specific prefixes.

OSPF stubby areas filter Type 5 LSAs from entering an area. Other stubby areas block Type 3 LSAs from entering the area, which can drastically reduce the size of LSDBs. Connectivity to the blocked LSAs is maintained via a default route that is generated by the ABR upon the blocking of the appropriate LSA. [Table 7-8](#) provides a review of the OSPF stubby area types and the LSAs they block.

<b>Area Type</b>	<b>Allows Type 4 and Type 5 LSAs</b>	<b>Allows Type 3 LSAs</b>	<b>Allows Redistribution (Type 7 in Lieu of Type 5)</b>
Stub	No	Yes	No
Totally stubby	No	No	No
NSSA	No	Yes	Yes
Totally NSSA	No	No	Yes

**Table 7-8** *OSPF Stubby Areas*

OSPF uses a two-tier hierarchical architecture, where Area 0 is a special area known as the *backbone*, and where all other OSPF routers must connect to Area 0. Nonbackbone areas advertise routes into the backbone, and the backbone then advertises routes into other nonbackbone areas. Virtual links extend backbone connectivity into other areas, providing connectivity for discontinuous network designs, or merging networks.

## REFERENCES IN THIS CHAPTER

RFC 2328, *OSPF Version 2*, J. Moy, IETF, <https://tools.ietf.org/rfc/rfc2328.txt>, April 1998.

RFC 3509, *Alternative Implementations of OSPF Area Border Routers*, A. Zinin, et al. IETF, <https://tools.ietf.org/rfc/rfc3509.txt>, April 2003.

RFC 5185, *OSPF Multi-Area Adjacency*, S. Mirtorabi, et al. IETF, <https://tools.ietf.org/rfc/rfc5185.txt>, May 2008.

Cisco. Cisco IOS Software Configuration Guides. <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides. <http://www.cisco.com>

## Chapter 8. IS-IS

This chapter covers the following topics:

- IS-IS fundamentals
- OSI addressing
- IS-IS inter-router communication
- IS-IS configuration
- IS-IS link-state packet database
- Authentication

Intermediate System-to-Intermediate System (IS-IS) Protocol is a nonproprietary link-state routing protocol that is commonly found in service providers and some enterprise networks. IS-IS provides fast convergence, supports a large number of networks, and can support multiple protocols.

### IS-IS FUNDAMENTALS

The first computer networking protocols were government or vendor proprietary. Connecting networks that used different protocols was a complicated task. The International Organization for Standardization (ISO) along with multiple vendors created a reference model composed of seven layers called the *OSI model*. The OSI model describes the communication between host to host over a network. Each layer describes a specific function with the notion that a layer can be modified or changed without needing changes to the layer above or below it. The OSI model provides a structured approach for interoperability between vendors.

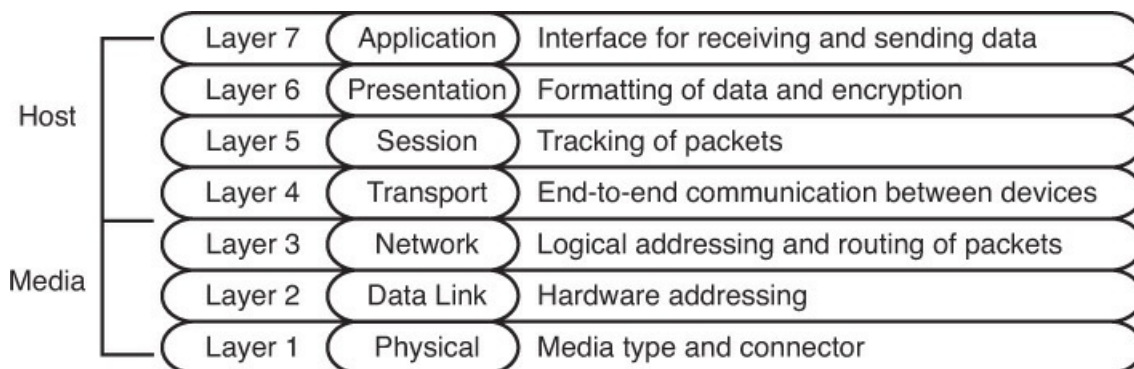


Figure 8-1 OSI Model

As the TCP/IP protocol suite was being developed by the IETF, ISO created the OSI protocol suite, too. The OSI protocol suite included the Connectionless Network Protocol (CLNP) for Layer 3 connectivity, which is the equivalent to the IP protocol. The Connectionless Network Service (CLNS) provides the mechanism for transporting protocol data units (PDUs), also known as *packets*, between nodes. In the OSI model, CLNS PDUs transport CLNP datagrams.

In the OSI protocol suite, end systems (ES) are any nonrouting network device. Routers are



called *intermediate systems* (IS), providing connectivity to end systems. Connections between end systems and routers are notated as ES-IS connections. Router-to-router communication therefore is called *intermediate system-to-intermediate system*.

Figure 8-2 illustrates the ES-IS and IS-IS connections in a topology between workstations and multiple routers.

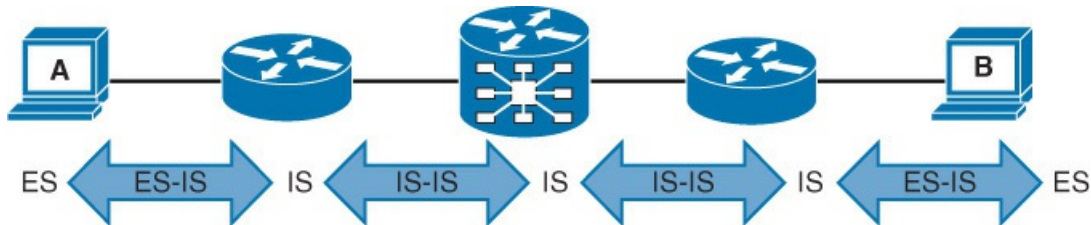


Figure 8-2 ES-IS and IS-IS

IS-IS and OSPF share the following commonalities:

- Nonproprietary
- Two-level hierarchy of areas to provide scalability
- Routers within an area maintain a complete topology table

IS-IS uses link-state packets (LSPs) for advertising topology. Open Shortest Path First (OSPF) Protocol uses link-state advertisements (LSAs) for advertising topology.

- Use a Dijkstra-based algorithm to compute the shortest-path first (SPF) tree

IS-IS uses a two-level hierarchy consisting of Level 1 (L1) and Level 2 (L2) connections. ES-IS communication occurs only at L1. IS-IS communication can occur at L1, L2, or both (L1-L2). L2 routers communicate only with other L2 routers, and L1 routers only communicate with other L1 routers. L1-L2 routers provide connectivity between the L1 and L2 levels. An L2 router can communicate with L2 routers in the same or a different area; whereas, an L1 router can communicate only with other L1 routers within the same area. The following list indicates the type of adjacencies that can form between IS-IS routers:

- L1 <-- --> L1
- L2 <-- --> L2
- L1-L2 <-- --> L1
- L1-L2 <-- --> L2
- L1-L2 <-- --> L1-L2

IS-IS uses the LSPs for building a link-state packet database (LSPDB) similar to OSPF's link-state database (LSDB). IS-IS then runs the Dijkstra Shortest Path First (SPF) algorithm to construct a loop-free topology of shortest paths. Each router sees itself as the top of the tree, and the tree

contains all network destinations within the IS-IS level. The SPF tree (SPT) will differ for each IS-IS router, but the LSPDB used to calculate the SPT is identical for all IS-IS routers in a specific level.

Figure 8-3 provides a simple IS-IS topology and the SPT from R1's and R4's perspective. Notice that the local router's perspective will always be the top of the tree. There is a difference in connectivity to the 10.3.3.0/24 network from R1's and R4's SPT. From R1's view, the serial link between R3 and R4 is not needed; and from R4's standpoint, the Ethernet link between R1 and R3 is missing.

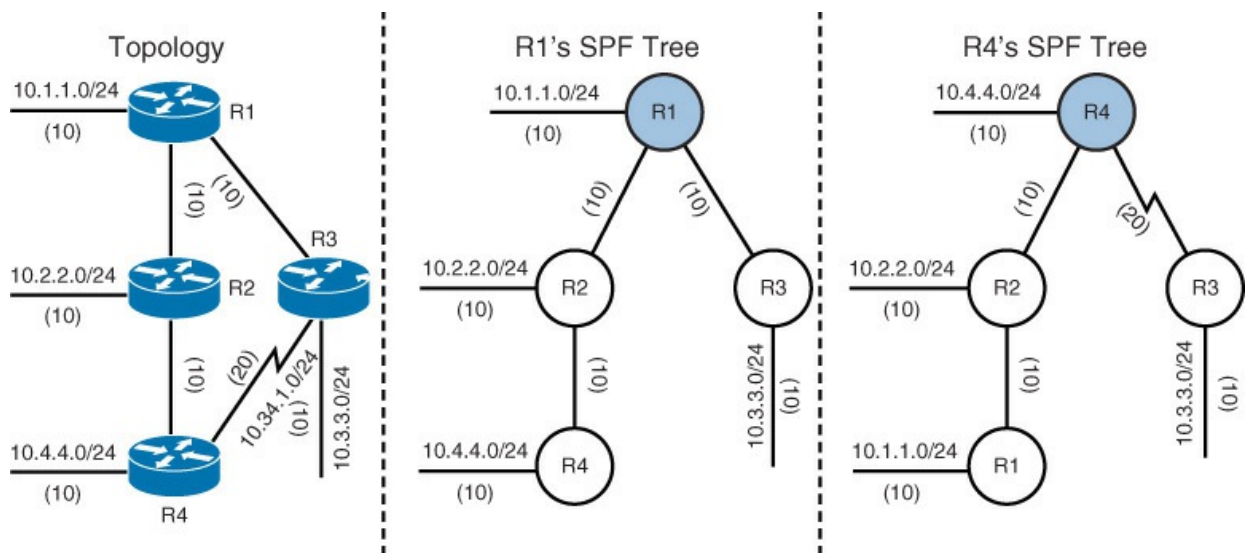


Figure 8-3 IS-IS SPT

The SPTs give the illusion that there is no redundancy in the network. Remember that the SPT shows the shortest path to reach a network and calculates off the LSPDB, which contains all the links for an area. The SPT rebuilds and changes during a topology change.

### Areas

OSPF and IS-IS use a two-level hierarchy, but they function differently between the protocols. OSPF provides connectivity between areas by allowing a router to participate in multiple areas, whereas IS-IS places the entire router and all its interfaces in a specific area. OSPF's hierarchy is based on areas advertising prefixes into the backbone, which then are advertised into nonbackbone areas. Level 2 is the IS-IS backbone and can cross multiple areas unlike OSPF as long as the L2 adjacencies are contiguous.

Figure 8-4 demonstrates these basic differences between OSPF and IS-IS. Notice that the IS-IS backbone extends across four areas, unlike OSPF's.

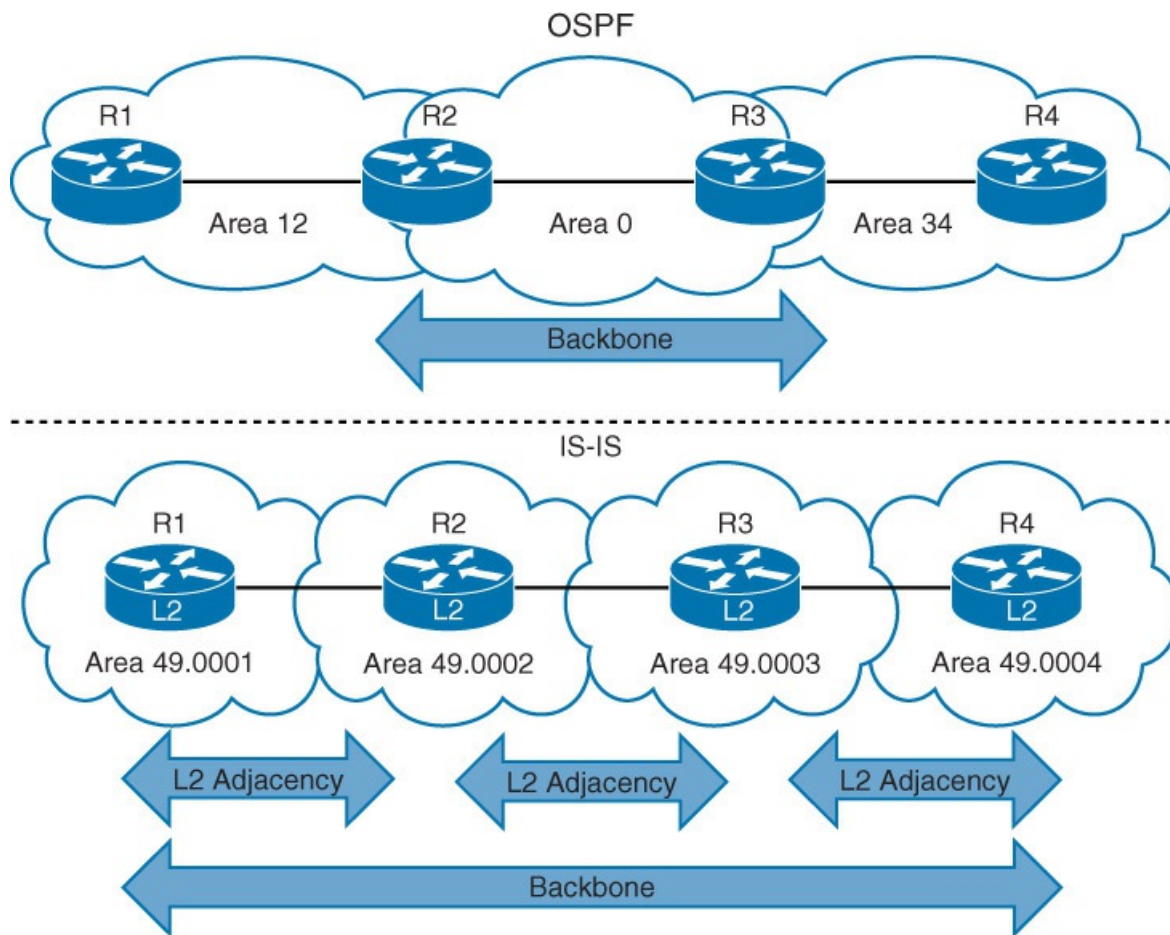


Figure 8-4 Areas Within OSPF and IS-IS

In Figure 8-5, R1 and R2 form an L1 adjacency with each other, and R4 and R5 form an L1 adjacency with each other. Although R2 and R4 are L1-L2 routers, R1 and R5 support only an L1 connection. The area address must be the same to establish an L1 adjacency. R2 establishes an L2 adjacency with R3, and R3 establishes an L2 adjacency with R4. R2 and R4 are L1-L2 routers and can form an L1 and L2 adjacency on them.

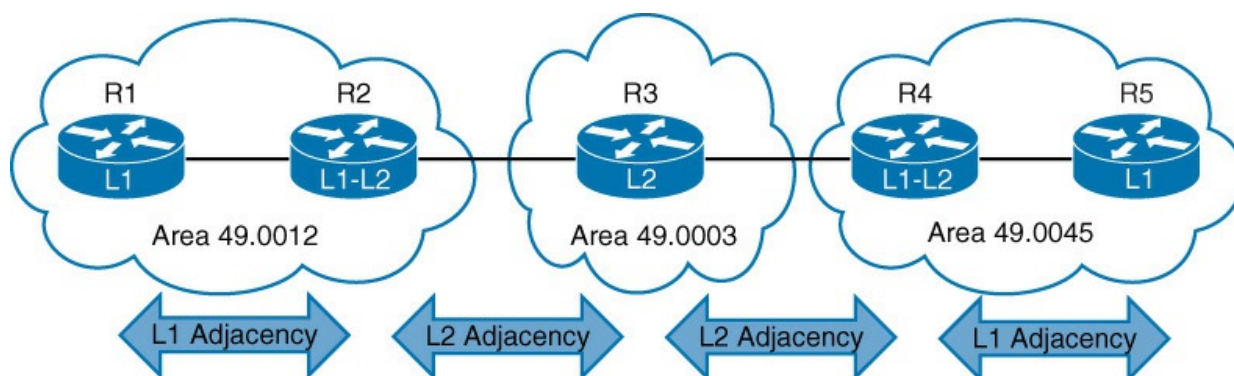


Figure 8-5 IS-IS Adjacencies

All L1 IS-IS routers in the same level maintain an identical copy of the LSPDB, and all L1 routers do not know about any routers or networks outside of their level (area). In a similar fashion, L2 routers maintain a separate LSPDB that is identical with other L2 routers. L2 routers are only aware of other L2 routers and networks in the L2 LSPDB.

L1-L2 routers inject L1 prefixes into the L2 topology. L1-L2 routers do not advertise L2 routes

into the L1 area, but they set the *attached bit* in their L1 LSP indicating that the router has connectivity to the IS-IS backbone network. If an L1 router does not have a route for a network, it searches the LSPDB for the closest router with the attached bit, which acts as a route of last resort. This concept is explained in greater depth in the following chapter.

Figure 8-6 provides a simple topology to illustrate this concept. R2 has all the route prefixes for all of the routers and provides R1 connectivity to Area 49.0003 and Area 49.0045 through the attached bit. The same logic applies to R4's routing table and advertisement of the attached bit to R5.

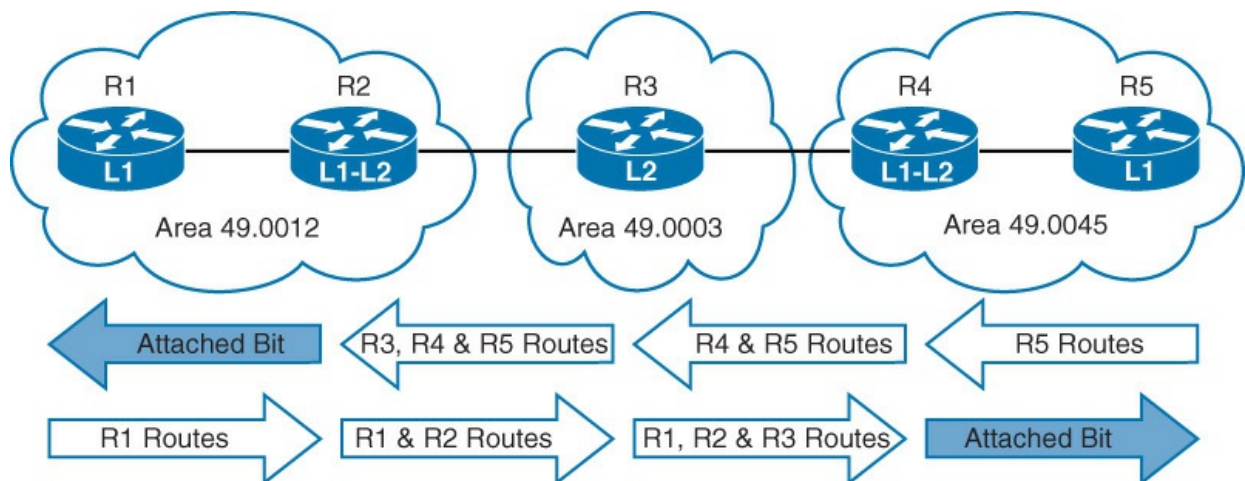


Figure 8-6 IS-IS Router Advertisements and Attached Bit

### OSI Addressing

Ensuring that a router is unique in an IS-IS routing domain is essential for properly building the LSPDB. The Network Service Access Point (NSAP) is a unique OSI address that resides between the third and the fourth layer of the OSI model and is used to exchange information from layer to layer in the OSI model. The NSAP address is between 8 to 20 bytes in length and is variable based on the logic for addressing domains.

The primary structure of an NSAP address consists of the Inter-Domain Part (IDP) and the Domain Specific Part (DSP). The IDP is standardized by ISO and is responsible for ensuring a unique area exists when exchanging routes with external domains. The IDP is managed by an external addressing authority. A portion of the DSP is assigned by the addressing authority specified in the IDP.

Figure 8-7 shows the NSAP structure.

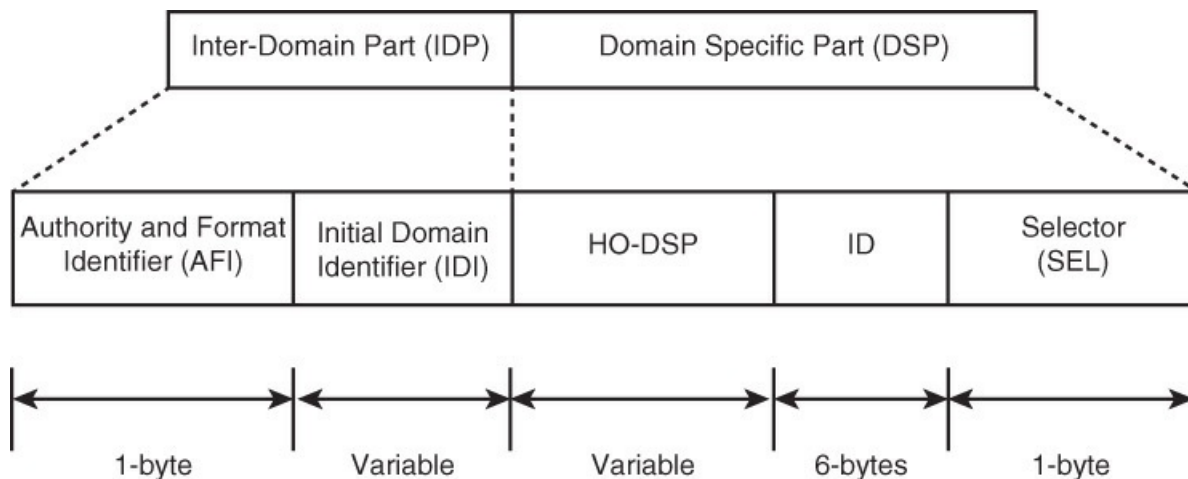


Figure 8-7 Primary NSAP Structure

### Inter-Domain Part

The IDP is categorized further into an Authority and Format Identifier (AFI) field, which references the primary addressing authority for issuing the interdomain addresses. The AFI indicates the top-level addressing authority that issued the external portion of the NSAP address. The AFI uses values between 00 and 99, and the AFI 49 is a private number similar to RFC 1918 for IP addresses and should never route between external IS-IS routing domains.

The Initial Domain Identifier (IDI) is a suborganization to the primary addressing authority (AFI) and further allocates the IDP address into smaller components. The variable length of the IDP is because each suborganization can have a different substructure from its peers. For example, organization ABC could use a 3-byte substructure, whereas organization XYZ could use a 5-byte substructure.

A similar analogy would be the IANA allocates blocks of IP addresses to ARIN, and ARIN then allocates blocks of IP addresses to the service providers, who then allocate to their customers. Each organization in the path may have a different logic for allocating IP addresses.

Figure 8-8 shows the complete logical sub structure of the IDP allocation scheme. This level of granularity is beyond the scope of this book, but RFC 941 explains them in full detail.

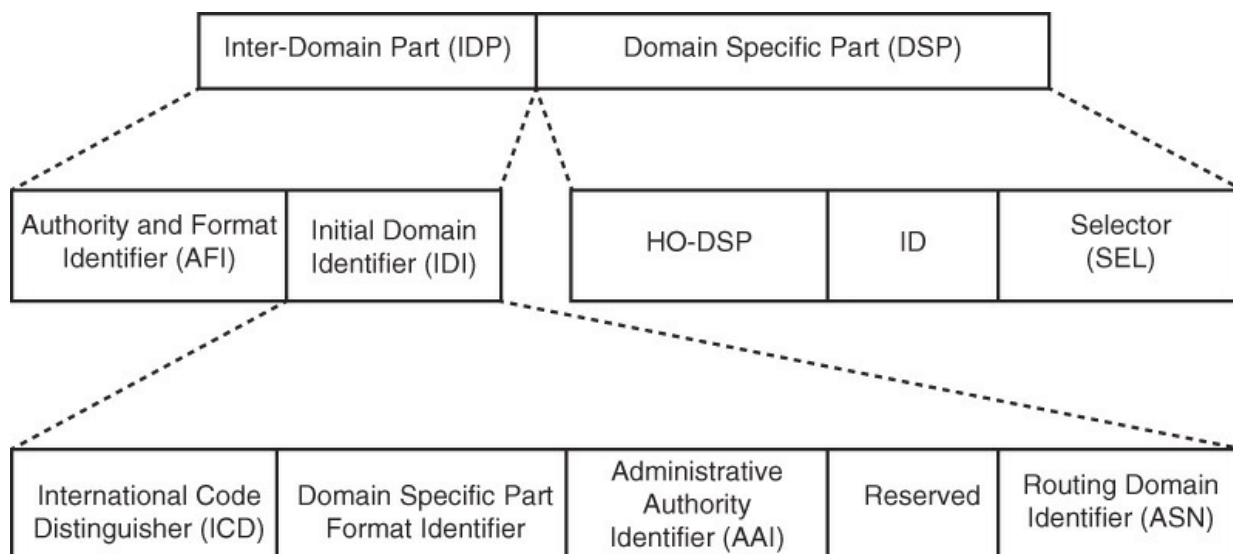


Figure 8-8 IDP Expansion of OSI NS(eAP)



## Domain Specific Part

The DSP portion of the NSAP address contains the High Order DSP (HO-DSP); ID, also known as station ID or system ID; and Selector (SEL) fields. The DSP is configurable by the organization after they receive the IDP portion of the NSAP address. The HO-DSP is a variable-length field that is used to further separate IS-IS routers into logical areas. All routers within the same area maintain an identical topology (LSPDB).

The system ID can be between 2 and 8 bytes in length per the OSI specification, but most networking vendors have standardized on using a 6-byte system ID. System IDs must be the same length, and unique within an area. Some common techniques for creating system IDs are as follows:

- **Incremental numbers:** Start at 0000.0000.0001, then 0000.0000.0002, then 0000.0000.0003, and so on.
- **MAC address:** Select a MAC address from the router. The downside for this approach is that the NSAP address may need to change if there is a hardware failure.
- **IPv4 address:** Use a manipulated form of an IPv4 address taken from a loopback interface. Each octet of the IP address should be three digits with 0s prepended as necessary, and then the octets are separated into four digits. For example, the IP address 192.168.1.1 will become 192.168.001.001, and then the system ID would be 1921.6800.1001.

The SEL is the selector and indicates the transport layer used. RFC 941 includes an addendum to the NSAP address to remove confusion around the OSI addressing scheme. It states that a selector value of 0x00 (00) indicates the router itself. Any NSAP address ending with a 00 is known as the *Network Entity Title* (NET). The NET address is a subset of the NSAP address.

## NET Addressing

IS-IS routers share an area topology through link-state packets (LSPs) that allows them to build an LSPDB. IS-IS uses NET addresses to build the LSPDB topology. The NET address is included in the IS header for all the LSPs. This section provides clarification and examples of commonly used NET addresses.

The dynamic length in the IDP portion of the NET address causes unnecessary confusion. Instead of reading the NET address left to right, most network engineers read the NET address from right to left. In the most simplistic form, the first byte will always be the SEL (with a value of 00), with the next 6-bytes as the system ID, and the remaining 1 to 13 bytes are the Area Address, as shown in [Figure 8-9](#).

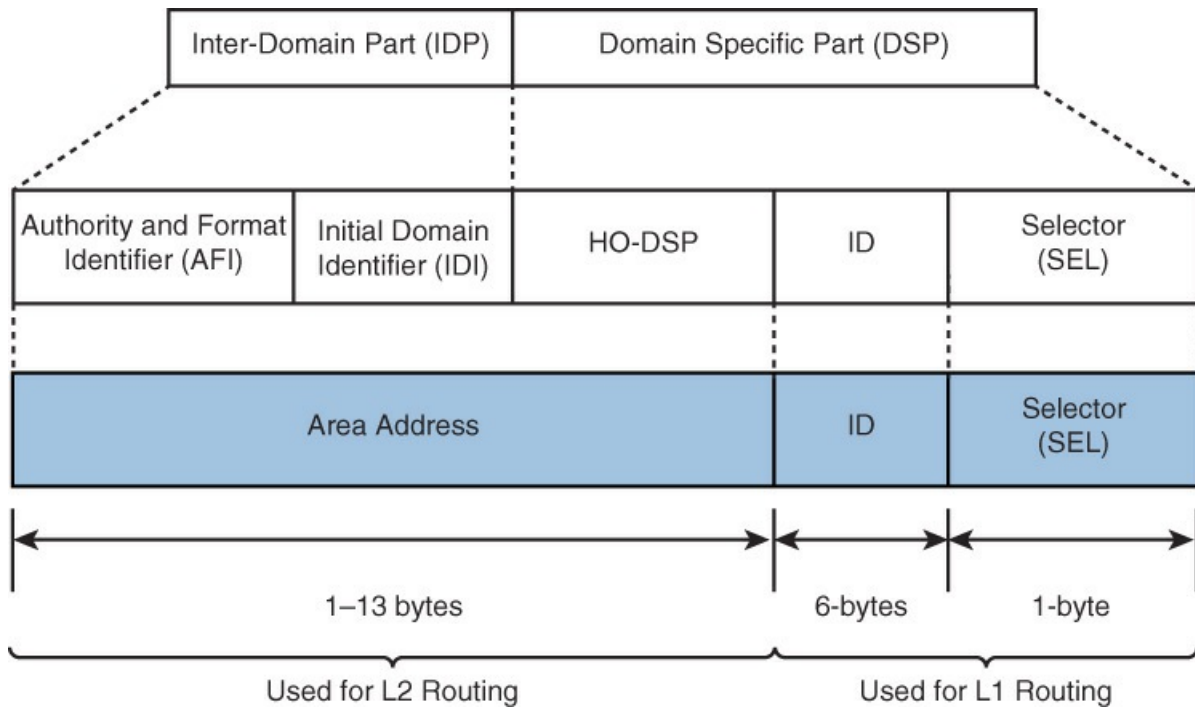


Figure 8-9 Expanded NSAP Address Structure

Figure 8-10 demonstrates a simple NET with 8 bytes. The AFI is not needed because the length does not enter into the IDP portion of the NSAP address. Notice that the Area Address is 07, providing up to 256 unique areas.



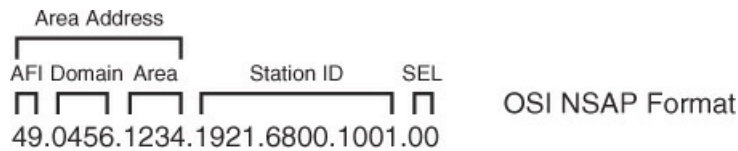
Figure 8-10 Minimal NET Format

Figure 8-11 demonstrates a common NET consisting of 10 bytes. The private AFI (49) is used, and the area uses 2 bytes, providing up to 65,535 unique areas. Notice that the Area Address is 49.1234.



Figure 8-11 Common Private NET

Figure 8-12 demonstrates a typical OSI NSAP address that includes the domain address. Notice that the Area Address is 49.0456.1234 and that the private AFI (49) is still used. If an AFI value besides 49 is selected, the IDP would need to be issued from the appropriate addressing authority.



**Figure 8-12** OSI NSAP Format with Private AFI

NET guidelines include the following:

- Two adjacent IS-IS routers that have the same area addresses belong to the same area.
- Every device in the same area must have a unique system ID.
- IS-IS routers can have multiple NET addresses assigned to them but can have only one system ID.

For simplicity, *area* will reference the *area address* throughout this book with regard to IS-IS.

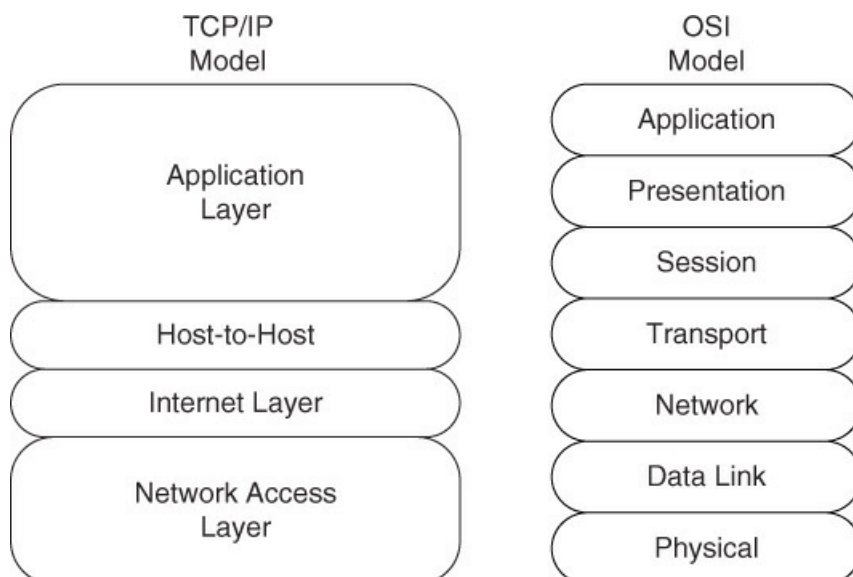
Note

Having multiple NET addresses allows the router to be associated with more than one L1 area. In essence, all the areas merge on the router allowing LSPs to be advertised between area boundaries. Using more than one NET address is recommended only for temporary usage during mergers and reconfiguration. IS-IS allows a router to belong to three areas by default.

## INTER-ROUTER COMMUNICATION

IS-IS was designed to operate in a pure CLNS environment and operates at the data link layer of the OSI model. Eventually the IS-IS routing protocol was modified to support the IP protocol and is called *Integrated IS-IS*. Integrated IS-IS is capable of routing packets for the OSI model or the TCP/IP model.

Figure 8-13 shows the TCP/IP model in alignment with the OSI model. The TCP/IP model was developed independently from the OSI model and consists of four layers, with some layers providing functionality of multiple layers of the OSI model. Notice that TCP/IP's Host-to-Host layer aligns with OSI's transport layer, and that TCP/IP's Internet layer corresponds with OSI's network layer.



**Figure 8-13** TCP/IP to OSI Model Mapping



For the remainder of this book, Integrated IS-IS will be referred to as *IS-IS*. IS-IS supports variable-length subnet masking (VLSM), which supports classless routing of networks with variable prefix lengths. IS-IS provides support for both IPv4 and IPv6 protocols.

Unlike other routing protocols, intermediate system (IS) communication is protocol independent because inter-router communication is not encapsulated in the third layer (network) of the OSI model. IS communication uses the second layer of the OSI model. IP, IPv6, and other protocols all use the third layer addressing in the OSI model.

IS PDUs (packets) follow a common header structure that identifies the type of the PDU. Data specific to each PDU type follows the header, and the last fields use optional variable-length fields that contain information specific to the IS PDU type.

IS packets are categorized into three PDU types, with each type differentiating between L1 and L2 routing information:

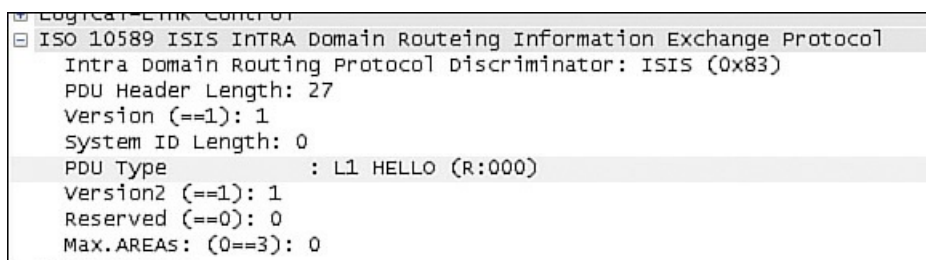
- **Hello packets:** Hello packets are responsible for discovering and maintaining neighbors.
- **Link-state packets (LSPs):** Provide information about a router, and associated networks. Similar to an OSPF LSA, except OSPF uses multiple LSAs.
- **Sequence number packets (SNPs):** Sequence number packets (SNPs) control the synchronization process of LSPs between routers.

Complete sequence number packets (CSNPs) provide the LSP headers for the LSPDB of the advertising router to ensure the LSPDB is synchronized.

Partial sequence number packets (PSNPs) acknowledge receipt of an LSP on point-to-point networks, and request missing link-state information when the LSPDB is identified as being out of sync.

### IS Protocol Header

Every IS packet includes a common header that describes the PDU. All eight fields are 1-byte long and are in all packets. Figure 8-14 shows the IS header from a packet capture.



```
Logical Link Control
ISO 10589 ISIS InTRA Domain Routing Information Exchange Protocol
  Intra Domain Routing Protocol Discriminator: ISIS (0x83)
  PDU Header Length: 27
  Version (==1): 1
  System ID Length: 0
  PDU Type      : L1 HELLO (R:000)
  Version2 (==1): 1
  Reserved (==0): 0
  Max.AREAS: (0==3): 0
```

Figure 8-14 IS Packet Header

Table 8-1 provides an explanation for the fields listed in the IS protocol header.

Field	Description
Intra Domain Routing Protocol Discriminator (Protocol Identifier)	Network layer identifier assigned by ISO. IS-IS communication uses 0x83. ES-IS communication uses 0x81.
PDU Header Length	Length of PDU header because it is dynamic in nature.
Version	Protocol Version Identifier.
System ID Length	A value between 1 and 8 bytes. 6-bytes is the default. A value of 255 indicates a null value.
PDU Type	1-byte representation of the PDU type. Indicates whether it is a hello, LSP, or SNP.
Reserved	Indicates the level of the packet. A value of 1 indicates Level 1 only; 2 indicates L2 with manual mode.
Max Areas	Value between 1 and 254 to represent the number of areas that a router will support. Default value is 3.

**Table 8-1** IS-IS Packet Types

ISO 10589 states that a value of 0 in the IS packet header is treated in a special way for the IS Type, LSPF Database Overload Bit, and Maximum Area Addresses field. A value of 0 infers the default setting indicated in [Table 8-1](#).

### TLVs

A portion of IS PDUs uses variable modules that contain routing information. Each module specifies the type of information, length of data, and the value itself (commonly referred to as *Type, Length, Value* (TLV) tuples. Every TLV maintains a 1-byte numeric label to identify the type (function) and length of the data. Data within the TLV can be up to 254 bytes in size. TLVs support the capability of nesting, so a sub-TLV can exist inside another TLV.

TLVs provide functionality and scalability to the IS protocol. Developing new features for the IS protocol involves the addition of TLVs to the existing structure. For example, IPv6 support was added to the IS protocol by adding TLV 232 (IPv6 Interface Address) and 236 (IPv6 Reachability).

When reading an IS PDU, if an unrecognizable TLV is encountered, the router skips the TLV and continues to the next supported TLV. As the PDUs that are flooded to other IS nodes, TLVs are not modified, so if a transit router does not understand a TLV, the information is still relayed to other IS routers that may understand the TLV.

### IS PDU Addressing

Communication between IS devices uses Layer 2 addresses. The source address is always the network interface's Layer 2 address, and the destination address varies depending on the network type.

ISO standards classify network media into two categories: Broadcast and General Topology.

Broadcast networks provide communication to multiple devices with a single packet. Broadcast interfaces communicate in a multicast fashion using well-known Layer 2 addresses so that only the nodes running IS-IS process the traffic. IS-IS does not send unicast traffic on broadcast network types because all routers on the segment should be aware of what is happening with the

network.

Table 8-2 lists the destination MAC addresses used for IS communication.

Name	Destination MAC Address
All L1 ISs	0180.c200.0014
All L2 ISs	0180.c200.0015
All ISs	0900.2b00.0005
All ESs	0900.2b00.0004

Table 8-2 IS-IS Destination MAC Addresses

General topology networks are based on network media that allows communication only with another device if a single packet is sent out. General topology networks are often referred to in IS-IS documentation as *point-to-point networks*. P2P networks communicate with a directed destination address that matches the Layer 2 address for the remote device. NBMA technologies such as Frame Relay may not guarantee communication to all devices with a single packet. A common best practice is to use P2P subinterfaces on non-broadcast multi-access (NBMA) technologies to ensure proper communication between IS-IS nodes.

#### Hello Packets

IS-IS hello (IIH) packets are responsible for discovering and maintaining neighbors. IS communication has five different types of hellos, listed in Table 8-3. Only the first three are involved with IS-IS neighbor adjacencies, and the other two are related to ES-IS communication.

Type	Description
L1 IS-IS hello (IIH) PDU Type 15	Discovers, forms, and maintains L1 IS-IS neighbors
L2 IS-IS hello (IIH) PDU Type 16	Discovers, forms, and maintains L2 IS-IS neighbors
P2P hello (IIH) IS-IS PDU Type 17	Discovers, forms, and maintains P2P IS-IS neighbors
End system hello (ESH)	Used for end systems (ESs) to discover intermediate systems (ISs) and vice versa; similar to Internet Control Message Protocol (ICMP)
Intermediate system hello (ISH)	Used for ESs to discover ISs and vice versa for router selection

Table 8-3 IS Hello Types

Routers that form an L1-L2 adjacency with another IS router will send both L1 and L2 IIHs on broadcast links. To save bandwidth on WAN links, P2P links will use the P2P hello, which services both L1 and L2 adjacencies.

Table 8-3 provides a brief overview of the five IS hello packet types.

Figure 8-15 provides a packet capture of an L1 IIH. Notice the multiple padding TLVs in the IIH

PDU.

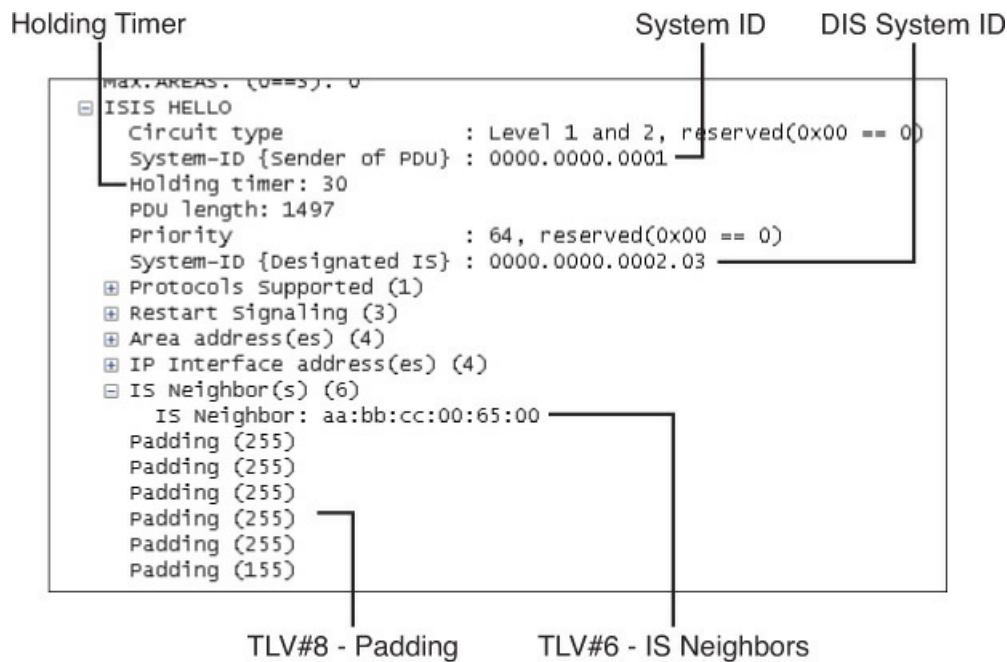


Figure 8-15 IS-IS Hello

Table 8-4 provides a brief list of information included in the IIH hello packet.

Type	Description
Circuit Type	0x1 L1 only 0x2 L2 only 0x3 L1 and L2
System-ID	System ID of router sending the IIH
Holding Timer	Holding timer to be used for this IS
PDU Length	Entire length of the PDU
Priority	Router interface priority for designated intermediate system (DIS) elections (This is not included on P2P IIHs.)
System-ID of DIS	System ID of DIS for the broadcast segment (This is not included on P2P IIHs.)

Table 8-4 Fields in IIH Packets

Table 8-5 provides a list of common TLVs found in IS hello PDUs.

TLV Number	Name	Description
1	Area Addresses	List of area addresses from advertising router.
6	IS Neighbors	List of subnetwork point of addresses (SNPA) from a neighboring IS router. SNPA is the Layer 2 hardware addresses for IS-IS routers. (Not included used on P2P hellos.)
8	Padding	TLVs used to inflate the packet to full MTU. Data within this TLV is ignored.
10	Authentication	Identifies the type of authentication and includes the plain-text password or the message digest 5 (MD5) hash.
129	Protocols Supported	Protocols that the transmitter supports (CLNS only, IP only, or both).
132	IP Interface Addresses	List of IP addresses from the transmitting interface, which includes secondary IP addresses.
240	Adjacency State	Used by P2P links to ensure three-way IS handshakes.

**Table 8-5** Common TLVs found in IIH PDUs

### Link-State Packets

LSPs are similar to OSPF LSAs, where they advertise neighbors and attached networks, except that IS-IS uses only two types of LSPs. IS-IS defines an LSP type for each level. L1 LSPs are flooded throughout the area from which they originate, and L2 LSPs are flooded throughout the L2 network. Table 8-6 lists the LSP types used by IS-IS.

Type	Description
L1 LSP PDU Type 18	Provides a list of adjacent IS routers and attached networks to IS-IS routers in the L1 area.
L2 LSP PDU Type 20	Provides a list of adjacent IS routers and attached networks to IS-IS routers in the L2 area.

**Table 8-6** IS-IS LSP Types

### LSP Lifetime

All LSPs include a 16-bit Lifetime field that acts as a secondary safety mechanism in the event the LSP was not purged when a router fails. The LSP lifetime works in a similar fashion to the OSPF LSA age explained in the previous chapter.

The advertising router populates the LSP Lifetime field with the advertising router's MaxAge value. The MaxAge default value defaults to 20 minutes (1200 seconds). The originating router will advertise a new LSP every 15 minutes (900 seconds) by default. Upon receipt of the LSP, the lifetime decrements by 1 every second. When the lifetime reaches 0, the expired LSAs are kept in the system for an additional 60 seconds (ZeroAgeLifetime) before the LSP is removed.

### LSP ID

The LSP ID is a fixed 8-byte field that provides a unique identification of the LSP originator. The LSP ID consists of the following:

- **System ID (6 bytes):** The system ID is extracted from the NET address configured on the router.

■ **Pseudonode ID (1 byte):** The pseudonode ID identifies the LSP for a specific pseudonode (virtual router) or for the physical router. LSPs with a pseudonode ID of 0 describes the links from the advertising router’s perspective, and can be called *non-pseudonode LSPs*. LSPs with a non-0 number indicate that the LSP is pseudonode LSP. The pseudonode ID will correlate to the router’s circuit ID for the interface performing the DIS function. If a router acts as a DIS for multiple broadcast segments, then the pseudonode ID is unique for each of the DIS broadcast segments for that level. In other words, the pseudonode ID will not repeat itself on the same router for a specific IS-IS level. Pseudonodes and DIS are explained later in this chapter.

■ **Fragment ID (1 byte):** If an LSP is larger than the maximum transmission unit (MTU) value of the interface it needs to be sent out of, that LSP must be fragmented. IS-IS fragments the LSP as it is created, and the fragment ID allows the receiving router to process fragmented LSPs.

Note

Only the LSPs originating router is allowed to fragment the LSP, other routers cannot fragment that LSP. LSPs that are larger than an interface’s configured MTU are not fragmented like an IP packet but are dropped instead. The default LSP size works well in most networks, but if you modify the default LSP MTU, examine the network to ensure that all interfaces are capable of supporting the chosen LSP MTU size. All routers participating in the IS-IS routing domain should have the same LSP MTU size configured.

Figure 8-16 shows two LSP IDs. The LSP ID on the left indicates that it is for a specific IS router, and the LSP ID on the right indicates that it is for the DIS because the pseudonode ID is not 0.

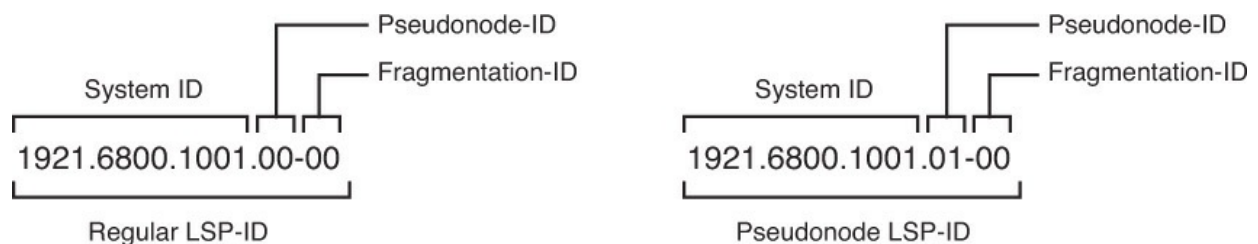


Figure 8-16 LSP ID Structure

**LSP Sequence**

LSPs use a 32-bit sequence number to control versioning. An IS router increments the sequence number by 1 every time it floods an LSP. A router will process only LSPs that contain a sequence number greater than the one in the LSPDB. The concept is similar to the sequence number with OSPF LSAs explained in the previous chapter.

An IS-IS router will process only an LSP with a sequence greater than the 1 in the LSPDB, and if an entry does not exist, the LSP is added to the LSPDB. This ensures that LSPs are processed in the order that they were generated and prevents older LSPs from being processed.

**Attribute Fields**

The last portion of the LSP header is an 8-bit section that references four components of the IS-IS specification:

■ **Partition bit:** The partition bit identifies whether a router supports the capability for partition repair. Partition repair allows a broken L1 area to be repaired by L2 routers that belong to the same area as the L1 routers. Cisco and most other network vendors do not support partition repair.

■ **Attached bit:** The next 4 bits reflect the attached bit set by an L1-L2 router connected to other

areas via the L2 backbone. The IS-IS specification provides support for multiple topologies but has deprecated to a default context and only 1 bit, the default bit (bit 4). The attached bit is in L1 LSPs.

■ **Overload bit:** The overload bit indicates when a router is in an overloaded condition. During SPF calculation, routers should avoid sending traffic through this router. Upon recovery, the router advertises a new LSP without the overload bit, and the SPF calculation occurs normally without avoiding routes through the previously overloaded node.

■ **Router type:** The last 2 bits indicate whether the LSP is from an L1 or L2 router.

### LSP Packet and TLVs

Figure 8-17 provides a packet capture of an L1 LSP. The LSP remaining lifetime, sequence number, and LSP attribute fields are identified. Notice that the IP Internal Reachability TLVs contain additional information in them.

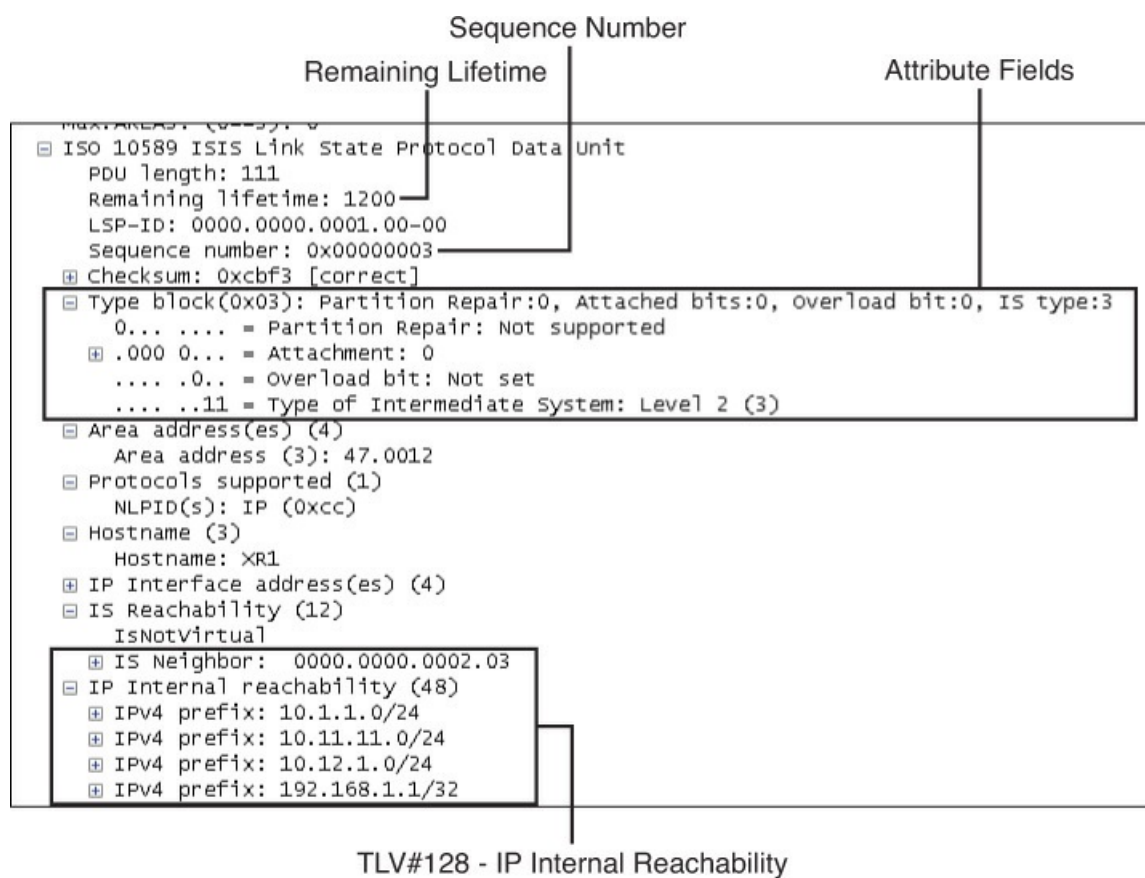


Figure 8-17 OSI NSAP Format with Private AFI

Table 8-7 provides a list of common TLVs used in L1 and L2 LSPs.

<b>TLV Number</b>	<b>Name</b>	<b>Description</b>
1	Area Addresses	List of area addresses on the configured router.
2	IS Neighbors	List of subnetwork point of addresses (SNPA) from a neighboring IS router and associated interface metric. SNPA is the Layer 2 hardware addresses for IS-IS routers.
10	Authentication	Identifies the type of authentication and includes the plaintext password or the MD5 hash.
128	IP Internal Reachability Information	List of internal IS-IS network and interface metric for the advertising router.
129	Protocols Supported	Protocols that the advertising router supports (CLNS only, IP only, or both).
130	IP External Reachability Information	List of external (redistributed) networks and metrics associated when redistributed into IS-IS. Metrics can be internal or external but are external by default.
132	IP Interface Addresses	List of IP addresses from the transmitting interface, which includes secondary IP addresses (limited to 63 IP addresses within the TLV).
137	Hostname	Router hostname so that it can be used to identify the router in lieu of the system ID.

**Table 8-7** Common TLVs Found in LSP PDUs

### **IS-IS Neighbor**

IS-IS requires that neighboring routers form an adjacency before LSPs are flooded or processed from nearby IS routers. The IS-IS neighbor adjacency process is simpler than OSPF and consists of three states: down, initializing, and up.

The method that IS-IS uses to become adjacent with another node varies if the network is a broadcast or a P2P link. The following section explains both processes.

### **Ethernet**

Figure 8-18 provides a brief overview of the IS-IS adjacency process for broadcast network segments.



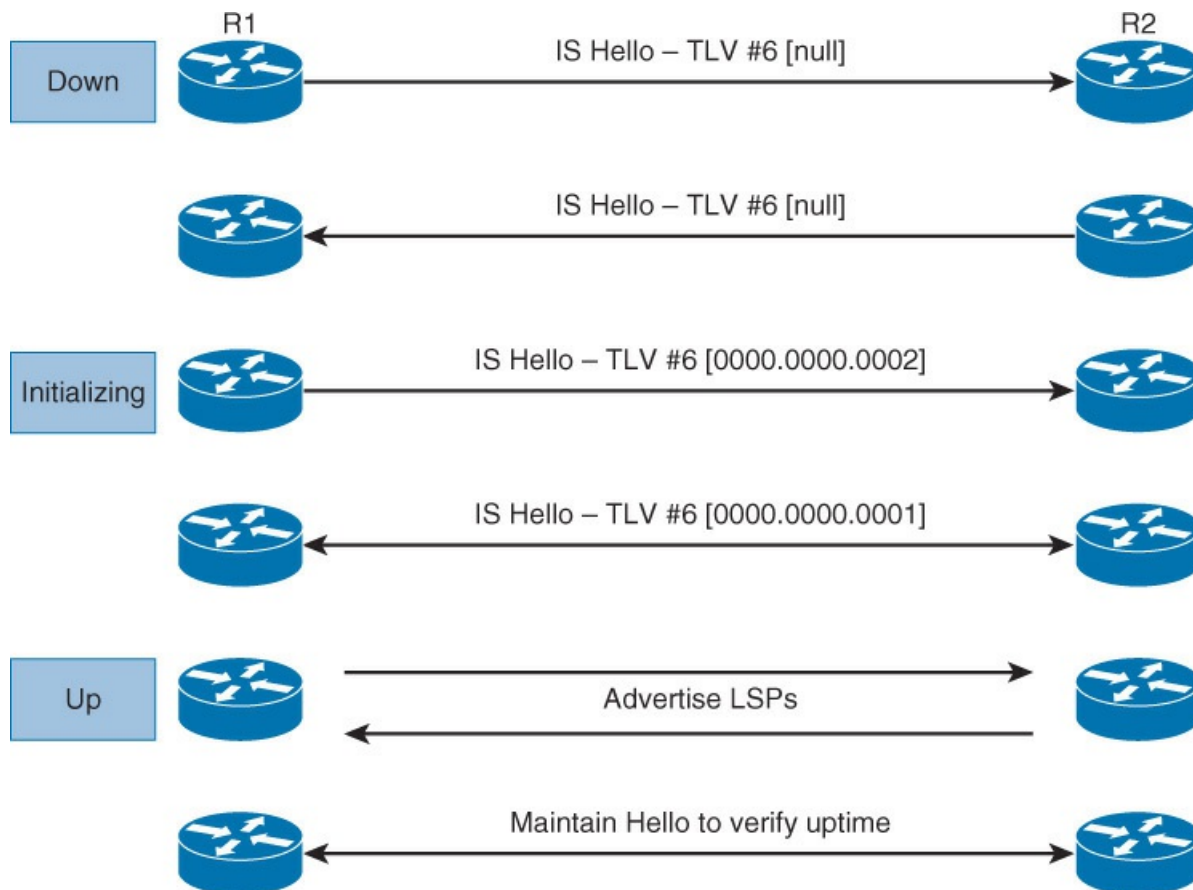


Figure 8-18 IS-IS Broadcast Adjacency Process

Figure 8-19 demonstrates two IS-IS routers trying to form a neighbor adjacency for Area 49.0012. R1 and R2 share a common network link (10.12.1.0/24) and have enabled IS-IS for all interfaces including their loopback addresses. The process for forming an L1 IS-IS adjacency follows.

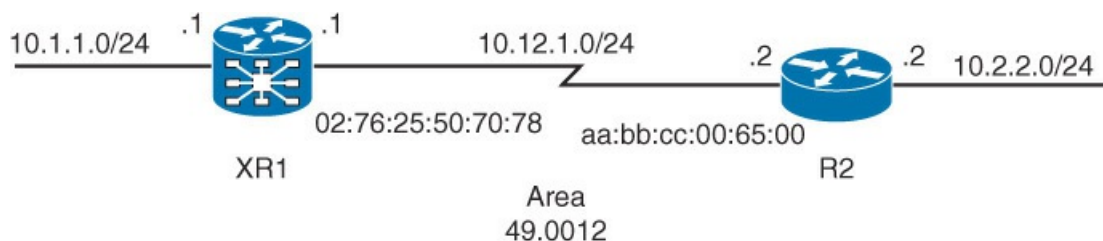
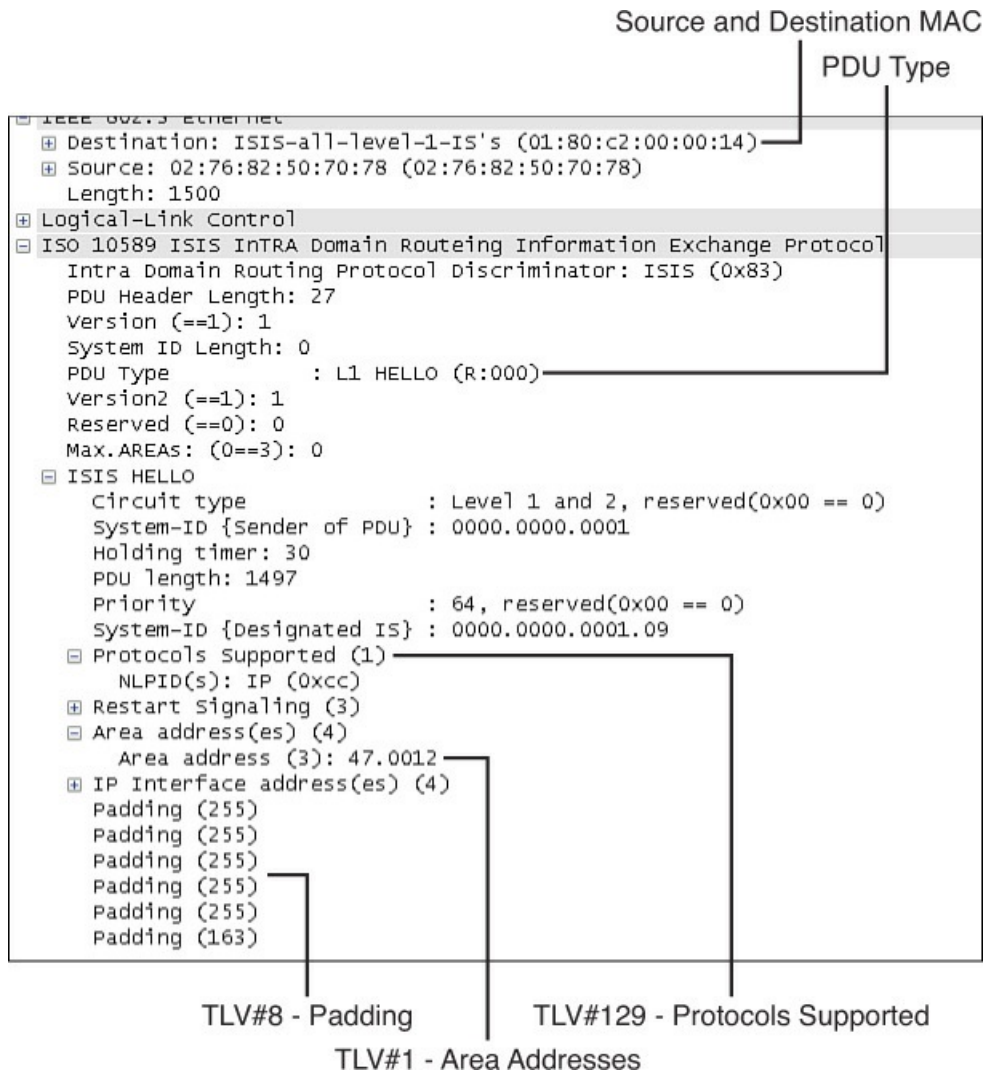


Figure 8-19 Simple IS-IS Topology

**Step 1.** XR1 sends out an L1 hello packet. The Active Neighbor (TLV 6) is not included in the IIH in Figure 8-20. Notice that the destination MAC address uses the L1 MAC address 01:80:C2:00:00:14.



**Figure 8-20** R1's L1 III Hello with No Neighbors Detected

**Step 2.** R2 sends out an L1 hello packet. Notice that the Active Neighbor TLV is still not included in [Figure 8-21](#).

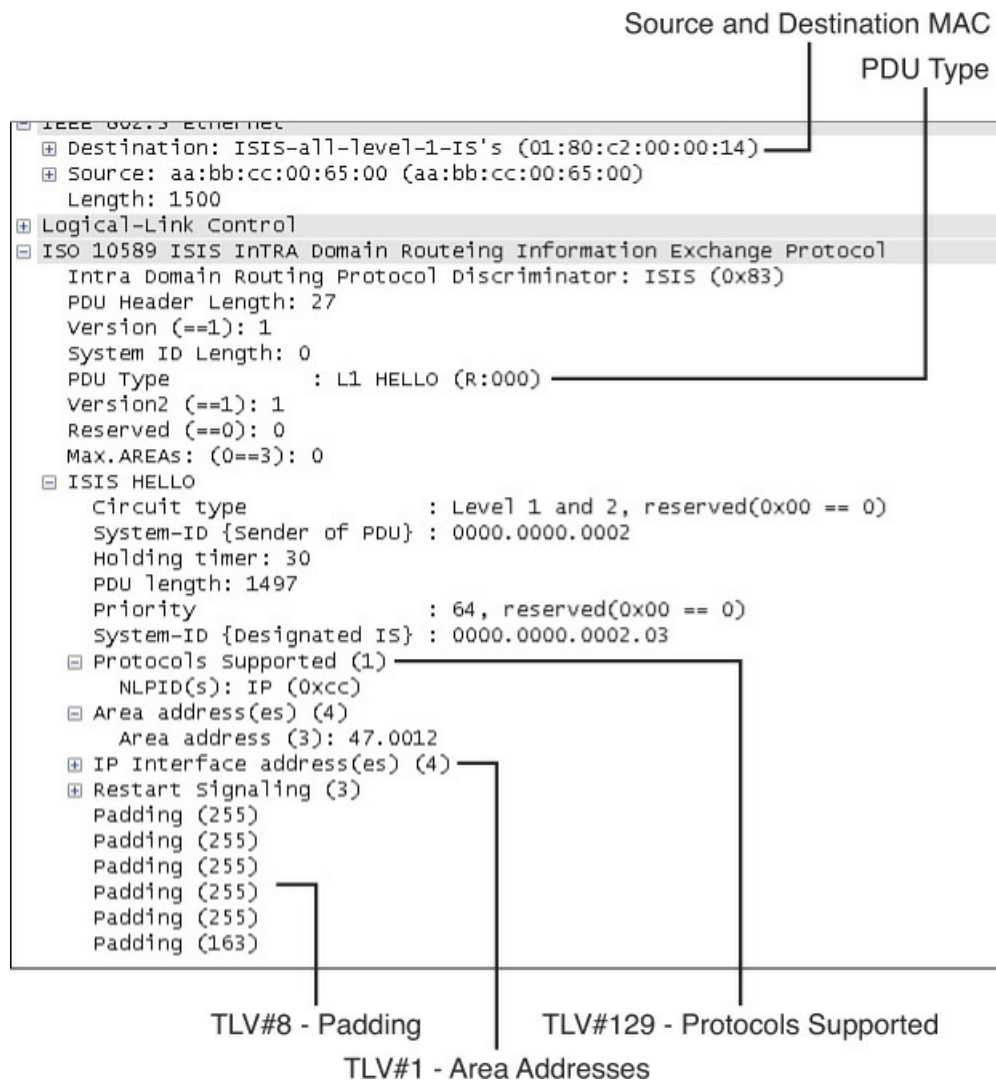


Figure 8-21 R2's L1 IIH Hello with No Neighbors Detected

**Step 3.** XR1 receives R2's IIH and verifies that the following parameters match:

- L1 routers can form adjacencies with L1 or L1-L2 routers but not L2.
- L2 routers can form adjacencies with L2 or L1-L2 routers but not L1.
- Authentication type and credentials (if any).
- L1 adjacencies require the area address to match.
- The system ID must be unique within the same area address.
- Ensures that the protocols supported (TLV 129) match (OSI, IP, or both).
- Ensures that the interfaces share a common subnet.
- MTU matches. IS-IS verifies MTU by inflating the size of the packet with multiple padding TLVs (8). The PDU length in the hello must match.

**Step 4.** Upon verifying that the listed parameters pass, XR1 creates a new entry in the neighbor

table and sets the state to *initializing* for the adjacency type (L1 or L2) until R1 receives an IIH hello with its SNPA in the IS Neighbors TLV.

**Step 5.** Figure 8-22 shows XR1 advertising the L1 IIH hello with R2's SNPA and completing the three-way handshake.

```
Max. AREAS: (0-3). 0
  ISIS HELLO
    Circuit type          : Level 1 and 2, reserved(0x00 == 0)
    System-ID {Sender of PDU} : 0000.0000.0001
    Holding timer: 30
    PDU length: 1497
    Priority               : 64, reserved(0x00 == 0)
    System-ID {Designated IS} : 0000.0000.0002.03
  Protocols Supported (1)
    NLPID(s): IP (0xcc)
  Restart Signaling (3)
  Area address(es) (4)
    Area address (3): 47.0012
  IP Interface address(es) (4)
  IS Neighbor(s) (6)
    IS Neighbor: aa:bb:cc:00:65:00
    Padding (255)
    Padding (255)
    Padding (255)
    Padding (255)
    Padding (255)
    Padding (155)
```

TLV#2 IS Neighbors

Figure 8-22 R2's L1 IIH Hello with IS-IS Neighbors Detected

**Step 6.** IS-IS has completed the three-way handshake, both routers set the neighbor state to *up* for the adjacency type (L1 or L2), and then each router advertise LSPs to each other.

Figure 8-23 provides the packet capture of R2's LSP. The LSP ID, hostname, IP reachability, and IS neighbors data has been highlighted. Notice that only the default metric is included with the IP Reachability and IS Neighbors TLV.

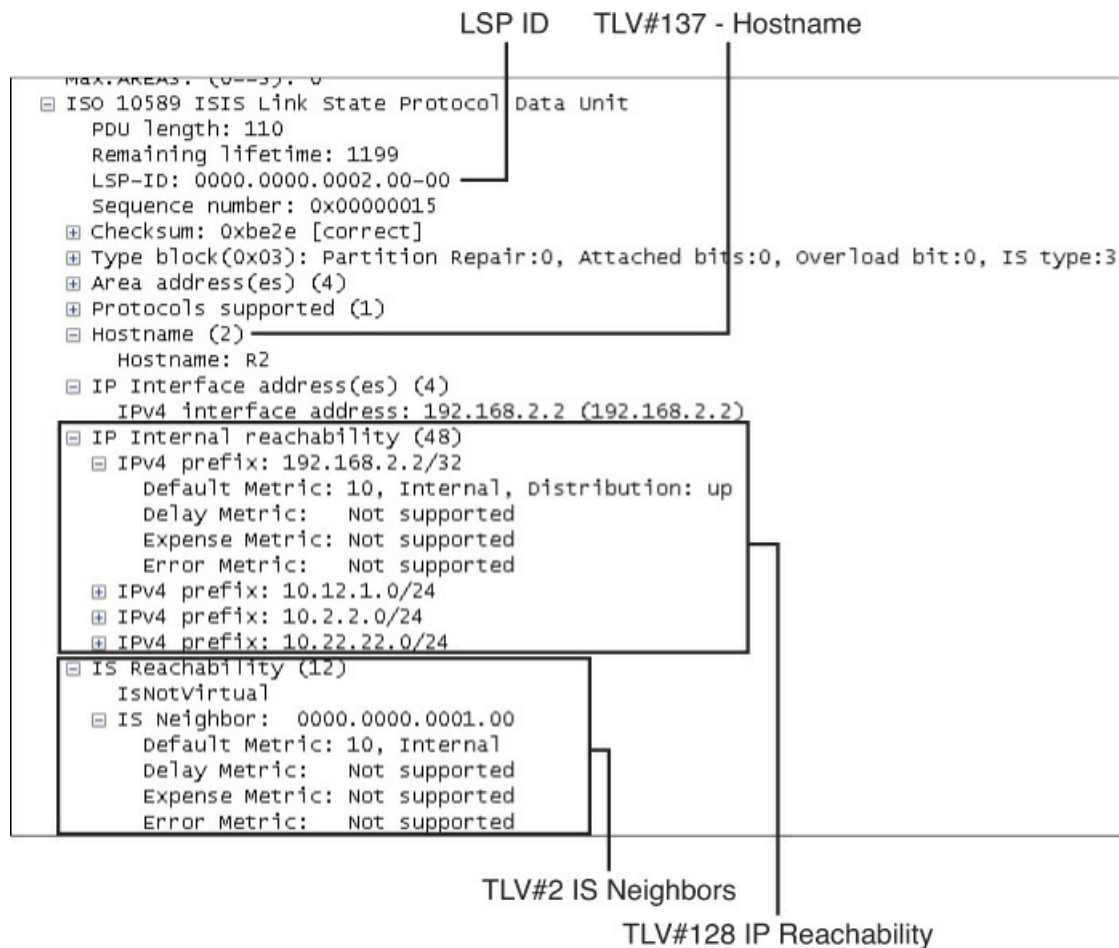


Figure 8-23 R2's L1 LSP

**Step 7.** Both routers have fully loaded their LSPDB and start to run the Dijkstra Shortest Path First (SPF) algorithm. Each router constructs a SPF tree and installs routes into the Routing Information Base (RIB).

#### Point-to-Point

P2P links use the same basic concept as broadcast interfaces with the following exceptions:

- Two-way IS handshakes were originally specified in the ISO 10589, but support for three-way IS handshakes was added in RFC 5303 through the addition of TLV 240 to track a neighbor status to prevent certain false adjacencies from forming. A router that does not support TLV 240 can form an adjacency with a two-way IS handshake.
- Each router sends a PSNP to acknowledge the receipt of an LSP.

Figure 8-24 provides a simple topology of two IS-IS routers trying to form a neighbor adjacency for Area 49.0012. R1 and R2 connect via a serial link (10.12.1.0/24). The process for forming an L1 IS-IS adjacency follows.

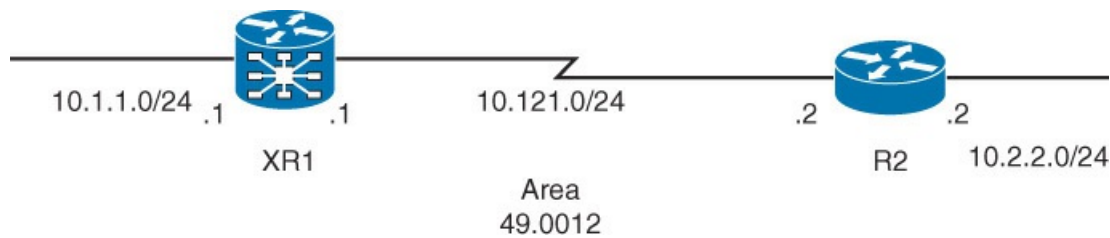


Figure 8-24 Simple IS-IS Topology

**Step 1.** R1 sends out a P2P hello packet. TLV 240 is set to an *adjacency down* state and includes the circuit ID.

**Step 2.** R2 sends out a P2P hello packet. TLV 240 is set to an *adjacency down* state circuit ID.

**Step 3.** R1 receives R2's P2P hello and verifies that the following parameters match:

- L1 routers can form adjacencies with L1 or L1-L2 routers, but not L2.
- L2 routers can form adjacencies with L2 or L1-L2 routers, but not L1.
- Authentication type and credentials (if any).
- L1 adjacencies require the area address to match.
- The system ID must be unique within the same area address.
- Ensures that the protocols supported (TLV 129) match (OSI, IP, or both).
- Ensures that the interfaces share a common subnet.
- MTU matches. IS-IS verifies MTU by inflating the size of the packet with multiple padding TLVs (8). The PDU length in the hello must match.

**Step 4.** Upon verifying the parameters pass, R1 creates a new entry in the neighbor table, and sends out a P2P hello packet. TLV 240 is set to an *initializing* state with R2's circuit ID moving the IS handshake into the second stage. R2 performs the same step.

**Step 5.** R2 receives the P2P hello packet, verifies the circuit ID is correct, and moves the neighbor table into an up state, completing the three-way IS handshake. R1 performs the same step.

**Step 6.** Both routers send a CSNP (LSP headers) to the remote router followed by the LSPs.

**Step 7.** A PSNP is sent acknowledging each of the LSPs that were sent out. A PSNP can acknowledge multiple LSPs in one PDU.

**Step 8.** Both routers have fully loaded their LSPDB and start to run the Dijkstra SPF algorithm. Each router constructs a SPF tree and installs routes into the RIB.

## BASIC IS-IS CONFIGURATION

IS-IS configuration is straightforward with some minor differences between IOS and IOS XR. Interfaces are explicitly selected for both operating systems, and all IP addresses (including secondary) are included in the IS-IS LSPDB. IOS and IOS XR try to form L1 and L2 adjacencies by default for all interfaces.

A router can run multiple IS-IS processes as long as each process uses a unique system ID. Each process maintains its own databases, and routes learned in one IS-IS process are not available to a different IS-IS process. The IS-IS processes are locally significant and do not have to match among peering routers.

### IOS

IOS configuration of the IS-IS routing protocol consists of defining the NET under the IS-IS routing process and enabling interfaces with interface parameter commands, the steps for which are outlined as follows:

#### Step 1. Create the IS-IS routing process.

Initialize the IS-IS process with the global configuration command **router isis** [*instance-id*].

#### Step 2. Identify the IS-IS NET.

Specify the NET for the IS-IS process with the IS-IS router configuration command **net** *network-entity-title*.

#### Step 3. Activate IS-IS on the interface.

Activate IS-IS with the interface parameter command **ip router isis** [*instance-id*].

If the *instance-id* is not specified during IS-IS configuration, IOS will use a null value for the *instance-id*. The *instance-id* must match between the process and any interface parameter commands for the two components to link to each other.

#### Note

IOS does not log IS-IS neighbor adjacencies forming or dissolving by default. The IS-IS router configuration command **log-adjacency-changes** will enable logging on IOS nodes, and is recommended.

### IOS XR

IOS XR's protocol configuration is hierarchical and allows for all the configuration to exist under the IS-IS process. IOS XR configuration consists of the following steps:

#### Step 1. Create the IS-IS routing process.

Initialize the IS-IS process with the global configuration command **router isis** *instance-id*.

#### Step 2. Identify the IS-IS NET.

Specify the NET for the IS-IS process with the IS-IS router configuration command **net** *network-entity-title*.

#### Step 3. Identify the interface.

Select the interface with the IS-IS router configuration command **interface** *interface-type interface-number*.

#### Step 4. Activate the address family.

Activate the IPv4 address family with the command **address-family ipv4 unicast** underneath the interface within the IS-IS router configuration.

##### Note

By default, IOS XR does not log IS-IS neighbor adjacencies forming or dissolving. The IS-IS router configuration command **log adjacency changes** will enable logging on IOS XR nodes, and is recommended.

#### Sample Topology and Configuration

Figure 8-25 provides a topology example to illustrate a basic IS-IS configuration. All three routers will have loopback IP addresses to match their node numbers (192.168.1.1, 192.168.2.2, and so on) and will use the IS-IS NET area of 49.0123. The system ID will use 0000.0000.0001, 0000.0000.0002, and so on, where the last digits reflect the router number.

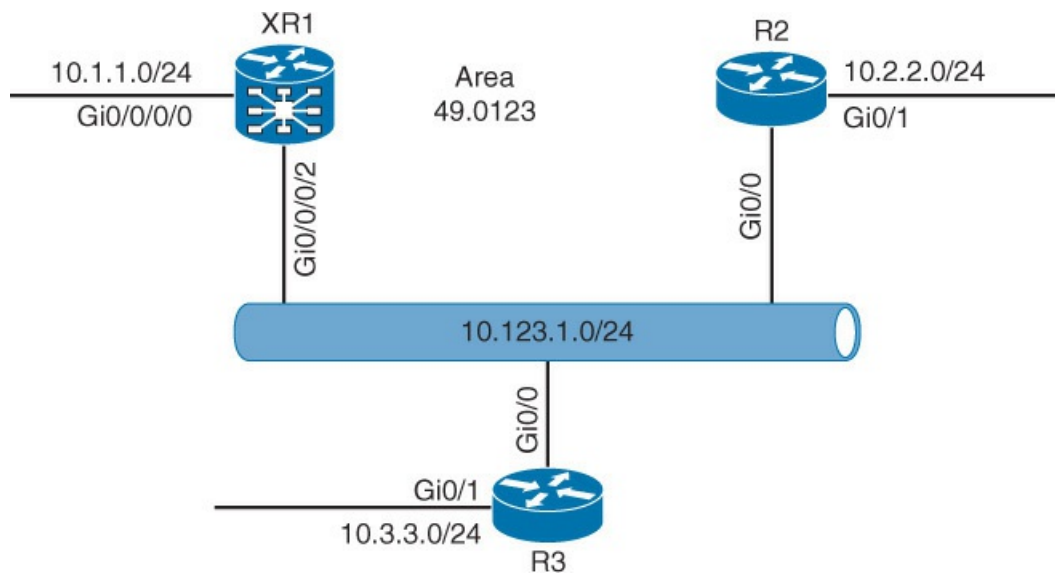


Figure 8-25 IS-IS Topology Example

All three nodes enable all interfaces for IS-IS. R2 will configure IS-IS using the null instance. XR1 and R3 will use the instance ID of CISCO. IS-IS should log neighbor adjacencies on all three routers. Example 8-1 provides the relevant configuration for all three routers.

#### Example 8-1 IS-IS Configuration Topology

[Click here to view code image](#)



**XR1**

```
router isis CISCO
net 49.0123.0000.0000.0001.00
log adjacency changes
interface Loopback0
  address-family ipv4 unicast
  !
  !
interface GigabitEthernet0/0/0/2
  address-family ipv4 unicast
  !
  !
interface GigabitEthernet0/0/0/3
  address-family ipv4 unicast
  !
  !
  !
```

**R2**

```
interface Loopback0
  ip address 192.168.2.2 255.255.255.255
  ip router isis
  !
interface GigabitEthernet0/0
  ip address 10.123.1.2 255.255.255.0
  ip router isis
  !
interface GigabitEthernet0/1
  ip address 10.2.2.2 255.255.255.0
  ip router isis
  !
router isis
net 49.0123.1921.6800.2002.00
log-adjacency-changes
```

**R3**

```
interface Loopback0
  ip address 192.168.3.3 255.255.255.255
  ip router isis CISCO
  !
interface GigabitEthernet0/0
  ip address 10.123.1.3 255.255.255.0
  ip router isis CISCO
  !
interface GigabitEthernet0/1
  ip address 10.3.3.3 255.255.255.0
  ip router isis CISCO

router isis CISCO
net 49.0123.1921.6800.3003.00
log-adjacency-changes
```

## Confirmation of IS-IS Interfaces

It is a good practice to verify that the correct interfaces are running IS-IS after any configuration changes. The IOS command **show clns interface** *[interface-type interface-number]* will list all the interfaces and any relevant information for IS-IS enabled interfaces. Interfaces that do not have IS-IS configured will display *CLNS protocol processing disabled*.

IOS XR will list only IS-IS interfaces by using the command **show isis interface** *[interface-type interface-number]*. Providing a specific interface limits the output to the specified interface.

Some of the output in [Example 8-2](#) has been omitted for brevity, but the following relevant information can be seen in the output:

- The IS-IS interface is operating as an L1-L2 interface (Cisco default).
- Two IS-IS adjacencies have formed at L1 and L2.
- The LAN ID (pseudonode ID) is R3.02.
- L1 and L2 metrics are set to 10 (Cisco default).
- The priority for the interface is 64 for L1 and L2 (Cisco Default). IOS XR shows that the pseudonode has a priority of 64.
- The IS PDU MTU is 1497.

## Example 8-2 IS-IS Protocol Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis interface GigabitEthernet0/0/0/2
```

```
! Output omitted for brevity
```

```
GigabitEthernet0/0/0/2      Enabled
  Circuit Type:              level-1-2
  Media Type:                LAN
  Circuit Number:           3
```

```
Level-1
```

```
  Adjacency Count:         2
  LAN ID:                  R3.02
  Priority (Local/DIS):     64/64
```

```
Level-2
```

```
  Adjacency Count:         2
  LAN ID:                  R3.02
  Priority (Local/DIS):     64/64
```

```
CLNS I/O
```

```
  Protocol State:          Up
  MTU:                     1497
  SNPA:                    0276.8250.7078
```

```
IPv4 Unicast Topology:     Enabled
```

```
  Metric (L1/L2):         10/10
```

```
R2#show clns interface GigabitEthernet0/0
GigabitEthernet0/0 is up, line protocol is up
Checksums enabled, MTU 1497, Encapsulation SAP
ERPDU enabled, min. interval 10 msec.
Routing Protocol: IS-IS
  Circuit Type: level-1-2
  Interface number 0x1, local circuit ID 0x2
  Level-1 Metric: 10, Priority: 64, Circuit ID: R3.02
  DR ID: R3.02
  Level-1 IPv6 Metric: 10
  Number of active level-1 adjacencies: 2
  Level-2 Metric: 10, Priority: 64, Circuit ID: R3.02
  DR ID: R3.02
  Level-2 IPv6 Metric: 10
  Number of active level-2 adjacencies: 2
```

Another technique that lists the IS-IS interfaces and provides an overview of the IS-IS configuration for the router that might seem more efficient uses the command **show clns protocol** on IOS nodes. The equivalent command **show isis protocol** is used on IOS XR nodes.

Example 8-3 provides output for both commands. Notice that the system ID, IS level, area address, and IS-enabled interfaces have been highlighted.

### **Example 8-3** *IS-IS Protocol Verification*

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis protocol
```

```
IS-IS Router: IS-IS
```

```
System Id: 0000.0000.0001
```

```
IS Levels: level-1-2
```

```
Manual area address(es):
```

```
49.0123
```

```
Routing for area address(es):
```

```
49.0123
```

```
Non-stop forwarding: Disabled
```

```
Most recent startup mode: Cold Restart
```

```
Topologies supported by IS-IS:
```

```
IPv4 Unicast
```

```
Level-1
```

```
Metric style (generate/accept): Narrow/Narrow
```

```
Metric: 10
```

```
ISPF status: Disabled
```

```
Level-2
```

```
Metric style (generate/accept): Narrow/Narrow
```

```
Metric: 10
```

```
ISPF status: Disabled
```

```
No protocols redistributed
```

```
Distance: 115
```

```
Advertise Passive Interface Prefixes Only: No
```

```
Interfaces supported by IS-IS:
```

```
Loopback0 is running actively (active in configuration)
```

```
GigabitEthernet0/0/0/2 is running actively (active in configuration)
```

```
GigabitEthernet0/0/0/3 is running actively (active in configuration)
```

```
R2#show clns protocol
```

```
IS-IS Router: <Null Tag>
```

```
System Id: 0000.0000.0002.00 IS-Type: level-1-2
```

```
Manual area address(es):
```

```
49.0123
```

```
Routing for area address(es):
```

```
49.0123
```

```
Interfaces supported by IS-IS:
```

```
GigabitEthernet0/1 - IP
```

```
GigabitEthernet0/0 - IP
```

```
GigabitLoopback0 - IP
```

```
Redistribute:
```

```
static (on by default)
```

```
Distance for L2 CLNS routes: 110
```

```
RRR level: none
```

```
Generate narrow metrics: level-1-2
```

```
Accept narrow metrics: level-1-2
```

```
Generate wide metrics: none
```

```
Accept wide metrics: none
```

## Verification of IS-IS Neighbor Adjacencies

The command **show clns neighbor [detail]** provides the IS-IS neighbor table for IOS nodes, and IOS XR nodes use the command **show isis neighbor [detail]**. The **detail** keyword parameter provides the neighbor's uptime, and any secondary IP addresses configured on the neighboring nodes. [Example 8-4](#) provides output of the nondetailed command on XR1, and R2, and R3.

### Example 8-4 IS-IS Neighbor Output

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis neighbors
```

```
IS-IS ISIS neighbors:
```

System Id	Interface	SNPA	State	Holdtime	Type	IETF-NSF
R2	Gi0/0/0/2	aabb.cc00.6500	Up	25	L1L2	Capable
R3	Gi0/0/0/2	aabb.cc00.6600	Up	7	L1L2	Capable

```
Total neighbor count: 2
```

```
R2#show clns neighbors
```

System Id	Interface	SNPA	State	Holdtime	Type	Protocol
XR1	Gi0/0	0276.8250.7078	Up	29	L1L2	IS-IS
R3	Gi0/0	aabb.cc00.6600	Up	9	L1L2	IS-IS

```
R3#show clns neighbors
```

```
Tag ISIS:
```

System Id	Interface	SNPA	State	Holdtime	Type	Protocol
XR1	Gi0/0	0276.8250.7078	Up	28	L1L2	IS-IS
R2	Gi0/0	aabb.cc00.6500	Up	24	L1L2	IS-IS

```
Tag ISIS\:
```

[Table 8-8](#) provides a brief overview of the fields used in [Example 8-4](#). Notice that the hold time for R3 is relatively low because R3 is the DIS for the 10.123.1.0/24 network.

Field	Description
System ID	The system ID (SEL) abstracted from the NET address
Interface	Interface used to peer with neighbor router
Subnetwork Point of Attachment (SNPA)	Layer 2 hardware addresses for IS-IS routers: Ethernet: MAC address X.25 or ATM: Virtual circuit ID Frame Relay: Data-link connection identifier (DLCI) Serial interfaces: High-Level Data Link Control (HDLC)
State	Displays whether the neighbor is up or down
Holdtime	Time required to receive another IIH to maintain the IS-IS adjacency
Type	Type of adjacency formed with a neighbor: L1, L2, or L1-L2

**Table 8-8** IS-IS Neighbor State Fields

Note

Notice that the system ID actually references the router's hostname rather than the 6-byte system ID. IS-IS provides a name to system ID mapping under the optional TLV 137 that is found as part of the LSP, as shown in [Figure 8-17](#).

This feature is enabled by default but can be disabled under the IS-IS router configuration on IOS nodes with the command **no hostname dynamic** or on IOS XR nodes with the command **hostname dynamic disable**.

### Verification of IS-IS Routes

The next step is to verify the IS-IS routes installed in the IP routing table. IS-IS routes that install into the RIB are shown with the command **show ip route isis** on IOS nodes and **show route isis** on IOS XR nodes. Routes that are learned via an L1 adjacency will display as *i L1*, and routes learned via an L2 adjacency will display as *i L2*.

[Example 8-5](#) provides sample output of the IS-IS routing table for XR1 and R2. In the output, two sets of numbers are in the brackets and look like [115/20]. The first number 115 refers to the default AD for IS-IS, and the second number is the metric of the path used for that network. Output for R3 will be similar.

### Example 8-5 IS-IS Routes Installed in the RIB

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route isis
```

```
! Output omitted for brevity
```

```
i L1 10.2.2.0/24 [115/20] via 10.123.1.2, 00:08:36, GigabitEthernet0/0/0/2
i L1 10.3.3.0/24 [115/20] via 10.123.1.3, 00:10:39, GigabitEthernet0/0/0/2
i L1 192.168.2.2/32 [115/20] via 10.123.1.2, 00:08:36, GigabitEthernet0/0/0/2
i L1 192.168.3.3/32 [115/20] via 10.123.1.3, 00:10:48, GigabitEthernet0/0/0/2
```

```
R2#show ip route isis
! Output omitted for brevity
```

```
10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
i L1    10.1.1.0/24 [115/10] via 10.123.1.1, 00:09:13, GigabitEthernet0/0
i L1    10.3.3.0/24 [115/20] via 10.123.1.3, 00:11:10, GigabitEthernet0/0
i L1    192.168.1.1 [115/10] via 10.123.1.1, 00:09:13, GigabitEthernet0/0
i L1    192.168.3.3 [115/20] via 10.123.1.3, 00:11:14, GigabitEthernet0/0
```

## DESIGNATED INTERMEDIATE SYSTEM

Broadcast networks allow more than two routers to exist on a network segment. This could cause scalability problems with IS-IS, as the number of routers on a segment increase. Additional routers will flood more LSPs on the segment, and ensuring that the databases are synchronized can become resource-intensive.

IS-IS overcomes this inefficiency by creating a *pseudonode* (virtual router) to manage the synchronization issues that arise on the broadcast network segment. A router on the broadcast segment, known as the *designated intermediate system* (DIS), assumes the role of the pseudonode. If the acting DIS router fails, another router becomes the new DIS and assumes the responsibilities. A pseudonode and DIS exist for each IS-IS level (L1 and L2), which means that a broadcast segment can have two pseudonodes and two DISs.

By inserting the logical pseudonode into a broadcast segment, the multi-access network segment is converted into multiple P2P networks in the LSPDB, as shown in Figure 8-26. This concept will become more coherent when the LSPDB is explained in detail.

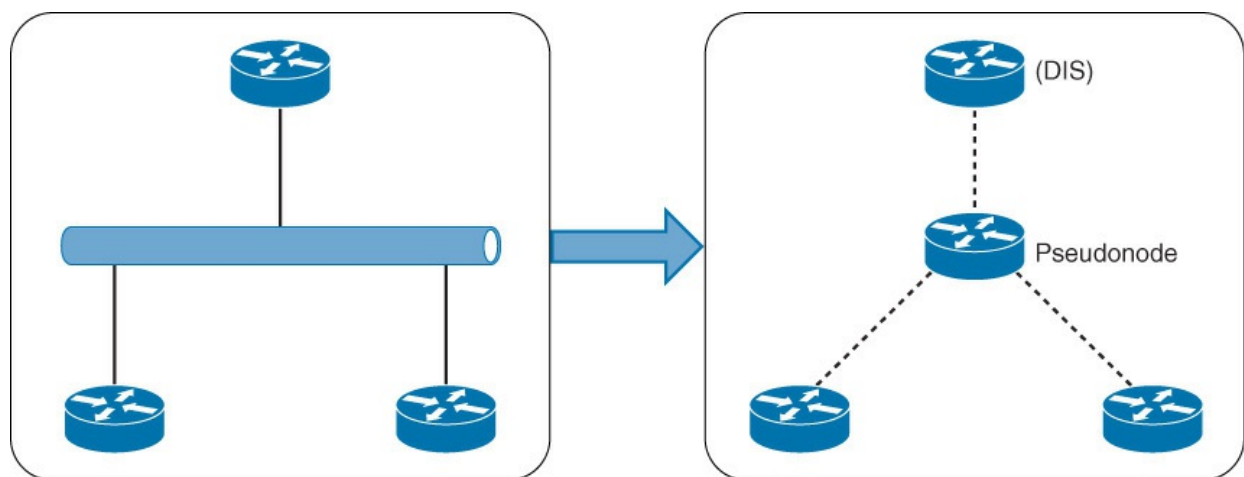
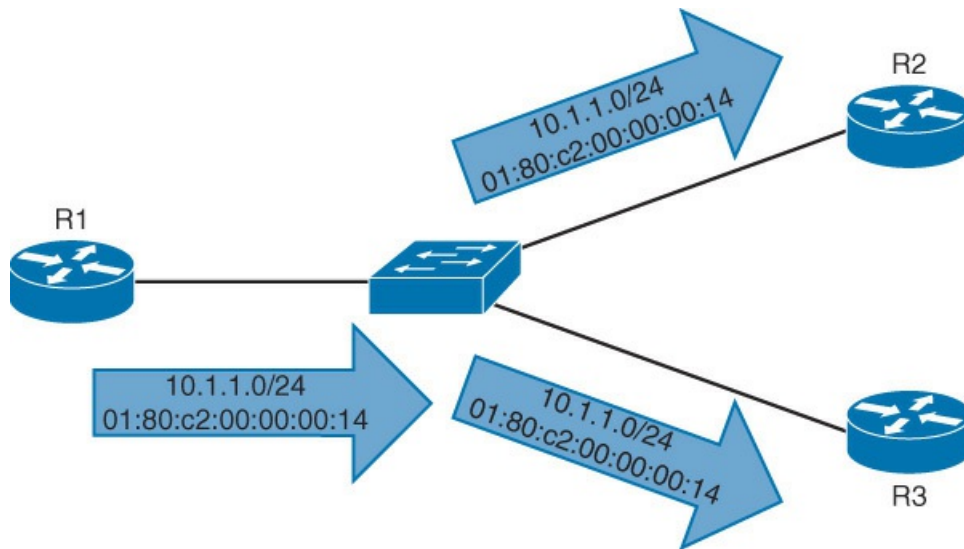


Figure 8-26 DIS Logical Drawing

There is a natural tendency to associate IS-IS DIS behavior with OSPF's designated router (DR) behavior, but they operate in a different way. All IS-IS routers form a full neighbor adjacency with each other, and any router can advertise non-pseudonode LSPs to all other IS-IS routers on that segment; whereas OSPF specifies that LSAs are sent to the DR to be advertised to the network segment.

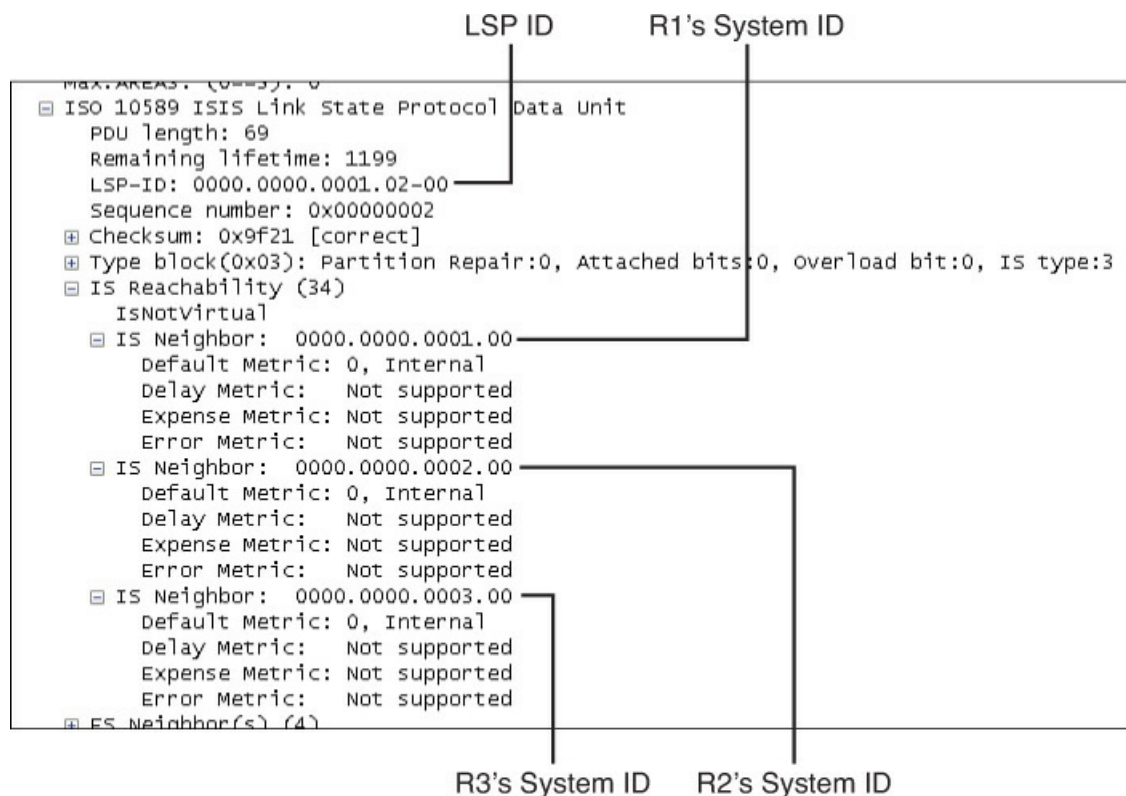
In Figure 8-27, R1 has just added the 10.1.1.0/24 network and is advertising the L1 LSP to the All L1 Router MAC address (01:80:c2:00:00:14). R2 and R3 receive the LSP and process the LSP accordingly into the LSPDB.



**Figure 8-27** LSP Advertisement on Broadcast Networks

The DIS advertises a pseudonode LSP that indicates the routers that attach to the pseudonode. The pseudonode LSP acts like an OSPF Type 2 LSA because it indicates the attached neighbors and informs the nodes which router is acting as the DIS.

Figure 8-28 shows a packet capture of a pseudonode LSP. Based on the LSP ID, the node 0000.0000.0001.00 is the acting DIS for this segment. The system IDs of the routers connected to the pseudonode are listed in the IS Reachability TLV, with an interface metric set to 0 because SPF will use the metric for the non-pseudonode LSPs for calculating the SPF tree.



**Figure 8-28** Pseudonode LSP Packet Capture

The pseudonode advertises the complete sequence number packets (CSNPs) every 10 seconds. IS-IS routers check their LSPDBs to verify that all LSPs listed in the CSNP exist and that the



sequence number matches the version in the CSNP:

- If an LSP is missing or the router has an outdated (lower sequence number) LSP, the router advertises a partial sequence number packet (PSNP) requesting the correct or missing LSP. All IS-IS routers will receive the PSNP, but only the DIS sends out the correct LSP thereby reducing traffic on that network segment.
- If a router detects that the sequence number in the CSNP is lower than the LSP that what is in its LSPDB, it advertises the LSP. All IS-IS routers will receive the LSP and process it accordingly. The DIS should send out an updated CSNP with the updated sequence number for the advertised LSP.

Figure 8-29 illustrates the pseudonode advertising the CSNP and pseudonode LSP.

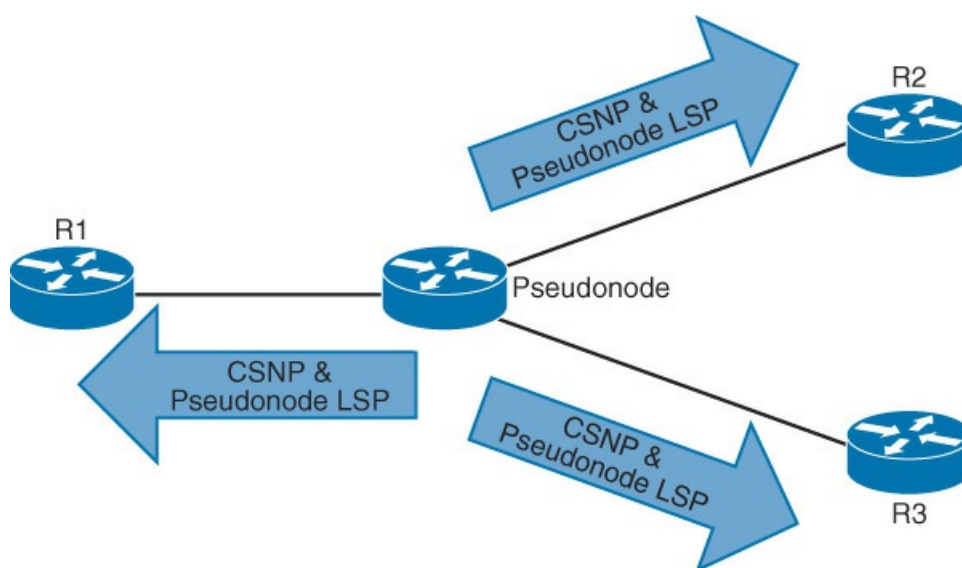


Figure 8-29 Pseudonode LSP and CSNP Advertisement

### DIS Elections

A broadcast network segment must have formed at least one *up* adjacency with an ES or IS and have waited two times the IS-IS hello timer (~20 seconds) before starting the DIS election process for a segment. Any router with the interface priority of 1–127 will attempt to become the DIS. Cisco sets the default priority to 64, and interfaces with a priority of 0 do not participate in the DIS election.

Routers will then receive and examine the IS-IS hellos from neighboring routers. If a router identifies itself as more favorable router than the IS-IS hellos it receives, it continues to send out hellos with its system ID in the DIS system ID portion of the hello PDU. If the hello received is more favorable, the router will update its hello packet to use the more preferable system ID in the DIS system ID portion of the hello PDU. Once the DIS is elected, it will advertise the pseudonode LSP and CSNP packets for that network segment.

IS-IS deems a router more preferable if the priority for the interface is the highest for that segment. If the priority is the same, the higher MAC address is more preferable. IS-IS supports preemption; so when a more preferred router is detected, the DIS ownership changes immediately. The new DIS advertises a new pseudonode LSP, while the other routers purge the old pseudonode LSP.

Note

If the DIS roles changes between nodes, IS-IS recalculates the SPF tree for the level because the pseudonode ID will change in all of the attached non-pseudonode LSPs in addition to the new pseudonode LSP being advertised.

IS-IS does not need a backup DIS because the primary purpose is to ensure that the LSPDB is synchronized and that a temporary outage does not really impact the advertisement of LSPs in the manner that it would for a failure of an OSPF's DR. The DIS sends IS-IS hello packets every 3 seconds to detect DIS failures quickly.

### DIS Placement

Figure 8-30 represents the topology example to explain a basic IS-IS configuration. R3 won the DIS election because all the IS-IS routers had the same priority (64), and the next decision point used the highest MAC address for that segment, which is R3's MAC address (aa:bb:cc:00:66:00).

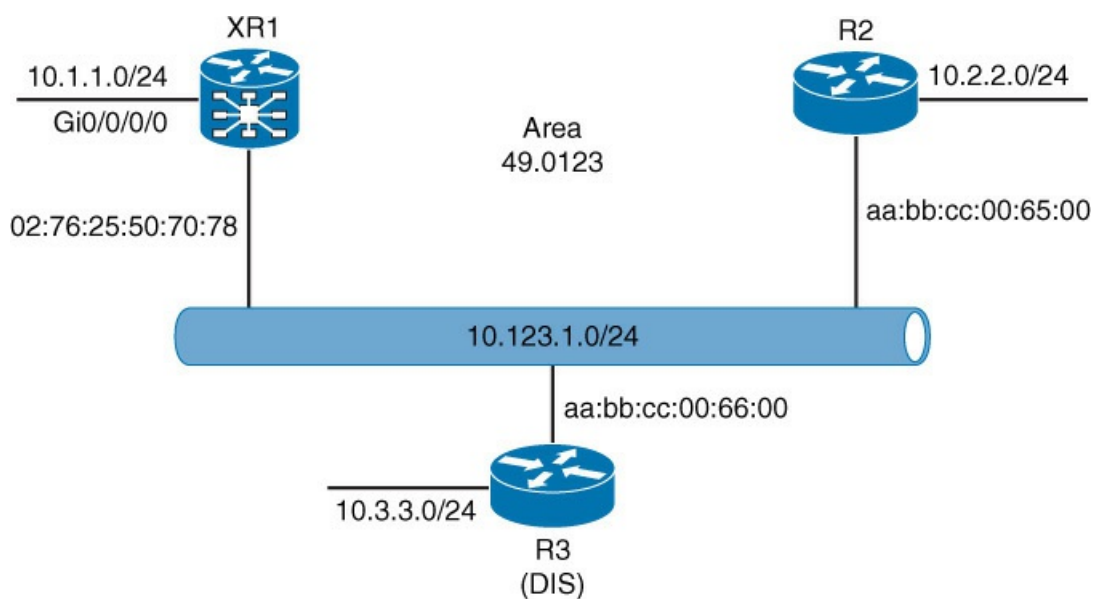


Figure 8-30 IS-IS Topology Example with MAC Addresses

Modifying a router's IS-IS interface priority to a higher value than the existing DR allows for the DIS to be moved to a different node

The priority can be set manually with the interface parameter command **ip isis priority 0-127** for IOS nodes, or with the command **priority 0-127** under the specified interface on IOS XR nodes. Setting an interface priority to 0 removes that interface from the DIS election immediately. Raising the priority above the default value (64) makes that interface more favorable over interfaces with the default value.

Example 8-6 provides the relevant configuration for XR1 to ensure that it is the DIS for the 10.123.1.0/24 segment and that R2 should not enter the DIS election process.

### Example 8-6 IS-IS Configuration with DIS Manipulation

[Click here to view code image](#)

### **XR1**

```
router isis CISCO
 net 49.0123.0000.0000.0001.00
interface GigabitEthernet0/0/0/0
 address-family ipv4 unicast
 !
 !
interface GigabitEthernet0/0/0/2
 priority 80
 address-family ipv4 unicast
 !
```

### **R2**

```
interface GigabitEthernet0/0
 ip address 10.123.1.2 255.255.255.0
 ip router isis
 isis priority 0
 !
interface GigabitEthernet0/1
 ip address 10.2.2.2 255.255.255.0
 ip router isis

router isis
 net 49.0123.0000.0000.0002.00
```

The easiest way to determine the interface role is by viewing the IS-IS interfaces and looking under *LAN ID* for IOS XR nodes, and *DR ID* for IOS nodes. [Example 8-7](#) provides verification that the DIS has successfully moved and that R2 cannot become the DIS.

### **Example 8-7** *Verification of DIS and Interface Priority*

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis interface gi0/0/0/2 | include "LAN ID|Priority"
LAN ID: XR1.03
Priority (Local/DIS): 80/80
LAN ID: XR1.03
Priority (Local/DIS): 80/80
```

```
R2#show clns interface GigabitEthernet0/0 | include DR|Priority
Level-1 Metric: 10, Priority: 0, Circuit ID: XR1.03
DR ID: XR1.03
Level-2 Metric: 10, Priority: 0, Circuit ID: XR1.03
DR ID: XR1.03
```

## **POINT-TO-POINT ADJACENCY ON BROADCAST MEDIA**

Ethernet interfaces that are directly connected or broadcast networks with only two IS-IS routers do not benefit from the use of a pseudonode. Resources are not wasted on electing a DIS, CSNPs are not continuously flooded into a segment, and an unnecessary pseudonode LSP is not included in the LSPDB of all routers in that level. IS-IS allows broadcast interfaces to behave like a P2P interface.

Configuring a broadcast interface as P2P uses the interface parameter command **isis network point-to-point** on IOS nodes. IOS XR nodes use the command **point-to-point** under the interface within the IS-IS router configuration.

Figure 8-31 provides a topology example where XR1 and R2 are connected via a direct Ethernet cable. XR1 and R2 use a P2P network type to reduce router resource consumption.

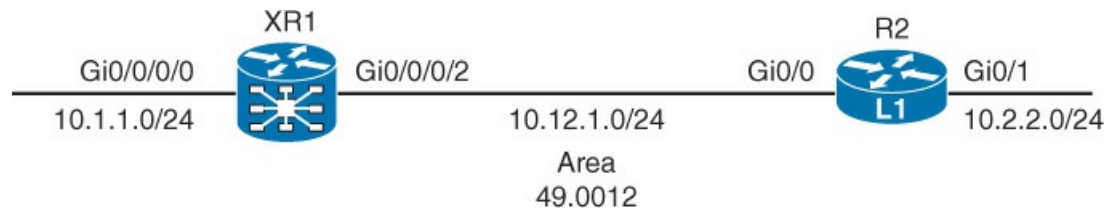


Figure 8-31 IS-IS Topology for Broadcast Circuit **D** P2P Circuit

Note

This topology will be used as a reference for other features later in the chapter.

Example 8-8 provides the relevant configuration for XR1 and R2.

### Example 8-8 IS-IS Configuration for Broadcast Circuit --> P2P

[Click here to view code image](#)

```
XR1
router isis CISCO
 net 49.0012.0000.0000.0001.00
 interface GigabitEthernet0/0/0/0
  address-family ipv4 unicast
  !
  !
 interface GigabitEthernet0/0/0/2
  point-to-point
  address-family ipv4 unicast
  !
  !
  !
```

```
R2
interface GigabitEthernet0/0
 ip address 10.12.1.2 255.255.255.0
 ip router isis
 isis network point-to-point
 !
 interface GigabitEthernet0/1
 ip address 10.2.2.2 255.255.255.0
 ip router isis
 router isis
 net 49.0012.0000.0000.0002.00
```

Example 8-9 verifies that a pseudonode does not connect XR1 to R2 and that there are not any

pseudonode LSPs because there are not any pseudonodes in the topology.

### Example 8-9 XR1's LSPDB

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis database level 1 detail | exclude IP
! Output omitted for brevity

IS-IS ISIS (Level-1) Link State Database
LSPID          LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
XR1.00-00      * 0x00000009  0x4133        693           0/0/0
  Metric: 10    IS R2.00
R2.00-00       0x00000006  0xd6f2        698           0/0/0
  Metric: 10    IS XR1.00

Total Level-1 LSP count: 2    Local Level-1 LSP count: 1
```

```
RP/0/0/CPU0:XR1#show isis database level 1

IS-IS ISIS (Level-1) Link State Database
LSPID          LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
XR1.00-00      * 0x00000009  0x4133        746           0/0/0
R2.00-00       0x00000006  0xd6f2        751           0/0/0

Total Level-1 LSP count: 2    Local Level-1 LSP count: 1
```

## LINK STATE PACKET DATABASE

The link-state packet database (LSPDB) is a critical component of the IS-IS routing protocol. The LSPDB contains all the LSPs, for a specific level, and L1-L2 routers will have two LSPDBs. All routers within the same level maintain an identical LSPDB.

Figure 8-32 provides a sample L1 topology to explain LSPs and the LSPDB in detail throughout this section. XR1 and XR2 connect via serial link; R3 and R4 connect via serial link; and XR2, R3, and R5 all connect via a broadcast (LAN) segment. Routers have formed only an L1 adjacency to simplify this section, but the same concept applies to L2 LSPs.

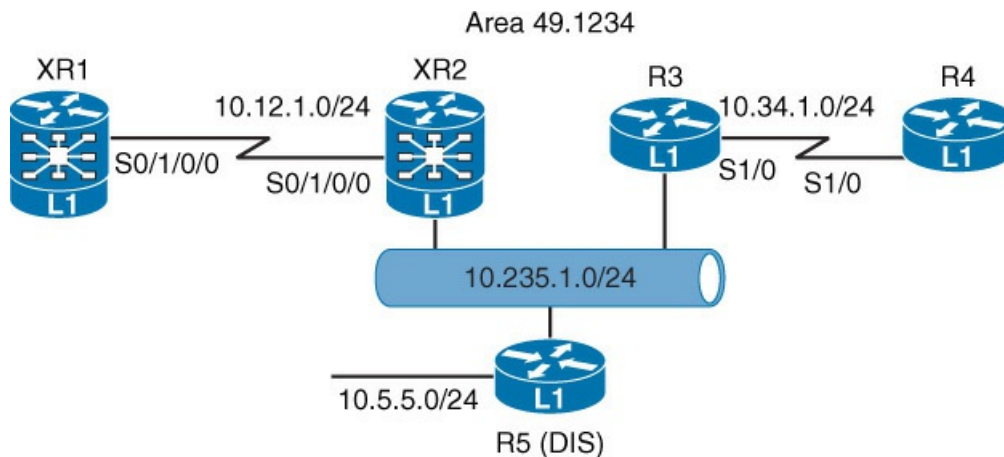


Figure 8-32 IS-IS Topology for Explaining LSPs

## Viewing the LSPDB

All the LSPs within a router's LSPDB are shown with the command **show isis database [level-1 | level-2]** on IOS nodes, and IOS XR nodes use the command **show isis database [level {1|2}]**. The optional **level** parameter provides a mechanism to reduce the query to a specific IS-IS level.

**Example 8-10** lists all the L1 LSPs from **Figure 8-32**. There are six LSPs and five routers in the topology. This is because five of the LSPs are non-pseudonode LSPs, and one of the LSPs (R5.02-00) is a pseudonode LSP for the 10.235.1.0/24 broadcast network. Notice that the output is identical for both of the routers because they have identical LSPDBs.

### Example 8-10 LSPs in IS-IS Database

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis database
```

```
IS-IS ISIS (Level-1) Link State Database
```

LSPID	LSP Seq Num	LSP Checksum	LSP Holdtime	ATT/P/OL
XR1.00-00	* 0x00000004	0x01e5	1061	0/0/0
XR2.00-00	0x00000006	0x6d46	1059	0/0/0
R3.00-00	0x00000004	0x6986	991	0/0/0
R4.00-00	0x00000002	0x3ade	989	0/0/0
R5.00-00	0x00000003	0x172d	993	0/0/0
R5.02-00	0x00000002	0xb701	1053	0/0/0

```
Total Level-1 LSP count: 6      Local Level-1 LSP count: 1
```

```
R4#show isis database
```

```
IS-IS Level-1 Link State Database:
```

LSPID	LSP Seq Num	LSP Checksum	LSP Holdtime	ATT/P/OL
XR1.00-00	0x00000004	0x01E5	1031	0/0/0
XR2.00-00	0x00000006	0x6D46	1025	0/0/0
R3.00-00	0x00000004	0x6986	952	0/0/0
R4.00-00	* 0x00000002	0x3ADE	952	0/0/0
R5.00-00	0x00000003	0x172D	950	0/0/0
R5.02-00	0x00000002	0xB701	1006	0/0/0

**Table 8-9** explains the fields shown in **Example 8-10**.

Field	Description
*	Indicates LSPs that are advertised from this router (that is, self-originated)
LSPID	The LSP ID that consists of the system ID, pseudonode ID, and fragmentation ID
LSP Sequence Number	The 32-bit sequence number used for version control for that LSP
LSP Holdtime	The remaining lifetime that the LSP remains valid on this router
ATT	Indicates whether the attached bit is set on the LSP
P	Indicates whether the partition repair bit is set on the LSP
OL	Indicates whether the overload bit is set on the LSP

**Table 8-9** IS-IS Neighbor State Fields

### Non-Pseudonode LSPs

All IS-IS routers advertise a non-pseudonode LSP. Within the TLVs, they contain a list of neighboring IS-IS routers for building the topology, and they can IP addresses that will overlay on top of the topology.

LSP information can be seen by querying the LSPDB with the command **show isis database** [*lsp-id*] [**detail**] on IOS nodes, and IOS XR nodes use the command **show isis database** [*lsp-id*] [**detail**]. Adding a specific LSP ID to the command restricts the view to a specific LSP.

**Example 8-11** shows the non-pseudonode LSP for XR1 and XR2. IS-IS uses the highlighted ‘IS Neighbors’ TLV2 to link the nodes together when building the IS-IS topology. P2P networks will reference the non-pseudonode-LSP; whereas broadcast interfaces refer to the pseudonode LSP for that segment because the pseudonode converts the physical broadcast network into multiple logical P2P connections. XR2 connects the non-pseudonode LSP XR1.00 and to the pseudonode LSP R5.02. Notice that both IS Neighbors and IP Reachability TLVs include a metric.

### Example 8-11 IS-IS Routes Installed in the RIB

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis database XR1.00-00 detail

LSPID                LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
XR1.00-00            0x00000004   0x01e5        1061          0/0/0
  Area Address: 49.1234
  NLPID:        0xcc
  Hostname:     XR1
  IP Address:   192.168.1.1
  Metric: 10    IS XR2.00
  Metric: 10    IP 10.12.1.0/24
  Metric: 10    IP 192.168.1.1/32
```

```
RP/0/0/CPU0:XR2#show isis database XR2.00-00 detail
```

```
LSPID                LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
XR2.00-00            0x00000006   0x6d46        937            0/0/0
  Area Address: 49.1234
  NLPID:         0xcc
  Hostname:      XR2
  IP Address:    192.168.2.2
  Metric: 10     IS XR1.00
  Metric: 10     IS R5.02
  Metric: 10     IP 10.12.1.0/24
  Metric: 10     IP 10.235.1.0/24
  Metric: 10     IP 192.168.2.2/32
```

## Pseudonode LSPs

Pseudonode LSPs are advertised only by the DIS for broadcast networks. Pseudonode LSPs contain only the IS Neighbors TLV, which links multiple routers together via the pseudonode. The metric for pseudonode LSPs will always be 0 because IS-IS uses the metric associated to the non-pseudonode LSP when calculating the shortest path.

### Note

Remember that pseudonode LSPs are LSPs with a pseudonode ID that is a non-0 number.

**Example 8-12** shows the pseudonode LSP that is being advertised by R5. Notice that the pseudonode ID is not 00 and is 02. The metric is always 0 in pseudonode LSPs.

## Example 8-12 IS-IS Routes Installed in the RIB

[Click here to view code image](#)

```
R5#show isis database R5.02-00 detail
```

```
IS-IS Level-1 LSP R5.02-00
LSPID                LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
R5.02-00            * 0x00000002  0xB701        499            0/0/0
  Metric: 0         IS R5.00
  Metric: 0         IS XR2.00
  Metric: 0         IS R3.00
```

## Building the Topology

An IS-IS router builds a level's topology by linking LSPs together, specifically by using the IS Neighbors TLV (2), which uses a combination of the system ID and the pseudonode ID for identifying the neighboring routers. On P2P networks, the IS Neighbor will refer to the system ID followed by .00. Broadcast networks will refer to the system ID and pseudonode ID associated to the DIS on that segment.

**Example 8-13** provides a parsed list of all the LSPs with the relevant information for building the IS-IS topology. Notice that R5 has a non-pseudonode LSP and a pseudonode LSP.

## Example 8-13 IS-IS Routes Installed in the RIB

[Click here to view code image](#)



```
RP/0/0/CPU0:XR1#show isis database level 1 detail | exclude "NLPID:|IP"
```

```
IS-IS ISIS (Level-1) Link State Database
```

LSPID	LSP Seq Num	LSP Checksum	LSP Holdtime	ATT/P/OL
XR1.00-00	* 0x00000005	0xfee6	1167	0/0/0
Area Address: 49.1234				
Hostname: XR1				
Metric: 10	IS XR2.00			
XR2.00-00	0x00000007	0x6b47	978	0/0/0
Area Address: 49.1234				
Hostname: XR2				
Metric: 10	IS XR1.00			
Metric: 10	IS R5.02			
R3.00-00	0x00000005	0x6787	902	0/0/0
Area Address: 49.1234				
Hostname: R3				
Metric: 10	IS R4.00			
Metric: 10	IS R5.02			
R4.00-00	0x00000003	0x38df	898	0/0/0
Area Address: 49.1234				
Hostname: R4				
Metric: 10	IS R3.00			
R5.00-00	0x00000004	0x152e	965	0/0/0
Area Address: 49.1234				
Hostname: R5				
Metric: 10	IS R5.02			
R5.02-00	0x00000003	0xb502	1045	0/0/0
Metric: 0	IS R5.00			
Metric: 0	IS XR2.00			
Metric: 0	IS R3.00			

```
Total Level-1 LSP count: 6      Local Level-1 LSP count: 1
```

Figure 8-33 shows the output from Example 8-13 linked together. Notice how XR2's, R3's, and R5's non-pseudonode LSP connects to R5's pseudonode LSP. The LSP topology looks identical to the physical topology. Conceptually, once the topology is built, the IP reachability information is placed on top of the built IS-IS topology.

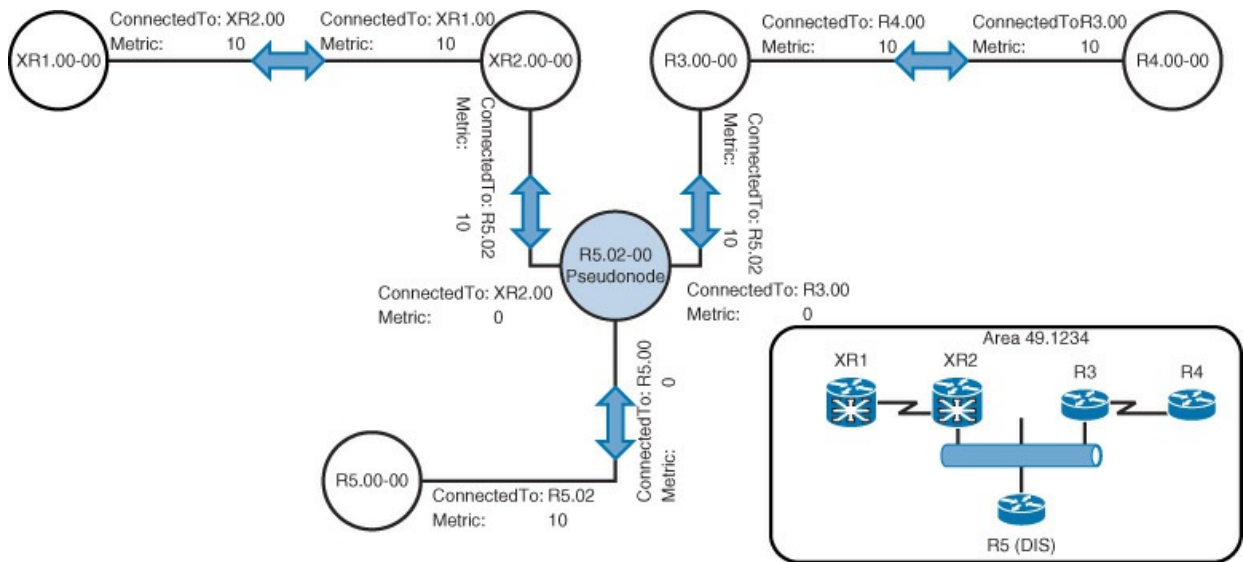


Figure 8-33 LSP Topology

### Viewing the Topology

The IS-IS topology displays a list of routers in the LSPDB for an IS-IS level. It includes the total path metric to the advertising router, next-hop router, outbound interface, and SNPA address from the perspective of the local router. The command **show isis topology** [hostname | system-ID] [level-1 | level-2] provides the topology for IOS nodes, and IOS XR nodes use the command syntax **show isis topology [systemid system-ID] [level {1|2}]**. The optional parameters allow for reducing the size of the query to a specific level or router.

Example 8-14 displays XR1's and R5's perspective of the IS-IS topology for Figure 8-32. The metric reflects the path metric to the advertising node, which is calculated by the SPT on the local router. Notice on XR1 that the path metric to XR2 is 10, the path metric to R3 and R5 is 20, and the path metric to R4 is 30.

### Example 8-14 IS-IS Topology

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis topology

IS-IS ISIS paths to IPv4 Unicast (Level-1) routers
System Id      Metric  Next-Hop      Interface      SNPA
XR1            --
XR2            10     XR2           Se0/1/0/0     *PtoP*
R3             20     XR2           Se0/1/0/0     *PtoP*
R4             30     XR2           Se0/1/0/0     *PtoP*
R5             20     XR2           Se0/1/0/0     *PtoP*
```

```
R5#show isis topology
```

```
IS-IS TID 0 paths to level-1 routers
```

System Id	Metric	Next-Hop	Interface	SNPA
XR1	20	XR2	Gi0/0	0213.9a10.68d1
XR2	10	XR2	Gi0/0	0213.9a10.68d1
R3	10	R3	Gi0/0	aabb.cc00.6600
R4	20	R3	Gi0/0	aabb.cc00.6600
R5	--	--	--	--

### SPF Calculations

A key concept with IS-IS that differs significantly from OSPFv2 is how each protocol performs the SPF calculation. OSPF builds the topology on IPv4 links, so a network prefix is interpreted as a node. Even prefix changes on an isolated interface are considered a topology change, resulting in a complete SPF calculation for all routers in that area.

IS-IS builds the topology based on links listed in the IS Neighbors TLV 2. The IP prefixes reside on top of the IS-IS topology. Prefix changes on an interface does not change the topology, so a complete SPF calculation does not occur for all routers in the area.

Figure 8-34 demonstrates the subtle differences in OSPF's SPF tree when compared to IS-IS. Notice that the 10.1.1.0/24, 10.2.2.0/24, 10.3.3.0/24, and 10.4.4.0/24 networks appear as nodes (circles) in OSPF's SPF tree, whereas in IS-IS they sit on top of the links for IS-IS.

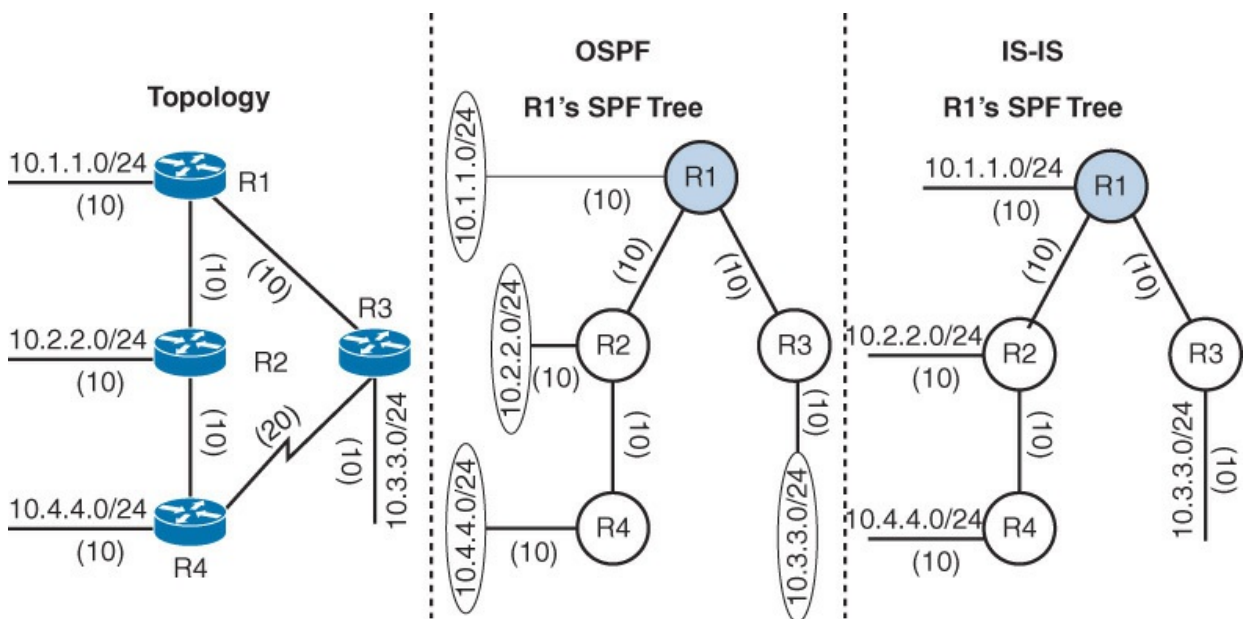


Figure 8-34 OSPF Versus IS-IS Shortest Path Tree

IS-IS saves router resources by executing a partial route computation (PRC) for changes that do not affect the topology (links between IS-IS nodes). The following list outlines the events that will invoke the PRC of a full SPF calculation:

- Adding/removing IP prefixes
- Configuring isolated interfaces as passive
- Changing the metric on isolated interfaces

- Enabling or disabling the passive state for an isolated interface
- Interarea and external route changes

All other changes invoke a full SPF calculation within a level.

## PASSIVE INTERFACES

Enabling an interface with IS-IS is the quickest way to advertise the network segment to other IS-IS routers. However, it might be easy for someone to plug in an unauthorized IS-IS router on a network segment, and introduce false routes, causing havoc in the network. Making the network interface passive still adds the network segment into the LSPDB, but prohibits the interface from forming IS-IS adjacencies. A passive interface does not send out IS-IS traffic and will not process any received IS-IS packets.

On IOS nodes, the command **passive interface** *interface-type interface-number* will make the interface passive, and the command **passive interface default** will make all interfaces passive. To allow for an interface to process IS-IS packets, the command **no passive interface** *interface-type interface-number* is used. On IOS XR nodes, the command **passive** under the interface within the IS-IS router configuration will make the interface passive.

### Note

On IOS nodes, when an interface is made passive, the interface parameter command **ip router isis** [*instance-id*] is removed. The interface's networks will still be advertised, but if the **passive** statement is removed, the **ip router isis** command will need to be reapplied for the network to be advertised.

Refer back to [Figure 8-31](#) for a topology example with an Ethernet link connecting XR1 to R2. XR1 and R2 will set all interfaces as passive except for the 10.12.1.0/24 interfaces. [Example 8-15](#) provides the relevant configuration. Notice that R2 does not have the **ip router isis** command on the Loopback 0 and Gigabit Ethernet 0/1 interface.

### Example 8-15 IS-IS Metric Configuration

[Click here to view code image](#)

```
XR1
router isis CISCO
 net 49.0012.0000.0000.0001.00
 interface Loopback0
  passive
  address-family ipv4 unicast
  !
  !
 interface GigabitEthernet0/0/0/0
  address-family ipv4 unicast
  passive
  !
 interface GigabitEthernet0/0/0/2
  address-family ipv4 unicast
```

```

R2
interface Loopback0
 ip address 192.168.2.2 255.255.255.255
!
interface GigabitEthernet0/0
 ip address 10.12.1.2 255.255.255.0
 ip router isis

!
interface GigabitEthernet0/1
 ip address 10.2.2.2 255.255.255.0
!
router isis
 net 49.0012.0000.0000.0002.00
 passive-interface default
 no passive-interface GigabitEthernet0/0

```

**Example 8-16** provides verification that XR1 and R2 can form an adjacency and that the remote networks are advertised to each other.

### **Example 8-16** XR1's and R2's Routing Table

**Click here to view code image**

```

RP/0/0/CPU0:XR1#show route isis

i L1 10.2.2.0/24 [115/10] via 10.12.1.2, 00:19:11, GigabitEthernet0/0/0/2
i L1 192.168.2.2/32 [115/10] via 10.12.1.2, 00:19:06, GigabitEthernet0/0/0/2

```

```

R2#show ip route isis
! Output omitted for brevity

10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
i L1 10.1.1.0/24 [115/10] via 10.12.1.1, 00:08:33, GigabitEthernet0Ethernet0/0
192.168.1.0/32 is subnetted, 1 subnets
i L1 192.168.1.1 [115/10] via 10.12.1.1, 00:23:29, GigabitEthernet0Ethernet0/0

```

**Example 8-17** verifies that interfaces are in a passive state.

### **Example 8-17** Verification of Passive Interfaces

**Click here to view code image**

```

RP/0/0/CPU0:XR1#show isis protocol
! Output omitted for brevity

IS-IS Router: ISIS
Interfaces supported by IS-IS:
 Loopback0 is running passively (passive in configuration)
 GigabitEthernet0/0/0/0 is running passively (passive in configuration)
 GigabitEthernet0/0/0/2 is running actively (active in configuration)

```

```
R2#show clns protocol
! Output omitted for brevity

IS-IS Router: <Null Tag>
  Interfaces supported by IS-IS:
    GigabitEthernet0/0 - IP
  Passive interface:
    GigabitEthernet0Ethernet0/1
    Loopback0
```

## REMOVAL OF HELLO PADDING

IS-IS hellos (IIHs) are padded with TLV 8 to reach the maximum transmission unit (MTU) size of the network interface. Padding IIHs provides the benefit of detecting errors with large frames or mismatched MTU on remote interfaces. Broadcast interfaces transmit L1 and L2 IIHs wasting bandwidth if both interfaces use the same MTU.

Cisco introduced a feature that disables the MTU padding after the router sends the first five IIHs out of an interface. This eliminates wasted bandwidth while still providing a mechanism for checking the MTU between routers. IOS nodes can disable the padding of IIH packets with the interface parameter command **no isis hello padding**. IOS nodes can also set all interfaces to the same setting with the command **no hello padding [multi-point | point-to-point]** under the IS-IS router configuration. The optional **multi-point** or **point-to-point** keywords isolate the setting to broadcast and P2P interfaces accordingly.

IOS XR provide similar functionality with the command **hello-padding {disable | sometimes} [level {1 | 2}]** under the interface within the IS-IS router configuration. The **disable** keyword will not pad any IIHs while the **sometimes** keyword will pad only the first five IIHs during adjacency formation. IOS XR allows the setting to be set for a specific ISIS level.

Refer back to [Figure 8-31](#) for a reference topology with an Ethernet link connecting XR1 to R2. [Example 8-18](#) provides the relevant configuration for removal of the IIH padding after the first five IIHs are sent for forming an IS-IS adjacency.

### Example 8-18 IIH Padding Removal Configuration

[Click here to view code image](#)

```
XR1
router isis CISCO
  net 49.0012.0000.0000.0001.00
  !
  interface GigabitEthernet0/0/0/0
    address-family ipv4 unicast
    !
  !
  interface GigabitEthernet0/0/0/2
    hello-padding sometimes
    address-family ipv4 unicast
```

## R2

```
interface GigabitEthernet0/0
ip address 10.12.1.2 255.255.255.0
ip router isis
no isis hello padding
!
interface GigabitEthernet0/1
ip address 10.2.2.2 255.255.255.0
ip router isis
router isis
net 49.0012.0000.0000.0002.00
```

## FAILURE DETECTION

A secondary function to the IS-IS hello packets is to ensure that adjacent IS-IS neighbors are still healthy and available. IS-IS sends hello packets at set intervals called the *hello timer*. IS-IS uses a hello multiplier to calculate the holding timer. The default hello multiplier is 3, so the default holding timer is three times the hello timer. Broadcast interfaces can configure the hello timers and multipliers for the L1 IIH PDUs that are independent from the hello timer and multipliers for the L2 IIH PDUS.

Upon receipt of the hello packet from a neighboring router, the holding timer resets to the initial value and starts to decrement again. If a router does not receive a hello before the holding timer reaches 0, the neighbor state is changed to down. The IS-IS router immediately sends out the appropriate LSP reflecting the topology change, and the SPF algorithm processes on all routers within the area.

### Hello Timer

The default IS-IS hello timer interval value is 10 seconds for all interfaces and 3 seconds for the DIS. IS-IS allows modification to the hello timer interval with values between 1–65,535 seconds. Changing the hello timer interval modifies the holding timer, too. IOS uses the command **isis hello-interval** [*1-65535* | **minimal**] [**level-1** | **level-2**] under the interface to modify the IS-IS hello timer, and the equivalent command **hello-interval** *1-65535* [**level** {**1** | **2**}] is used on IOS XR nodes under the interface within the IS-IS router configuration.

The **minimal** keyword configures IS-IS subsecond hellos providing failure detection to 1 second. The hello PDU interval is calculated by the hello multiplier. A higher hello multiplier will send more hello packets with a shorter interval.

#### Note

IOS XR does not support IS-IS subsecond hellos. IOS and IOS XR support other technologies that are less likely to provide false positives. Technologies like BFD provide fast convergence and are more preferred.

Displaying an IS-IS interface will display when the next IIH PDU should be transmitted. [Example 8-19](#) shows sample output for the IIH timers.

### Example 8-19 IS-IS Interface IIH Frequency

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis interface GigabitEthernet0/0/0/0
```

```
! Output omitted for brevity
```

```
Level-1
  Next LAN IIH in:      6 s
Level-2
  Next LAN IIH in:      5 s
```

```
R2#show clns interface GigabitEthernet0/0
```

```
! Output omitted for brevity
```

```
Next IS-IS LAN Level-1 Hello in 2 seconds
Next IS-IS LAN Level-2 Hello in 8 seconds
```

## Hello Multiplier and Holding Timer

The hello multiplier can be set between a value of 3–1000. The hello multiplier can be change with the **isis hello-multiplier 3-1000 [level-1 | level-2]** interface parameter command on IOS nodes. IOS XR nodes use the equivalent command **hello-multiplier 3-1000 [level {1|2}]** under the interface within the IS-IS router configuration.

### Note

IS-IS does not require the hello timers to match to form an adjacency; however, a hello packet must be received within the holding timer to ensure the adjacency is maintained.

## AUTHENTICATION

Authentication is a mechanism for ensuring that only authorized routers participate within the IS-IS routing domain. IS-IS allows for the authentication of hello packets and LSPs. Hello packets are required to form an adjacency and are configured on an interface-by-interface perspective. LSP packets are required to exchange routes, and authentication is configured for the appropriate level within the router process. IS-IS authentication can use different settings for each IS-IS level. Authenticating on one PDU type is sufficient for most designs.

IS-IS provides two types of authentication:

- **Plaintext:** Plaintext mode provides little security, as anyone with access to the link can see the password with a network sniffer.
- **An MD5 cryptographic hash:** MD5 cryptographic hash uses a hash instead, so the password is never included in the PDUs, and this technique is widely accepted as being the more secure mode.

All IS-IS authentication is stored in TLV 10, which is included in IIHs and PDUs.

### IS-IS Hello Authentication

IOS nodes enable hello authentication with the interface parameter command **isis authentication key-chain *key-chain-name* [{level-1 | level-2}]**. The authentication type is identified with the command **isis key-chain authentication mode {md5 | text} [{level-1 | level-2}]**. If a level is not specified the authentication applies to L1 and L2 hello PDUs. The legacy interface parameter command **isis password *password*** enables plaintext IS-IS hello authentication, too.

IOS XR nodes enable hello authentication with the command **hello-password {text *password* | hmac-md5 *password* | keychain *key-chain-name*} [level {1 | 2}]** under the interface within the IS-IS router configuration.



Note

IOS XR nodes will need to specify the MD5 encryption with the command **cryptographic-algorithm MD5** in the key chain. Plaintext authentication is not possible on IOS XR with key chains.

### IS-IS LSP Authentication

IOS nodes enable LSP authentication with the command **authentication key-chain** *key-chain-name* [{**level-1** | **level-2**}] under the IS-IS process. The authentication type is identified with the command **key-chain authentication mode** {**md5** | **text**} [{**level-1** | **level-2**}]. The legacy interface parameter command **domain-password** *password* enables plaintext IS-IS L2 LSP authentication, and the legacy interface parameter command **area-password** *password* enables plaintext IS-IS L1 LSP authentication.

IOS XR nodes enable LSP authentication with the command **lsp-password** {**text** *password* | **hmac-md5** *password* | **keychain** *key-chain-name*} [**level** {**1** | **2**}] under the interface within the IS-IS router configuration.

Note

Key chain configuration was shown earlier in Chapter 5, "EIGRP."

Figure 8-35 demonstrates IS-IS authentication. IS-IS hellos authenticate with the password ISISHELLO using plaintext, L1 LSPs authenticate with the password LOCAL, and L2 LSPs authenticate with the password REMOTE. All LSP passwords use an MD5 hash.

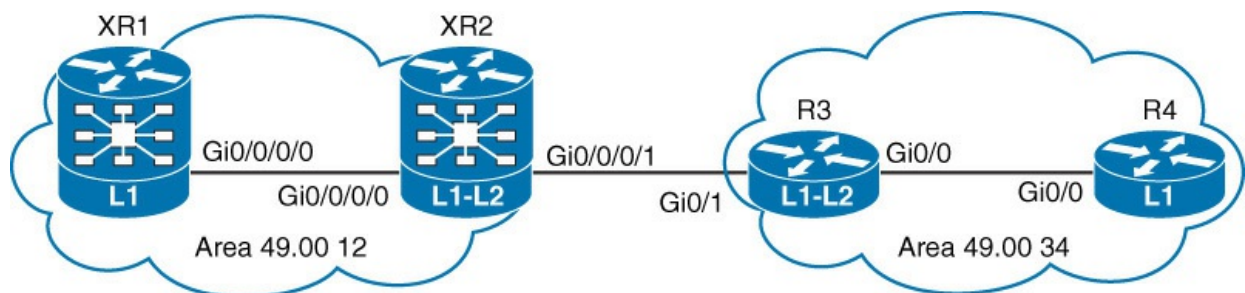


Figure 8-35 IS-IS Authentication

Example 8-20 provides the relevant configuration for all the routers.

### Example 8-20 IS-IS Authentication Configuration

[Click here to view code image](#)

**XR1**

```
key chain L1PASS
  key 1
    key-string LOCAL
    cryptographic-algorithm MD5
  !
!
router isis CISCO
  is-type level-1
  net 49.0012.0000.0000.0001.00
  lsp-password keychain L1PASS level 1
  interface GigabitEthernet0/0/0/0
    hello-password text ISISHELLO
```

**XR2**

```
key chain L1PASS
  key 1
    key-string LOCAL
    cryptographic-algorithm MD5
  !
!
key chain L2PASS
  key 1
    key-string REMOTE
    cryptographic-algorithm MD5
  !
!
router isis CISCO
  net 49.0012.0000.0000.0002.00
  lsp-password keychain L1PASS level 1
  lsp-password keychain L2PASS level 2
  interface GigabitEthernet0/0/0/0
    hello-password text ISISHELLO
  !
  interface GigabitEthernet0/0/0/2
    hello-password text ISISHELLO
```

**R3**

```
key chain L1PASS
  key 1
    key-string LOCAL
!
key chain L2PASS
  key 1
    key-string REMOTE
!
key chain HELLO
  key 1
    key-string ISISHELLO

interface GigabitEthernet0/0
  ip address 10.23.1.3 255.255.255.0
  ip router isis
  isis authentication mode text
  isis authentication key-chain HELLO
!
interface GigabitEthernet0/1
  ip address 10.34.1.3 255.255.255.0
  ip router isis
  isis authentication mode text
  isis authentication key-chain HELLO
!
router isis
  net 49.0034.0000.0000.0003.00
  authentication mode md5
  authentication key-chain L1PASS level-1
  authentication key-chain L2PASS level-2
  log-adjacency-changes
```

**R4**

```
key chain L1PASS
  key 1
    key-string LOCAL
!
interface GigabitEthernet0/1
  ip address 10.34.1.4 255.255.255.0
  ip router isis
  isis password ISISHELLO
!
router isis
  net 49.0034.0000.0000.0004.00
  is-type level-1
  authentication mode md5
  authentication key-chain L1PASS level-1
```

**SUMMARY**

The IS-IS routing protocol is based on the second layer of the OSI model and is not encapsulated in a protocol such as IPv4. This allows IS-IS to support multiple protocols by building the topology first and then overlaying protocols like IPv4 on top of the topology. The IS-IS PDU structure provides a basic structure between the three packet types (hello, link-state packets, and sequence number packets) while providing flexibility through TLVs. IS-IS classifies network interfaces into only two types (broadcast and P2P) removing some of the complexities found with OSPF's five network types.

IS-IS routers check the following items when forming an adjacency:

- L1 routers can form adjacencies with L1 or L1-L2 routers, but not L2.
- L2 routers can form adjacencies with L2 or L1-L2 routers, but not L1.
- Authentication type & credentials (if any).
- L1 adjacencies require the area address to match.
- The system ID must be unique within the same area address.
- Ensure that the protocols supported (OSI, IP, or Both) match.
- Ensure that the interfaces share a common subnet.
- MTU matches.

IS-IS uses a pseudonode (virtual router) to manage LSPDB synchronizations for a broadcast networks. The designated intermediate system (DIS) is an elected router that acts as the pseudonode for the broadcast segment. Inserting the pseudonode into the logical topology for broadcast networks creates multiple logical P2P connections when the IS-IS topology is constructed.

Passive interfaces provide a method of advertising network prefixes into IS-IS, while preventing an adjacency from forming on that interface. Enabling IS-IS authentication provides a mechanism to ensure that only authorized routers can form an adjacency and that the integrity of the LSPs has not been compromised. This prevents manipulation routing table, which could lead to data interception, distributed denial-of-service (DDoS) attacks, or other malicious activity.

The next chapter explains advanced IS-IS topics involving the multilevel topologies, SPF path selection, summarization techniques, and common problematic IS-IS design scenarios.

## REFERENCES IN THIS CHAPTER

ISO, "Intermediate System to Intermediate System Intra-Domain Routing Information Exchange Protocol for Use in Conjunction with the Protocol for Providing the Connectionless-mode Network Service (ISO 8473)," International Standard 10589, 1992.

RFC 941, *Addendum to the Network Service Definition Covering Network Layer Addressing*, McKenzie, Alex, IETF, <http://tools.ietf.org/rfc/rfc941.txt>, April 1985

RFC 5303, *Three-Way Handshake for IS-IS Point-to-Point Adjacencies*, Katz, Dave et al., IETF, <https://tools.ietf.org/html/rfc5303>, October 2008

Doyle, Jeff and Jennifer Carrol. *Routing TCP/IP Volume I, 2nd Edition*. Indianapolis: Cisco Press, 2006.

Martey, Aber. *IS-IS Network Design Solutions*. Indianapolis: Cisco Press, 2002. Print.

Cisco. Cisco IOS Software configuration guides. <http://www.cisco.com>

Cisco. Cisco IOS XR Software configuration guides. <http://www.cisco.com>

## Chapter 9. Advanced IS-IS

This chapter covers the following topics:

- IS-IS interlevel routing
- IS-IS path selection
- IS-IS interface metrics
- Summarization
- Prefix suppression

The Intermediate System-to-Intermediate System (IS-IS) Protocol scales well with proper planning. IS-IS network design should consider all the following factors: IP addressing schemes, level and area segmentation, address summarization, and hardware capabilities for each level.

This chapter expands upon [Chapter 8, “IS-IS,”](#) and explains the functions and features found in larger networks. By the end of this chapter, you should have a solid understanding of route advertisement in a multilevel IS-IS routing domain, path selection, and techniques to optimize an IS-IS environment.

### ADVANCED IS-IS ROUTING

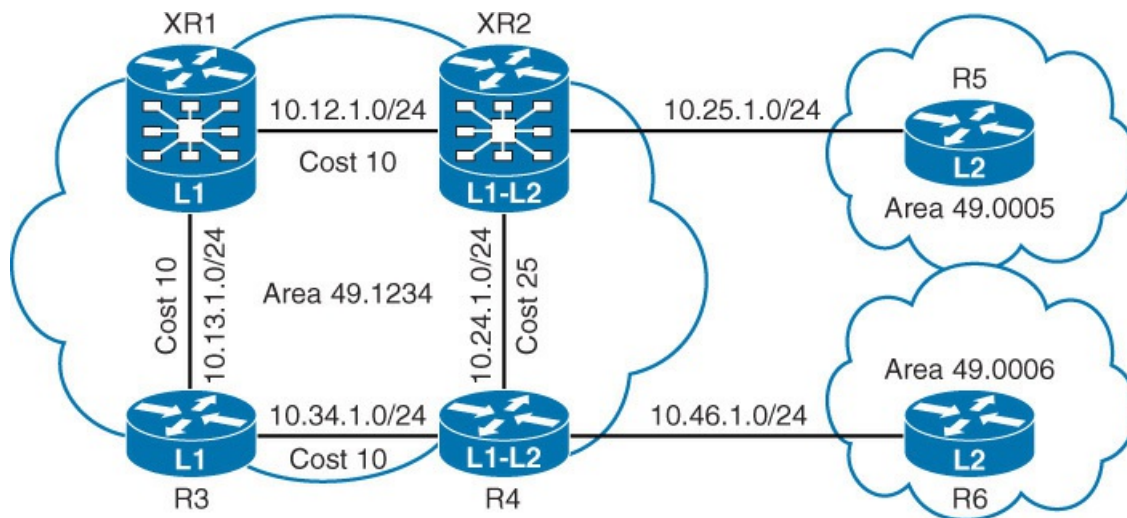
Route scalability is a large factor for the interior gateway protocols (IGP) used by service providers because there can be thousands of routers running in the network. Splitting up an IS-IS routing domain into levels and areas reduces the size of the link-state database (LSPDB) for a router. Newer routers have more memory and faster processors than those in the past; however, because all routers have an identical copy of the LSPDB, IS-IS levels need to accommodate the smallest and slowest router for that level.

Level 1–Level 2 (L1-L2) routers maintain a separate LSPDB for both levels, L1 and L2. Routes from the L1 level are advertised to the L2 topology populating the L1 topology metric into the L2 link-state packet (LSP) metric. L1-L2 routers set the attached bit to their L1 LSPs, providing L1 routers connectivity to networks in a different area. If an L1 router does not have a route for a destination network, it searches for the closest router with the attached bit set in the LSP to forwarding traffic.

Segmenting an IS-IS routing domain into multiple levels (and areas from an L1 perspective) provides the following benefits:

- Hides the topology of other IS-IS levels so topology changes do not require a full shortest path first (SPF) calculation when a transit link flaps in a different level
- Shrinks the size of the LSPDB
- Allows for summarization between IS-IS levels to further shrink the LSPDB

In [Figure 9-1](#), Area 49.1234 connects to Area 49.0005, and Area 49.0006. XR1 and R3 are L1 routers, and XR2 and R4 are L1-L2 routers.



**Figure 9-1** IS-IS Interarea Topology

In [Example 9-1](#), notice that the XR2 and R4 are advertising the highlighted attached bit in their LSPs.

### Example 9-1 IS-IS LSPDB from XR1 and R4

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis database
```

```
! Output omitted for brevity
```

```
IS-IS ISIS (Level-1) Link State Database
```

LSPID	LSP Seq Num	LSP Checksum	LSP Holdtime	ATT/P/OL
XR1.00-00	* 0x00000007	0x8f0d	1108	0/0/0
XR2.00-00	0x00000008	0xb5f0	1075	1/0/0
R3.00-00	0x00000007	0xc211	1106	0/0/0
R4.00-00	0x00000008	0x7d4b	926	1/0/0

```
R3#show isis database
```

```
! Output omitted for brevity
```

```
IS-IS Level-1 Link State Database:
```

LSPID	LSP Seq Num	LSP Checksum	LSP Holdtime	ATT/P/OL
XR1.00-00	0x00000007	0x8F0D	1080	0/0/0
XR2.00-00	0x00000008	0xB5F0	1046	1/0/0
R3.00-00	* 0x00000007	0xC211	1081	0/0/0
R4.00-00	0x00000008	0x7D4B	898	1/0/0

XR1 and R3 have installed default routes into the Routing Information Base (RIB) as shown in [Example 9-2](#). Default routes are not advertised in L1 LSPs. The L1 routers interpret the attached bit as a route of last resort and install a default route into the local routing table. XR2 and R4 contain both L1 and L2 routes in their routing tables.

## Example 9-2 IS-IS Routes Installed in the RIB

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route isis
! Output omitted for brevity

i*L1 0.0.0.0/0 [115/10] via 10.12.1.2, 00:29:29, GigabitEthernet0/0/0/0
! Transit links have been removed from the output, only Loopback addresses exist
i L1 192.168.2.2/32 [115/20] via 10.12.1.2, 00:29:29, GigabitEthernet0/0/0/0
i L1 192.168.3.3/32 [115/20] via 10.13.1.3, 00:29:29, GigabitEthernet0/0/0/2
i L1 192.168.4.4/32 [115/30] via 10.13.1.3, 00:18:57, GigabitEthernet0/0/0/2
```

```
RP/0/0/CPU0:XR2#show route isis
! Output omitted for brevity

! Transit links have been removed from the output, only Loopback addresses exist
i L1 192.168.1.1/32 [115/20] via 10.12.1.1, 00:29:13, GigabitEthernet0/0/0/0
i L1 192.168.3.3/32 [115/30] via 10.12.1.1, 00:18:37, GigabitEthernet0/0/0/0
i L1 192.168.4.4/32 [115/35] via 10.24.1.4, 00:07:02, GigabitEthernet0/0/0/2
i L2 192.168.5.5/32 [115/20] via 10.25.1.5, 00:29:21, GigabitEthernet0/0/0/3
i L2 192.168.6.6/32 [115/45] via 10.24.1.4, 00:07:02, GigabitEthernet0/0/0/2
```

```
R3#show ip route isis
! Output omitted for brevity

i*L1 0.0.0.0/0 [115/10] via 10.34.1.4, 00:31:48, GigabitEthernet0/0
! Transit links have been removed from the output, only Loopback addresses exist
i L1 192.168.1.1 [115/20] via 10.13.1.1, 00:30:46, GigabitEthernet0/1
i L1 192.168.2.2 [115/30] via 10.13.1.1, 00:22:12, GigabitEthernet0/1
i L1 192.168.4.4 [115/20] via 10.34.1.4, 00:31:58, GigabitEthernet0/0
```

```
R4#show ip route isis
! Output omitted for brevity

! Transit links have been removed from the output, only Loopback addresses exist
i L1 192.168.1.1 [115/30] via 10.34.1.3, 00:21:23, GigabitEthernet0/0
i L1 192.168.2.2 [115/35] via 10.24.1.2, 00:08:03, GigabitEthernet0/1
i L1 192.168.3.3 [115/20] via 10.34.1.3, 00:31:09, GigabitEthernet0/0
i L2 192.168.5.5 [115/45] via 10.24.1.2, 00:08:03, GigabitEthernet0/1
i L2 192.168.6.6 [115/20] via 10.46.1.6, 00:31:09, GigabitEthernet0/2
```

The L1-L2 routers advertise the L1 prefixes into the L2 level. In the topology example, XR2 and R4 are advertising the L1 prefixes so that R5 and R6 have connectivity into the Area 49.1234, as shown in [Example 9-3](#).

## Example 9-3 R5 and R6 Route Table

[Click here to view code image](#)



```
R5#show ip route isis
```

```
! Output omitted for brevity
```

```
i L2 10.12.1.0/24 [115/20] via 10.25.1.2, 00:39:30, GigabitEthernet0/2
i L2 10.13.1.0/24 [115/30] via 10.25.1.2, 00:39:13, GigabitEthernet0/2
i L2 10.24.1.0/24 [115/35] via 10.25.1.2, 00:17:07, GigabitEthernet0/2
i L2 10.34.1.0/24 [115/40] via 10.25.1.2, 00:03:07, GigabitEthernet0/2
i L2 10.46.1.0/24 [115/45] via 10.25.1.2, 00:17:04, GigabitEthernet0/2
i L2 192.168.1.1 [115/30] via 10.25.1.2, 00:39:13, GigabitEthernet0/2
i L2 192.168.2.2 [115/20] via 10.25.1.2, 00:39:30, GigabitEthernet0/2
i L2 192.168.3.3 [115/40] via 10.25.1.2, 00:03:07, GigabitEthernet0/2
i L2 192.168.4.4 [115/45] via 10.25.1.2, 00:17:04, GigabitEthernet0/2
i L2 192.168.6.6 [115/55] via 10.25.1.2, 00:17:04, GigabitEthernet0/2
```

```
R6#show ip route isis
```

```
! Output omitted for brevity
```

```
i L2 10.12.1.0/24 [115/40] via 10.46.1.4, 00:03:38, GigabitEthernet0/2
i L2 10.13.1.0/24 [115/30] via 10.46.1.4, 00:03:38, GigabitEthernet0/2
i L2 10.24.1.0/24 [115/35] via 10.46.1.4, 00:17:55, GigabitEthernet0/2
i L2 10.25.1.0/24 [115/45] via 10.46.1.4, 00:17:53, GigabitEthernet0/2
i L2 10.34.1.0/24 [115/20] via 10.46.1.4, 00:40:59, GigabitEthernet0/2
i L2 192.168.1.1 [115/40] via 10.46.1.4, 00:03:38, GigabitEthernet0/2
i L2 192.168.2.2 [115/45] via 10.46.1.4, 00:17:53, GigabitEthernet0/2
i L2 192.168.3.3 [115/30] via 10.46.1.4, 00:03:38, GigabitEthernet0/2
i L2 192.168.4.4 [115/20] via 10.46.1.4, 00:40:59, GigabitEthernet0/2
i L2 192.168.5.5 [115/55] via 10.46.1.4, 00:17:55, GigabitEthernet0/2
```

Depending on the topology, L1 routers may use suboptimal routing paths because a more specific prefix does not exist, and packets are forwarded to the closest router with an attached bit. In the topology example, XR1 uses the path XR1-->XR2-->R4-->R6, which has a total path metric of 35 versus the path XR1-->R3-->R4-->R6 with a total path metric of 30.

Example 9-4 shows the trace route for the path from XR1 to R6, and R3 to R5. Notice that both routers select a suboptimal path.

### Example 9-4 XR1 and R4 Suboptimal Routing

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#traceroute 192.168.6.6
```

```
Type escape sequence to abort.
```

```
Tracing the route to 192.168.6.6
```

```
 1 10.12.1.2 22 msec  2 msec  1 msec
 2 10.24.1.4  2 msec  2 msec  1 msec
 3 10.46.1.6  1 msec  *  5 msec
```

```
R3#traceroute 192.168.5.5
Type escape sequence to abort.
Tracing the route to 192.168.5.5
VRF info: (vrf in name/id, vrf out name/id)
 1 10.34.1.4 0 msec 0 msec 1 msec
 2 10.24.1.2 4 msec 3 msec 3 msec
 3 10.25.1.5 6 msec * 5 msec
```

## Route Leaking

Suboptimal routing can be corrected by route-leaking, a technique that redistributes the L2 level routes into the L1 level. Route leaking normally uses a restrictive route map or route policy to control which routes are leaked; otherwise, running all the area routers in L2 mode makes more sense.

IS-IS route leaking is configured on IOS nodes with the IS-IS configuration command

[Click here to view code image](#)

```
redistribute isis ip {level-1 | level-2} into {level-1 | level-2} route-map route-
map-name
```

IOS XR nodes use the command **propagate level {1 | 2} into level {1 | 2} route-policy route-policy-name** under the address family within the IS-IS router configuration.

[Example 9-5](#) provides the relevant configuration of XR2 and R4 leaking L2 routes into L1. Redistribution, route maps, and route policies are explained in greater detail in [Chapter 11](#), “Route Maps and Route Policy,” and [Chapter 13](#), “Route Redistribution.”

## Example 9-5 XR2 and R4 Route Leaking

[Click here to view code image](#)

```
XR2
route-policy PASS-ALL
  pass
end-policy

router isis CISCO
 net 49.1234.0000.0000.0002.00
 log adjacency changes
 address-family ipv4 unicast
 propagate level 2 into level 1 route-policy PASS-ALL
```

```
R2
router isis
 net 49.1234.0000.0000.0004.00
 log-adjacency-changes
 redistribute isis ip level-2 into level-1 route-map PASS-ALL
!
route-map PASS-ALL permit 10
```

[Example 9-6](#) shows the route table for XR1 and R3 after the L2 routes have been leaked into the L1 level. Notice that the 192.168.5.5/32 and 192.168.6.6/32 show as *ia* in the routing table. RFC 2966 explicitly states that leaked routes are interarea and external to the L1 area because they were redistributed into the L1 area from an L2 router.

### Example 9-6 XR1 and R2 Route Table with Leaked Routes

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route isis
! Output omitted for brevity

i*L1 0.0.0.0/0 [115/10] via 10.12.1.2, 00:12:13, GigabitEthernet0/0/0/0
i L1 10.24.1.0/24 [115/35] via 10.12.1.2, 00:12:13, GigabitEthernet0/0/0/0
i L1 10.25.1.0/24 [115/20] via 10.12.1.2, 00:12:13, GigabitEthernet0/0/0/0
i L1 10.34.1.0/24 [115/20] via 10.13.1.3, 00:16:35, GigabitEthernet0/0/0/2
i L1 10.46.1.0/24 [115/30] via 10.13.1.3, 00:16:34, GigabitEthernet0/0/0/2
i L1 192.168.2.2/32 [115/20] via 10.12.1.2, 00:13:01, GigabitEthernet0/0/0/0
i L1 192.168.3.3/32 [115/20] via 10.13.1.3, 00:16:35, GigabitEthernet0/0/0/2
i L1 192.168.4.4/32 [115/30] via 10.13.1.3, 00:16:34, GigabitEthernet0/0/0/2
i ia 192.168.5.5/32 [115/30] via 10.12.1.2, 00:06:48, GigabitEthernet0/0/0/0
i ia 192.168.6.6/32 [115/40] via 10.13.1.3, 00:01:57, GigabitEthernet0/0/0/2
```

```
R3#show ip route isis
! Output omitted for brevity

Gateway of last resort is 10.34.1.4 to network 0.0.0.0

i*L1 0.0.0.0/0 [115/10] via 10.34.1.4, 00:25:35, GigabitEthernet0/0
i L1 10.12.1.0/24 [115/20] via 10.13.1.1, 00:17:31, GigabitEthernet0/1
i L1 10.24.1.0/24 [115/30] via 10.34.1.4, 00:25:45, GigabitEthernet0/0
i L1 10.25.1.0/24 [115/30] via 10.13.1.1, 00:13:09, GigabitEthernet0/1
i L1 10.46.1.0/24 [115/20] via 10.34.1.4, 00:25:45, GigabitEthernet0/0
i L1 192.168.1.1 [115/20] via 10.13.1.1, 00:17:31, GigabitEthernet0/1
i L1 192.168.2.2 [115/30] via 10.13.1.1, 00:13:58, GigabitEthernet0/1
i L1 192.168.4.4 [115/20] via 10.34.1.4, 00:25:45, GigabitEthernet0/0
i ia 192.168.5.5 [115/168] via 10.13.1.1, 00:07:48, GigabitEthernet0/1
i ia 192.168.6.6 [115/158] via 10.34.1.4, 00:02:57, GigabitEthernet0/0
```

[Example 9-7](#) shows the IS-IS LSP entry from XR2 and R4. Notice that 192.168.5.5/32 and 192.168.6.6/32 entries show as *IP-InterArea* and indicates that the information was received in TLV 130 (IP External Reachability Information) versus TLV 128 (IP Internal Reachability Information).

TLV 130 allows for the differentiation between internal and external metric types. Cisco uses the internal type by default, and the two path types are explained in more detail later in this chapter, in the section [“Path Selection.”](#)

### Example 9-7 XR2 and R4 Interarea TLVs

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis database XR2.00-00 detail
```

```
IS-IS ISIS (Level-1) Link State Database
```

LSPID	LSP Seq Num	LSP Checksum	LSP Holdtime	ATT/P/OL
XR2.00-00	0x00000005	0xa9b9	1056	1/0/0
Area Address: 49.1234				
NLPID:	0xcc			
Hostname:	XR2			
IP Address:	192.168.2.2			
Metric: 10	IS XR1.00			
Metric: 25	IS R4.00			
Metric: 10	IP 10.12.1.0/24			
Metric: 25	IP 10.24.1.0/24			
Metric: 10	IP 10.25.1.0/24			
Metric: 10	IP 192.168.2.2/32			
Metric: 20	IP-Interarea 192.168.5.5/32			
Metric: 45	IP-Interarea 192.168.6.6/32			

```
R3#show isis database R4.00-00 detail
```

```
IS-IS Level-1 LSP R4.00-00
```

LSPID	LSP Seq Num	LSP Checksum	LSP Holdtime	ATT/P/OL
R4.00-00	0x00000005	0xF320	1193	1/0/0
Area Address: 49.1234				
NLPID:	0xCC			
Hostname:	R4			
IP Address:	192.168.4.4			
Metric: 10	IP 192.168.4.4 255.255.255.255			
Metric: 10	IP 10.34.1.0 255.255.255.0			
Metric: 20	IP 10.24.1.0 255.255.255.0			
Metric: 10	IP 10.46.1.0 255.255.255.0			
Metric: 20	IS XR2.00			
Metric: 10	IS R3.00			
Metric: 168	IP-Interarea 192.168.5.5 255.255.255.255			
Metric: 148	IP-Interarea 192.168.6.6 255.255.255.255			

## Backbone Continuity

As stated in [Chapter 8](#), the L2 backbone must be contiguous to ensure that connectivity is maintained between the endpoint areas. [Figure 9-2](#) demonstrates three areas that are advertising transit links and Loopback 0 addresses into IS-IS. Connectivity problems between XR1 (Area 49.0001) and R5 (Area 49.0005) exist because a junior network engineer configured R3 as an L1 router to reduce router resources.

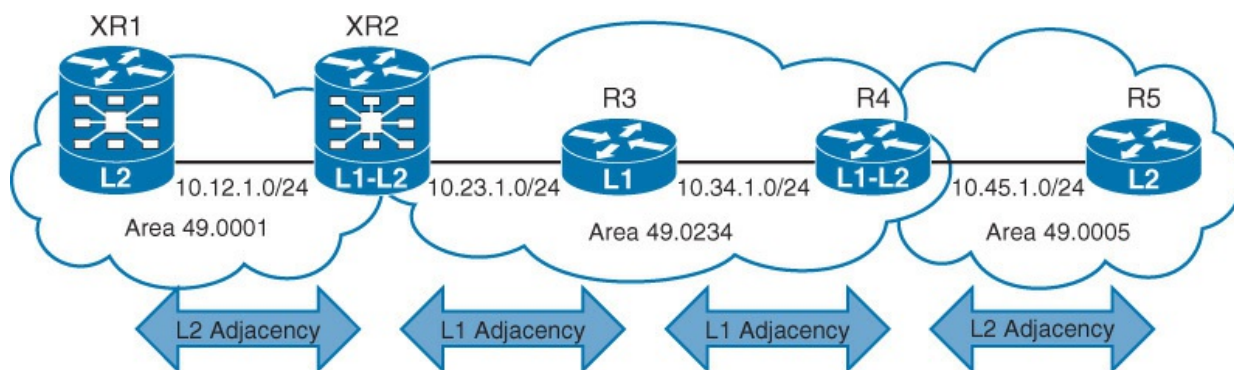


Figure 9-2 Noncontiguous IS-IS L2 Backbone

Figure 9-3 provides the IS-IS IP routing table for all five routers. XR1 does not have a route for the 192.168.5.5/32 network, nor does it have a route of last resort directed toward R5. R5 does not have a route for the 192.168.1.1/32 network or a route of last resort.

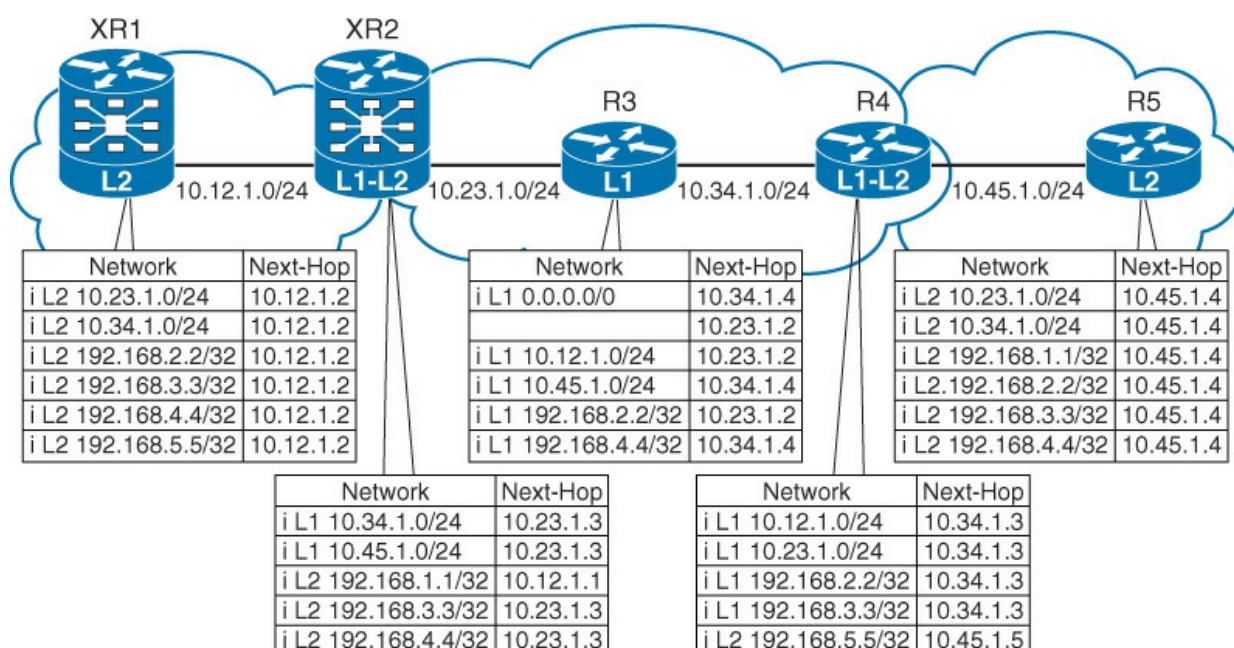


Figure 9-3 Routing Tables for Noncontiguous IS-IS L2 Backbone

XR2 is missing the 192.168.5.5/32 network and does not install a default route by using the attached bit in R4's LSP. R4 is missing the 192.168.1.1/32 network and does not have a default route installed for the XR2's attached bit either.

Example 9-8 verifies that R4 and XR2 are setting the attached bit in the LSP. IS-IS routers with L1-L2 connectivity do not install a default route using the attached bit.

### Example 9-8 XR1 and R4 Suboptimal Routing

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show isis database
```

```
! Output omitted for brevity
```

```
IS-IS ISIS (Level-1) Link State Database
```

LSPID	LSP Seq Num	LSP Checksum	LSP Holdtime	ATT/P/OL
XR2.00-00	* 0x00000019	0xeb0d	991	1/0/0
R3.00-00	0x00000006	0x5f73	963	0/0/0
R4.00-00	0x00000018	0x44dd	810	1/0/0

```
R4#show isis database
```

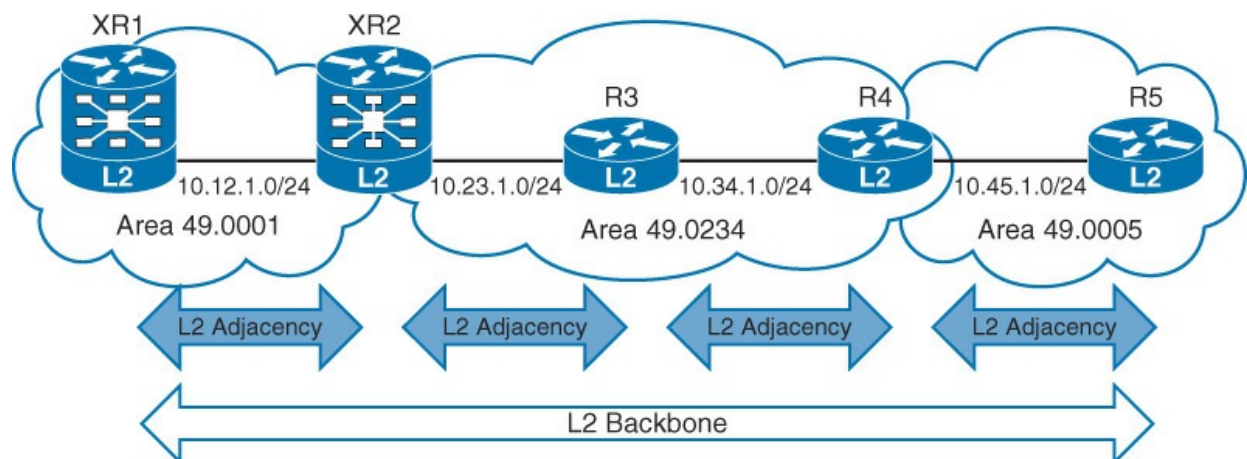
```
! Output omitted for brevity
```

```
IS-IS Level-1 Link State Database:
```

LSPID	LSP Seq Num	LSP Checksum	LSP Holdtime	ATT/P/OL
XR2.00-00	0x00000002	0xD33E	546	1/0/0
R3.00-00	0x00000007	0x38A0	494	0/0/0
R4.00-00	* 0x00000008	0xFB3B	612	1/0/0

Even if XR2 and R4 installed a default route using the attached bit, connectivity between R3 would be sporadic because the attached bit points to XR2 and R4, inferring that R3 has a 50 percent chance of sending traffic out the correct interface.

The solution to the issue is to provide R3 with L2 routing capability that extends the backbone through all three areas. Area 49.0234 does not need L1 routing capability, because all the network prefixes are advertised into L2. Removing the L1 routing removes the burden of maintaining the two link-state packet databases (LSPDBs) and performing SPF calculation twice. [Figure 9-4](#) demonstrates the solution for the previous topology.



**Figure 9-4** Contiguous IS-IS L2 Backbone

### Loop Prevention

[Figure 9-5](#) expands the topology example to include R6 within Area 49.0234. R6 is an L1-only router and establishes L1 sessions with XR2 and R4. XR2 and R4 act as L1-L2 routers to provide connectivity to R6. R3 is an L2 router and establishes L2 sessions with XR2 and R4 to maintain backbone contiguity between the areas.

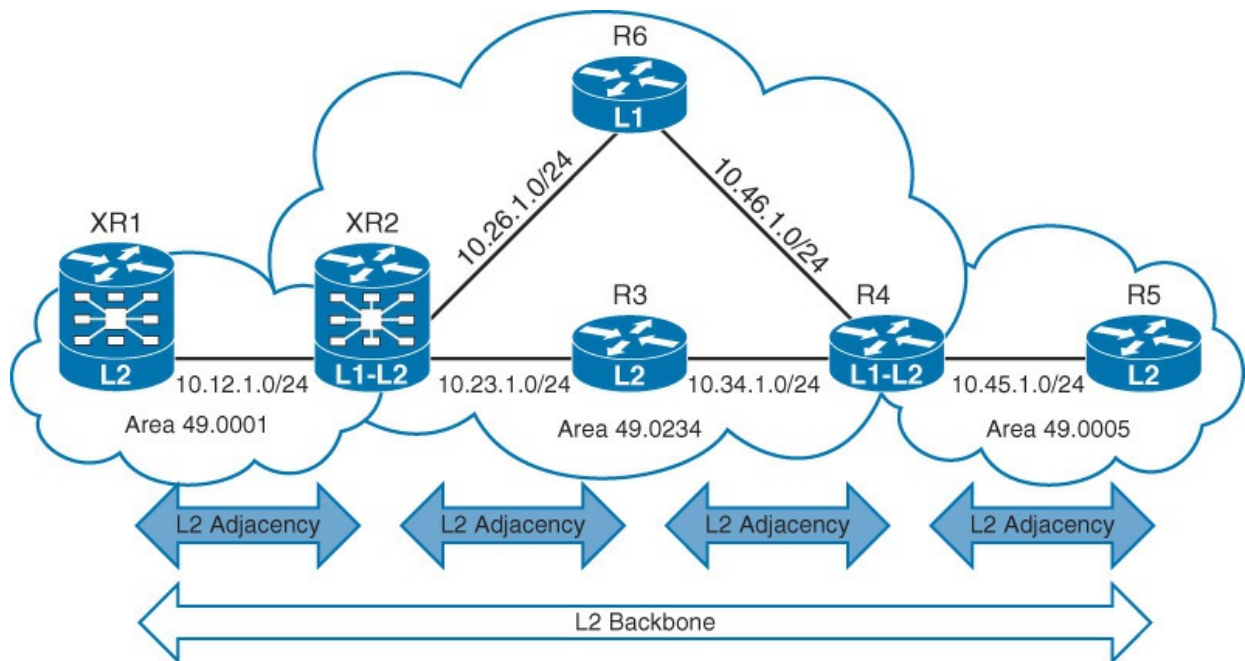


Figure 9-5 Contiguous IS-IS L2 Backbone to Demonstrate L1 ISDPB Population

Example 9-9 displays that the routing tables for XR1, R3, and R5 have full reachability.

### Example 9-9 XR1 and R5 Routing Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route isis
! Output omitted for brevity
! Transit links have been removed from the output, only Loopback addresses exist
i L2 192.168.2.2/32 [115/20] via 10.12.1.2, 03:14:09, GigabitEthernet0/0/0/0
i L2 192.168.3.3/32 [115/30] via 10.12.1.2, 00:00:09, GigabitEthernet0/0/0/0
i L2 192.168.4.4/32 [115/40] via 10.12.1.2, 00:00:58, GigabitEthernet0/0/0/0
i L2 192.168.5.5/32 [115/50] via 10.12.1.2, 00:00:05, GigabitEthernet0/0/0/0
i L2 192.168.6.6/32 [115/30] via 10.12.1.2, 00:01:10, GigabitEthernet0/0/0/0
```

```
R3#show ip route isis
! Output omitted for brevity
! Transit links have been removed from the output, only Loopback addresses exist
i L2 192.168.1.1 [115/30] via 10.23.1.2, 00:18:37, GigabitEthernet0/0
i L2 192.168.2.2 [115/20] via 10.23.1.2, 00:18:37, GigabitEthernet0/0
i L2 192.168.4.4 [115/20] via 10.34.1.4, 00:18:37, GigabitEthernet0/1
i L2 192.168.5.5 [115/30] via 10.34.1.4, 00:18:37, GigabitEthernet0/1
i L2 192.168.6.6 [115/30] via 10.34.1.4, 00:18:37, GigabitEthernet0/1
[115/30] via 10.23.1.2, 00:18:37, GigabitEthernet0/0
```

```
R5#show ip route isis
! Output omitted for brevity
```

```
! Transit links have been removed from the output, only Loopback addresses exist
i L2    192.168.1.1 [115/50] via 10.45.1.4, 00:00:43, GigabitEthernet0/0
i L2    192.168.2.2 [115/40] via 10.45.1.4, 00:01:30, GigabitEthernet0/0
i L2    192.168.3.3 [115/30] via 10.45.1.4, 00:00:43, GigabitEthernet0/0
i L2    192.168.4.4 [115/20] via 10.45.1.4, 03:58:26, GigabitEthernet0/0
i L2    192.168.6.6 [115/30] via 10.45.1.4, 00:01:30, GigabitEthernet0/0
```

[Example 9-10](#) displays R6's routing table. Notice that R3's loopback address (192.168.3.3/32) is not in the routing table, even though R3 and R6 are in the same area. L1-L2 routers will not advertise an L2 route with the same area address as theirs into the L1 area. This behavior prevents routing loops. Connectivity between R3 and R6 is still possible through the attached bits set on the LSPs of XR2 and R4.

### Example 9-10 R6 Routing Table

[Click here to view code image](#)

```
R6#show ip route isis
! Output omitted for brevity
```

```
Gateway of last resort is 10.46.1.4 to network 0.0.0.0

i*L1 0.0.0.0/0 [115/10] via 10.46.1.4, 00:21:12, Ethernet0/2
      [115/10] via 10.26.1.2, 00:21:12, Ethernet0/0
i L1  10.12.1.0/24 [115/20] via 10.26.1.2, 00:21:24, Ethernet0/0
i L1  10.23.1.0/24 [115/20] via 10.26.1.2, 00:17:50, Ethernet0/0
i L1  10.34.1.0/24 [115/20] via 10.46.1.4, 00:13:22, Ethernet0/2
i L1  10.45.1.0/24 [115/20] via 10.46.1.4, 00:21:12, Ethernet0/2
i L1  192.168.2.2 [115/20] via 10.26.1.2, 00:21:24, Ethernet0/0
i L1  192.168.4.4 [115/20] via 10.46.1.4, 00:21:12, Ethernet0/2
```

[Example 9-11](#) verifies that R3's loopback address was advertised as the Type 2 LSP.

### Example 9-11 R3's L2 LSP

[Click here to view code image](#)



```
R4#show isis database R3.00-00 detail
```

```
IS-IS Level-2 LSP R3.00-00
LSPID                LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
R3.00-00             0x00000008   0x9737        790           0/0/0
  Area Address: 49.0234
  NLPID:         0xCC
  Hostname: R3
  IP Address:   192.168.3.3
  Metric: 10    IS R3.02
  Metric: 10    IS R4.04
  Metric: 10    IP 10.23.1.0 255.255.255.0
  Metric: 10    IP 10.34.1.0 255.255.255.0
  Metric: 10    IP 192.168.3.3 255.255.255.255
```

### Router-Specific IS-IS Levels

IS-IS routers do not have a mechanism to detect whether their area is at the end (edge) or middle (transit) of the L2 backbone. In Figure 9-34, R3 does not know whether it is routing transit traffic between Area 49.0001 and 49.0005. Only humans can identify an area as transit, so Cisco defaults to making all routers L1-L2. The default behavior guarantees that all routers will be able to route transit traffic but also limits scalability of the protocol.

The IS-IS level that a router operates at can be set on IOS and IOS XR nodes with the IS-IS router configuration command **is-type {level-1 | level-1-2 | level-2-only}**. [Example 9-12](#) demonstrates the configuration for explicitly setting the IS-IS router level for XR1 and R3 from [Figure 9-5](#).

### Example 9-12 XR1 and R3 as L2 Routers

[Click here to view code image](#)

```
XR1
router isis CISCO
  is-type level-2-only
  net 49.0001.0000.0000.0001.00

interface GigabitEthernet0/0/0/0
  address-family ipv4 unicast
```

```
R3
router isis
  net 49.0234.0000.0000.0002.00
  is-type level-2-only
```

### Interface Specific IS-IS Levels

Other topology designs may specify that a specific interface should establish only a specific IS-IS level adjacency. This can be accomplished with the interface parameter command **isis circuit-type {level-1 | level-1-2 | level-2-only}** on IOS nodes, and IOS XR nodes use the command **circuit-type {level-1 | level-1-2 | level-2-only}**.

[Figure 9-6](#) provides a simple topology between XR1 and R2 in Area 49.0012. Both routers need to stay as L1-L2 routers but should form only an L1 adjacency.

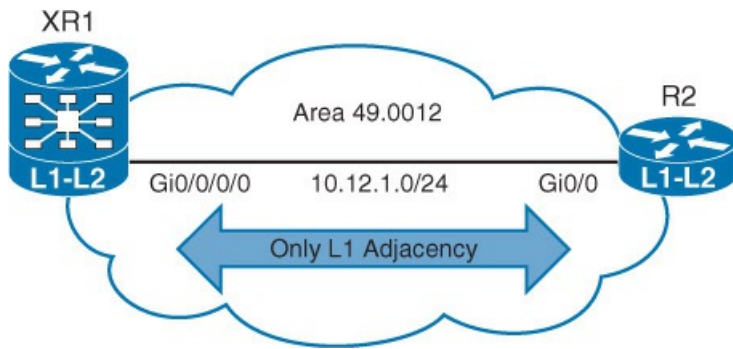


Figure 9-6 Interface-Specific IS-IS Circuit Types Topology

Example 9-13 demonstrates the configuration for XR1 and R2 for setting the IS-IS level for the interfaces between XR1 and R2.

### Example 9-13 XR1 and R2 with Restricted L1 Connectivity

[Click here to view code image](#)

```
XR1
router isis CISCO
 net 49.0012.0000.0000.0001.00
!
interface GigabitEthernet0/0/0/0
 circuit-type level-1
 address-family ipv4 unicast
```

```
R2
interface GigabitEthernet0/0
 ip address 10.12.1.2 255.255.255.0
 isis circuit-type level-1

router isis
 net 49.0012.0000.0000.0002.00
```

It is possible to set the router to a specific IS-IS level functionality with a different setting for a circuit from the router-specific setting. When the settings are combined, the router uses the most restrictive level when forming an adjacency. Table 9-1 displays the capable adjacencies for a router based solely on the IS-IS router type and IS-IS circuit-type.

	Router Set IS-IS Type L1	Router Set IS-IS Type L2	Router Set IS-IS Type L1-L2
Circuit Type L1	Level-1	None	Level-1
Circuit Type L2	None	Level-2	Level-2
Circuit Type L1-L2	Level-1	Level-2	Level-1 and Level-2

Table 9-1 IS-IS Neighbor Adjacency Capability Chart

## PATH SELECTION

IS-IS path selection is quite straightforward after reviewing the following key definitions:

■ **Intra-area routes** are routes that are learned from another router within the same level and area address.

■ **Interarea routes** are routes that are learned from another L2 router that came from an L1 router or from an L2 router from a different area address.

■ **External routes** are routes that are redistributed into the IS-IS domain. External routes can choose between two metric types:

■ **Internal metrics** are directly comparable with IS-IS path metrics and are used during redistribution by default by IOS and IOS XR. IS-IS treats these routes with the same preferences as those advertised normally via TLV 128.

■ **External metrics** cannot be comparable with internal path metrics and must be set in a route map or route policy during redistribution. Internal metrics are always preferred to external metrics.

IS-IS best-path selection uses the following processing order, identifying the route with the lowest path metric for each stage:

1. L1 intra-area routes

L1 external routes with internal metrics

2. L2 intra-area routes

L2 external routes with internal metric

L1 --> L2 interarea routes

L1 --> L2 interarea external routes with internal metrics

3. Leaked routes (L2 --> L1) with internal metrics

4. L1 external routes with external metrics

5. L2 external routes with external metric

L1 --> L2 interarea external routes with external metrics

6. Leaked routes (L2 --> L1) with external metrics

Note

Under normal IS-IS configuration, only the first three steps are used; external routes with external metrics require the external metric type to be explicitly specified in the route map or route policy at the time of redistribution.

## Equal Cost Multi-Path

If IS-IS identifies multiple best-paths after performing the best-path selection processing, then those routes will be installed into the routing table as Equal Cost Multi-Path (ECMP) routing. The default maximum ECMP paths for IOS nodes are 4 routes and 16 routes for IOS XR nodes.

The setting can be changed on IOS nodes with the command **maximum-paths** *maximum-paths* under the IS-IS process. IOS XR nodes can change the value with the command **maximum-paths** *maximum-paths* under the appropriate address family within the IS-IS router configuration.

## Interface Metrics

The IS-IS interface metric is a vital component within the LSP. RFC 1195 specified that the interface metric as a 6-bit field that supports a value between 1 and 63. The metrics are included in the IS Neighbors TLV (2) and IP Reachability TLVs (128 and 130) and are commonly referred to as *narrow metrics*. IS-IS assigns a default metric of 10 to all interfaces regardless of the interface bandwidth. A 1-Mbps link uses the same path metric as a 1-Gbps link by default.

Setting all interfaces to the same metric can lead to suboptimal routing, and some designs might require changing the interface metric. Limiting the interface metric to 63 introduces other issues dependent on the variance of network bandwidth in an IS-IS topology. Correlating a 40-Gigabit interface to a 1-Gigabit interface could be difficult, especially if the area has fractional T1s in it.

### Note

Although the IS-IS link metric is limited to 63 because of LSP limitations, the total path metric can exceed that value because the cost is added up locally from the LSPDB.

RFC 3784 introduced the concept of sub-TLVs for Multiprotocol Label Switching (MPLS) Traffic Engineering and also provided a method for the interface metric to use a 24-bit number that allows for the metric to be set between 1 and 16,777,214. The 24-bit metrics are available in the Extended IS Reachability TLV 22 and the Extended IP Reachability TLV#135 and are commonly referred to as *wide metrics*.

Table 9-2 shows the difference between wide and narrow metrics.

	TLV Number	Interface Metric	Metric Style
IS Neighbors	2	1-63	Narrow
Extended IS Neighbors	22	1-16,777,214	Wide
IS Reachability	128 and 130	63	Narrow
Extended IS Reachability	135	1-16,777,214	Wide

Table 9-2 IS-IS Narrow and Wide Metrics

IOS and IOS XR use narrow metrics by default for the IPv4 address family. Enabling wide metrics on one router while using narrow metrics on the adjacent router will cause problems because the two TLVs are independent of each other.

In Figure 9-7, XR1 and XR2 are configured with wide metrics, and R3 and R4 are configured with narrow metrics. All routers will still form the appropriate IS-IS adjacency and can still build a

complete IS-IS topology, but IS-IS will not be able to build a complete SPF tree.

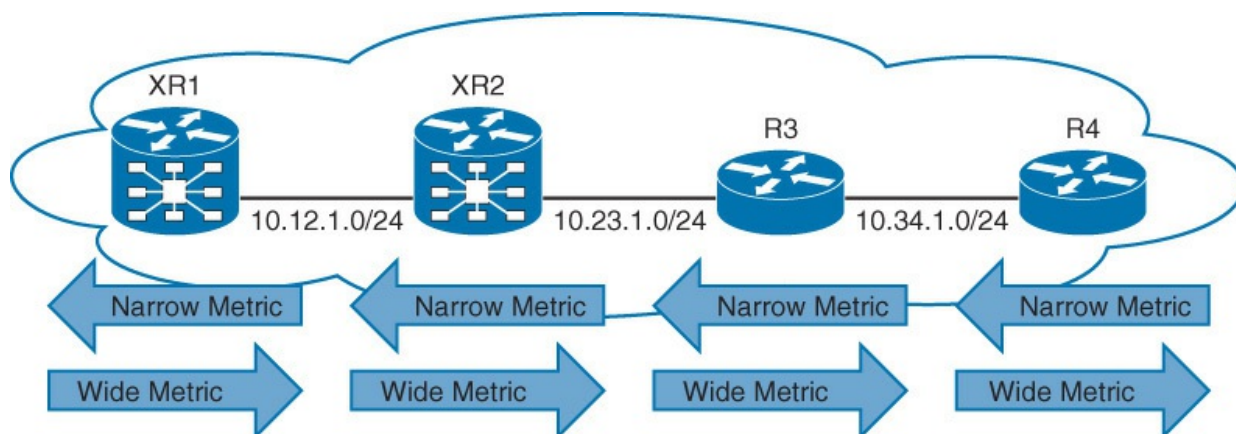


Figure 9-7 IS-IS Wide and Narrow Metric Problems

In [Example 9-14](#), the double asterisks (\*\*) indicate that the router cannot compute the path metric to those nodes because of a metric style mismatch. XR1 can calculate a path metric to XR2, but cannot calculate a path metric to R3 and R4. The same effect occurs from R4's perspective, where R4 can calculate a path metric to R3, but not to XR1 or XR2.

### Example 9-14 IS-IS Topology with Mismatch Metrics

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis topology
```

```
IS-IS ISIS paths to IPv4 Unicast (Level-1) routers
```

System Id	Metric	Next-Hop	Interface	SNPA
XR1	--			
XR2	10	XR2	Gi0/0/0/0	0267.5e1e.d0f7
R3	**			
R4	**			

```
R4#show isis topology
```

```
IS-IS TID 0 paths to level-1 routers
```

System Id	Metric	Next-Hop	Interface	SNPA
XR1	**			
XR2	**			
R3	10	R3	Gi0/1	aabb.cc00.6610
R4	--			

[Example 9-15](#) shows that XR1 can see the R4's narrow metrics and that R4 can see XR1's extended metrics. However, XR1 is using wide metrics for the SPF calculation, and R4 has not advertised wide metric. R4 is using narrow metrics for the SPF calculation, and XR1 has not advertised narrow metrics; therefore, both routers cannot compute a complete SPF algorithm to each other.

### Example 9-15 XR1 and R4 LSPs with Different TLVs

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis database R4.00-00 detail
```

```
IS-IS ISIS (Level-1) Link State Database
LSPID                LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
R4.00-00             0x0000000a   0xdb34        990            0/0/0
  Area Address: 49.1234
  NLPID:         0xcc
  Hostname:      R4
  IP Address:    192.168.4.4
  Metric: 10     IP 192.168.4.4/32
  Metric: 10     IP 10.34.1.0/24
  Metric: 10     IS R4.02
```

```
R4#show isis database XR1.00-00 detail
```

```
IS-IS Level-1 LSP XR1.00-00
LSPID                LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
XR1.00-00           0x00000003   0xc55A        884            0/0/0
  Area Address: 49.1234
  NLPID:         0xCC
  Hostname:      XR1
  IP Address:    192.168.1.1
  Metric: 10     IS-Extended XR2.03
  Metric: 10     IP 10.12.1.0/24
  Metric: 10     IP 192.168.1.1/32
```

Trying to convert a 24-bit value to 6-bit value is not viable, and RFC 3787 outlines the process of converting an IS-IS topology from narrow to wide without affecting the IS-IS routing topology. The first phase involves setting the router into a transitional mode, where wide and narrow TLVs are advertised at the same time. Once all routers are operating in a transitional state, they may be converted to wide metrics and can set the interface metric above the 63 value.

IOS nodes can configure the metric style under the IS-IS router configuration with the command **metric-style {narrow | wide | transition} [level-1 | level-2 | level-1-2]**. IOS XR nodes define the metric style under the address family within the IS-IS router configuration using the command **metric-style {narrow | wide | transition} [level {1|2}]**. The metric style for each IS-IS level can be set independently. If the optional level is not specified, the L1-L2 setting is inferred.

The metric for a specific interface is configured on IOS nodes with the interface parameter command **isis metric {1-16777214 | maximum} [level-1 | level-2]**. IOS XR nodes set the metric with the command **metric {1-16777214 | maximum} [level {1|2}]** under the address family associated to the interface. Values over 63 should only be used with wide metrics.

Using the **maximum** keyword removes the link from SPF calculation because RFC 3784 states that links advertised with the maximum metric (224 – 1) must not be considering during SPF calculation.

You can set a default metric for all interfaces on an IOS node with the command **metric 1-16777214 [level-1 | level-2]** under the IS-IS router configuration. IOS XR nodes define the default metric under the address-family within the IS-IS router configuration with the command **metric 1-16777214 [level {1|2}]**.

Figure 9-8 provides a reference topology to show the configuration. XR1 and R2 will enable narrow metrics on the L1 adjacency and wide metrics for the L2 adjacency. A default metric of 50 for L1 interfaces, and a default metric of 5000 for L2 interfaces. Interfaces on the 10.12.1.0/24 network will explicitly be set with an L1 metric of 11 and an L2 metric of 2000.



Figure 9-8 IS-IS Metric Topology

Example 9-16 provides the relevant configuration.

### Example 9-16 IS-IS Metric Configuration

[Click here to view code image](#)

```
XR1
router isis CISCO
net 49.0012.0000.0000.0001.00
address-family ipv4 unicast
metric-style narrow level 1
metric-style wide level 2
metric 50 level 1
metric 5000 level 2
!
interface Loopback0
address-family ipv4 unicast
!
interface GigabitEthernet0/0/0/0
address-family ipv4 unicast
!
!
interface GigabitEthernet0/0/0/2
address-family ipv4 unicast
metric 11 level 1
metric 2000 level 2
!
!
```

## R2

```
interface Loopback0
 ip address 192.168.2.2 255.255.255.255
 ip router isis
!
interface GigabitEthernet0/0
 ip address 10.12.1.2 255.255.255.0
 ip router isis
 isis metric 11 level-1
 isis metric 2000 level-2
!
interface GigabitEthernet0/1
 ip address 10.2.2.2 255.255.255.0
 ip router isis

router isis
 net 49.0012.0000.0000.0002.00
 metric-style narrow level-1
 metric-style wide level-2
 metric 50 level-1
 metric 5000 level-2
```

### Note

This example is not a normal configuration, but it illustrates most of the various options for configuring IS-IS metrics.

**Example 9-17** verifies that the metrics match between XR1 and R2 for both IS-IS levels.

### Example 9-17 IS-IS Interface Metric Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis topology
```

```
IS-IS ISIS paths to IPv4 Unicast (Level-1) routers
```

System Id	Metric	Next-Hop	Interface	SNPA
XR1	--			
R2	11	R2	Gi0/0/0/2	aabb.cc00.6500

```
IS-IS ISIS paths to IPv4 Unicast (Level-2) routers
```

System Id	Metric	Next-Hop	Interface	SNPA
XR1	--			
R2	2000	R2	Gi0/0/0/2	aabb.cc00.6500



```
R2#show isis topology
```

```
IS-IS TID 0 paths to level-1 routers
```

System Id	Metric	Next-Hop	Interface	SNPA
XR1	11	XR1	Et0/0	0276.8250.7078
R2	--			

```
IS-IS TID 0 paths to level-2 routers
```

System Id	Metric	Next-Hop	Interface	SNPA
XR1	2000	XR1	Et0/0	0276.8250.7078
R2	--			

**Example 9-18** shows that the default metric was applied to XR1's Loopback 0 and Gio/o/o/o interface, and that Gio/o/o/2 preempted the default setting as expected. Notice that the L1 LSP uses the normal IP Reachability TLV and the L2 LSP uses the Extended IP Reachability TLV.

### Example 9-18 IS-IS Interface IIF Frequency

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis database XR1.00-00 level 1 detail
```

```
IS-IS ISIS (Level-1) Link State Database
```

LSPID	LSP Seq Num	LSP Checksum	LSP Holdtime	ATT/P/OL
XR1.00-00	* 0x00000006	0x67ba	624	0/0/0

Area Address: 49.0012  
NLPID: 0xcc  
Hostname: XR1  
IP Address: 192.168.1.1  
Metric: 11 IS R2.01  
Metric: 50 IP 10.1.1.0/24  
Metric: 11 IP 10.12.1.0/24  
Metric: 50 IP 192.168.1.1/32

```
RP/0/0/CPU0:XR1#show isis database XR1.00-00 level 2 detail
```

```
! Output omitted for brevity
```

```
IS-IS ISIS (Level-2) Link State Database
```

LSPID	LSP Seq Num	LSP Checksum	LSP Holdtime	ATT/P/OL
XR1.00-00	* 0x0000000a	0xda3e	681	0/0/0

Area Address: 49.0012  
NLPID: 0xcc  
Hostname: XR1  
IP Address: 192.168.1.1  
Metric: 2000 IS-Extended R2.01  
Metric: 5000 IP-Extended 10.1.1.0/24  
Metric: 2000 IP-Extended 10.12.1.0/24  
Metric: 5000 IP-Extended 192.168.1.1/32

## Overload Bit

The overload bit indicates when a router is in an overloaded condition. During the IS-IS SPF calculation, routers avoid sending traffic through routers that set the overload bit. Upon recovery, the router advertises a new LSP without the overload bit, and the SPF calculation occurs normally without avoiding routes through the previously overloaded node.

Originally, the overload bit signified memory exhaustion, but current routers have a significant amount of memory making those situations very rare. Setting the overload bit on a router during maintenance windows is a common technique to route traffic around the nodes being worked on. Newer IS-IS functionality allows a router to set the overload bit when it first starts up. This allows the router to form IS-IS or BGP adjacencies, and load the full routing table to avoid blackholing traffic.

The overload bit is set on IOS nodes with the command **set-overload-bit [on-startup] [{5-86400 | wait-for-bgp}]** under the IS-IS router configuration. IOS XR nodes can change the value with the command **set-overload-bit [on-startup {5-86400 | wait-for-bgp}] [level {1 | 2}]** under the IS-IS router configuration. Using the **wait-for-bgp** option maintains the overload bit until all BGP sessions converge or 10-minutes, whichever comes first.

Figure 9-9 provides a reference topology where XR1 is connecting to the 192.168.4.4/32 network on R4.

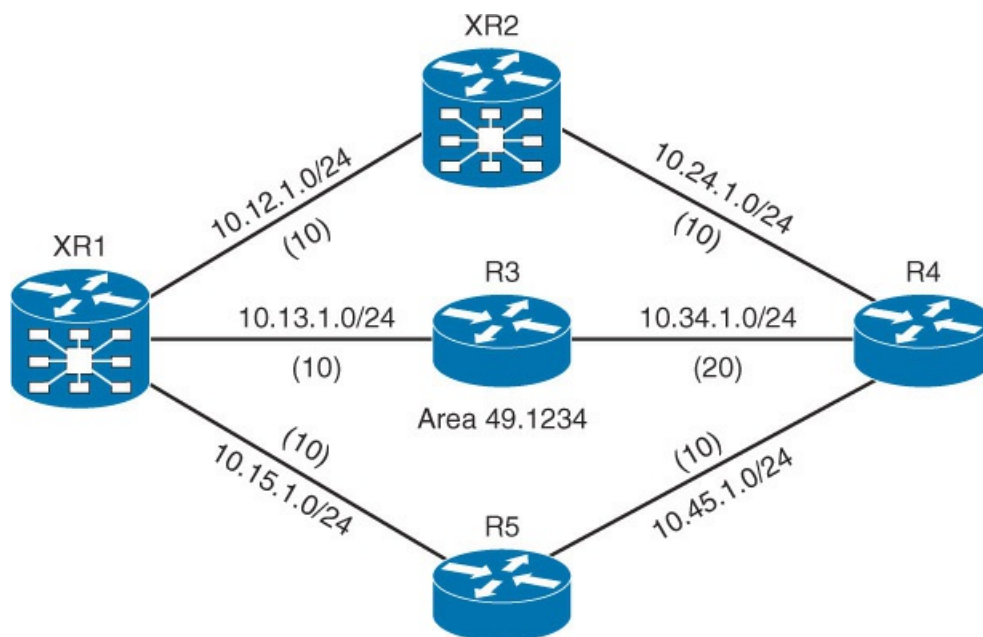


Figure 9-9 IS-IS Overload Bit Topology

Example 9-19 shows that XR1 has two ECMP paths to the 192.168.4.4/32 network through XR2 or R5.

### Example 9-19 XR1 Route Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route isis
```

```
! Transit links have been removed from the output, only Loopback addresses exist
i L1 192.168.2.2/32 [115/20] via 10.12.1.2, 00:25:55, GigabitEthernet0/0/0/0
i L1 192.168.3.3/32 [115/20] via 10.13.1.3, 00:25:56, GigabitEthernet0/0/0/3
i L1 192.168.4.4/32 [115/30] via 10.15.1.5, 00:25:04, GigabitEthernet0/0/0/2
   [115/30] via 10.12.1.2, 00:25:04, GigabitEthernet0/0/0/0
i L1 192.168.5.5/32 [115/20] via 10.15.1.5, 00:25:51, GigabitEthernet0/0/0/2
```

[Example 9-20](#) provides the relevant configuration with the overload bit set on XR2 and R5.

### Example 9-20 XR2 and R5 Overload Bit Configuration

[Click here to view code image](#)

#### XR2

```
router isis CISCO
  set-overload-bit
net 49.1234.0000.0000.0002.00
```

#### R5

```
router isis
net 49.1234.0000.0000.0005.00
  set-overload-bit
```

[Example 9-21](#) shows the IS-IS database with the overload bit set on XR2 and R5.

### Example 9-21 Overload Bit on XR2 and R5

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis database
```

```
! Output omitted for brevity
```

```
IS-IS ISIS (Level-1) Link State Database
```

LSPID	LSP Seq Num	LSP Checksum	LSP Holdtime	ATT/P/OL
XR1.00-00	* 0x00000007	0x71d6	1046	0/0/0
XR2.00-00	0x0000000c	0x2557	1124	0/0/1
R3.00-00	0x00000009	0x5564	1031	0/0/0
R4.00-00	0x0000000c	0x8baa	1065	0/0/0
R5.00-00	0x00000009	0xa406	1155	0/0/1
R5.03-00	0x00000003	0x7ccc	1124	0/0/0

[Example 9-22](#) shows that the XR1 is now routing traffic through R3 where feasible. Notice that the networks 192.168.2.2/32 and 192.168.5.5/32 still route appropriately because they can be reached only through XR2 and R5 accordingly.

### Example 9-22 XR1 Route Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route isis
```

```
! Transit links have been removed from the output, only Loopback addresses exist
i L1 192.168.2.2/32 [115/20] via 10.12.1.2, 00:29:54, GigabitEthernet0/0/0/0
i L1 192.168.3.3/32 [115/20] via 10.13.1.3, 00:29:54, GigabitEthernet0/0/0/3
i L1 192.168.4.4/32 [115/40] via 10.13.1.3, 00:02:01, GigabitEthernet0/0/0/3
i L1 192.168.5.5/32 [115/20] via 10.15.1.5, 00:29:49, GigabitEthernet0/0/0/2
```

## SUMMARIZATION

In addition to shrinking the size of the LSPDB by segmenting the IS-IS routing domain into levels, another method involves summarizing network prefixes. Summarization of network ranges helps the SPF calculations run faster. A router that has 10,000 network entries will take longer to perform a SPF calculation than a router with 500 network entries. Because all routers within a level must maintain an identical copy of the LSPDB, summarization occurs when routers enter an IS-IS level, such as

- L1 routes entering the L2 backbone
- L2 routes leaking into the L1 backbone
- Redistribution of routes into an area

Figure 9-10 illustrates two concepts of summarization within an IS-IS routing domain. XR1 has 15 networks, starting at 172.16.1.0/24 through 172.16.15.0/24, that are advertised in L1 LSPs. XR2 is configured with two summarization ranges of 172.16.0.0/21 and 172.16.8.0/21 that reduce the 15 networks in the L1 LSP to two networks in the L2 LSPs. R4 is leaking all L2 routes into the L1 Area 49.0045. R4 is summarizing the two L2 networks into one L1 leaked network (172.16.0.0/20).

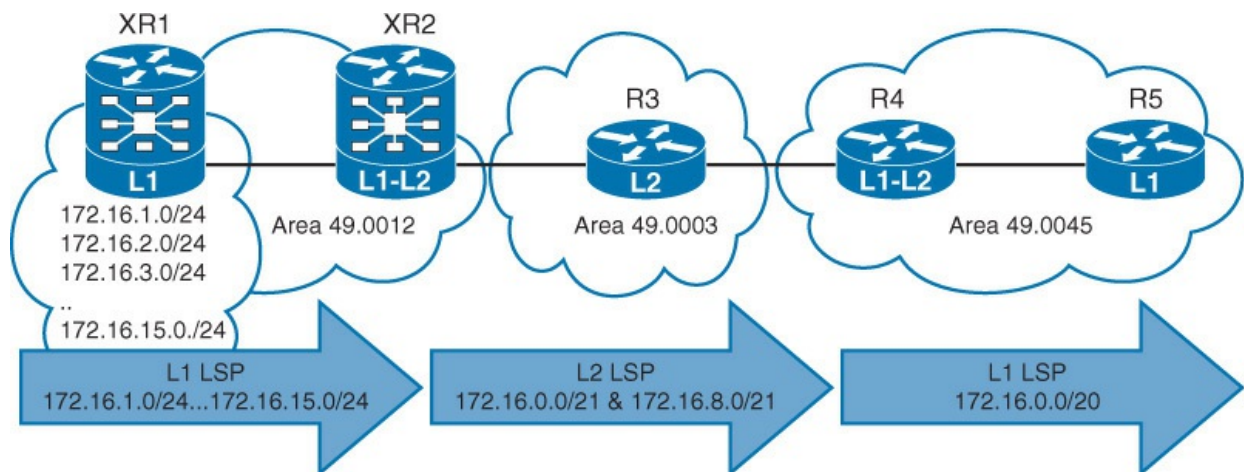


Figure 9-10 IS-IS Summarization

### Note

IS-IS summarization does not occur between areas within the L2 backbone because all routers in the L2 backbone maintain an identical copy of the LSPDB. In Figure 9-10, R3 could not summarize any of the network prefixes.

The default metric for the summary range is the smallest metric associated with any matching network prefix; however, it can be set as part of the configuration.

In Figure 9-11, R1 summarizes the 172.16.1.0/24 and 172.16.3.0/24 network into the 172.16.0.0/16 summary range. The 172.16.1.0/24 network has the lowest path metric, so that summary range uses a metric of 20.

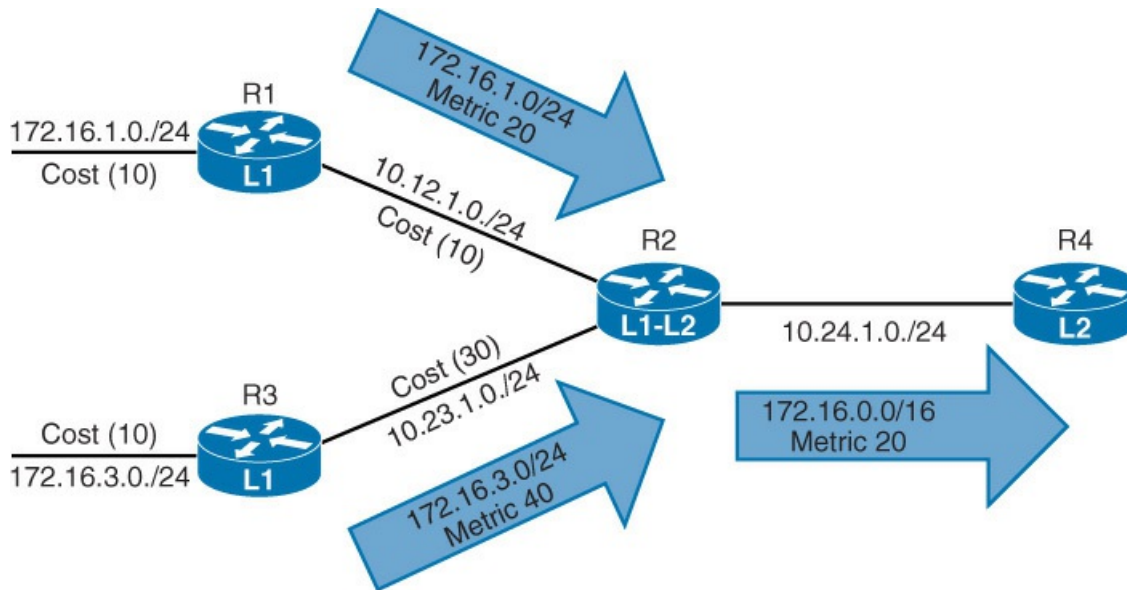


Figure 9-11 IS-IS Summarization Metric

IS-IS behaves identically to OSPF and checks every prefix within the summarization range when a matching prefix is advertised or withdrawn between levels. If a lower metric is available, the summary entry is made, and a new LSP is generated with a new metric; and if the lowest metric is removed, a new metric is identified, and a new LSP is advertised.

The summarization range is defined on IOS nodes with the command **summary-address network subnet-mask [level-1 | level-2 | level-1-2] [metric 1-4294967295]** under the IS-IS process. IOS XR nodes use the command **summary-prefix network/prefix-length [{level {1|2}}]** under the address family within the IS-IS router configuration. The default level of summarization occurs only for routes advertised into L2.

Figure 9-12 will be used to demonstrate summarization within an IS-IS routing domain. XR1 is advertising the 172.16.1.0/24 and 172.16.15.0/24 network, and R5 is redistributing static routes to 172.31.1.0/24 and 172.31.15.0/24. XR2 will use a summary range of 172.16.0.0/16 for routes as they enter the L2 level, and R5 will summarize the redistributed routes into the 172.31.0.0/16 range.

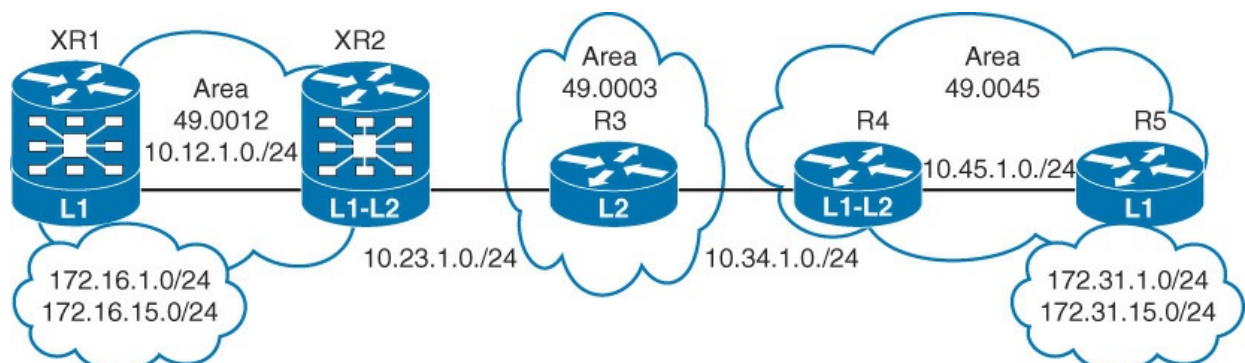


Figure 9-12 IS-IS Summarization Topology

[Example 9-23](#) shows the routing table of XR2 and R4. The networks from XR1 and R5 have been highlighted.

### Example 9-23 XR2 and R4 Routing Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route isis

i L2 10.34.1.0/24 [115/20] via 10.23.1.3, 00:01:39, GigabitEthernet0/0/0/2
i L2 10.45.1.0/24 [115/30] via 10.23.1.3, 00:01:26, GigabitEthernet0/0/0/2
i L1 172.16.1.0/24 [115/20] via 10.12.1.1, 00:00:25, GigabitEthernet0/0/0/0
i L1 172.16.15.0/24 [115/20] via 10.12.1.1, 00:00:25, GigabitEthernet0/0/0/0
i L2 172.31.1.0/24 [115/30] via 10.23.1.3, 00:01:26, GigabitEthernet0/0/0/2
i L2 172.31.15.0/24 [115/30] via 10.23.1.3, 00:01:26, GigabitEthernet0/0/0/2
```

```
R4#show ip route isis
! Output omitted for brevity

i L2 10.12.1.0/24 [115/30] via 10.34.1.3, 00:02:34, GigabitEthernet0/1
i L2 10.23.1.0/24 [115/20] via 10.34.1.3, 00:02:19, GigabitEthernet0/1
i L2 172.16.1.0 [115/40] via 10.34.1.3, 00:01:04, GigabitEthernet0/1
i L2 172.16.15.0 [115/40] via 10.34.1.3, 00:01:04, GigabitEthernet0/1
i L1 172.31.1.0 [115/10] via 10.45.1.5, 00:08:05, GigabitEthernet0/0
i L1 172.31.15.0 [115/10] via 10.45.1.5, 00:08:00, GigabitEthernet0/0
```

[Example 9-24](#) provides the relevant IS-IS router configuration on XR2 and R4 where the L1 routes are advertised into the L2 level.

### Example 9-24 IS-IS Summarization Configuration

[Click here to view code image](#)

```
XR2
router isis CISCO
 net 49.0012.0000.0000.0002.00
 address-family ipv4 unicast
  summary-prefix 172.16.0.0/16
```

```
R4
router isis
 net 49.0045.0000.0000.0004.00
 summary-address 172.31.0.0 255.255.0.0
```

[Example 9-25](#) provides R3's routing table. Notice that the smaller prefixes from XR1 and R5 are not present, but the 172.16.0.0/16 and 172.32.0.0/16 network ranges are present instead.

### Example 9-25 R3 Routing Table

[Click here to view code image](#)

```
R3#show ip route isis
```

```
! Output omitted for brevity
```

```
i L2 10.12.1.0/24 [115/20] via 10.23.1.2, 00:12:08, GigabitEthernet0/0
i L2 10.45.1.0/24 [115/20] via 10.34.1.4, 00:11:40, GigabitEthernet0/1
i L2 172.16.0.0/16 [115/30] via 10.23.1.2, 00:07:36, GigabitEthernet0/0
i L2 172.31.0.0/16 [115/30] via 10.34.1.4, 00:00:11, GigabitEthernet0/1
```

The router performing summarization will install a discard route to Null0 for the summarized network range. Discard routes prevent routing loops where portions of the summarized network range do not have a more specific route in the RIB. This is similar to the static route to Null 0 in Chapter 4. Example 9-26 shows the discard route installed on XR2 and R4.

### Example 9-26 XR2 and R4 Discard Routes

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route isis
```

```
i L2 10.34.1.0/24 [115/20] via 10.23.1.3, 00:14:09, GigabitEthernet0/0/0/2
i L2 10.45.1.0/24 [115/30] via 10.23.1.3, 00:01:01, GigabitEthernet0/0/0/2
i su 172.16.0.0/16 [115/20] via 0.0.0.0, 00:09:53, Null0
i L1 172.16.1.0/24 [115/20] via 10.12.1.1, 00:12:55, GigabitEthernet0/0/0/0
i L1 172.16.15.0/24 [115/20] via 10.12.1.1, 00:12:55, GigabitEthernet0/0/0/0
i L2 172.31.0.0/16 [115/40] via 10.23.1.3, 00:02:28, GigabitEthernet0/0/0/2
```

```
R4#show ip route isis
```

```
! Output omitted for brevity
```

```
i L2 10.12.1.0/24 [115/30] via 10.34.1.3, 00:14:37, GigabitEthernet0/1
i L2 10.23.1.0/24 [115/20] via 10.34.1.3, 00:14:22, GigabitEthernet0/1
i L2 172.16.0.0/16 [115/40] via 10.34.1.3, 00:10:05, GigabitEthernet0/1
i su 172.31.0.0/16 [115/20] via 0.0.0.0, 00:02:42, Null0
i L1 172.31.1.0/24 [115/20] via 10.45.1.5, 00:04:28, GigabitEthernet0/0
i L1 172.31.15.0/24 [115/20] via 10.45.1.5, 00:04:28, GigabitEthernet0/0
```

## DEFAULT ROUTES

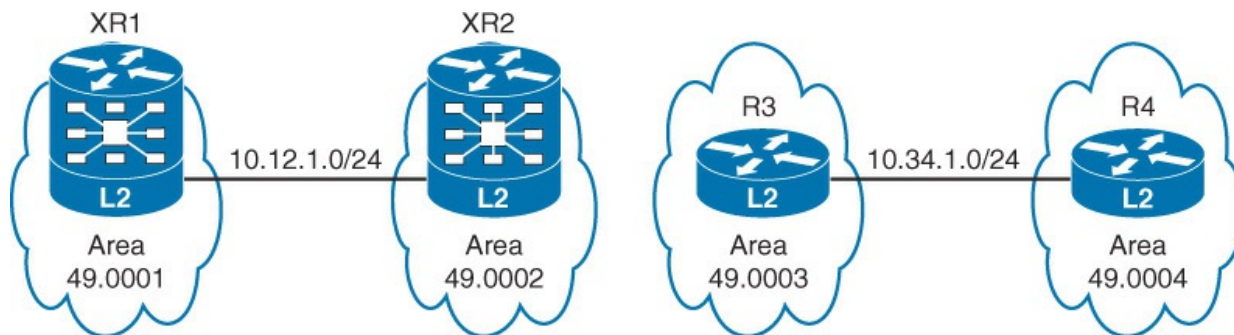
IS-IS uses the attached bit to provide a route of last resort for L1 routers to locate an L1-L2 router attached to the backbone. The command **default-information originate** advertises a default route in the L2 LSPs so that backbone routers can have a route of last resort, too.

The configuration is placed under the IS-IS router configuration on IOS nodes and under the address family within the IS-IS router configuration on IOS XR nodes.

Note

A default route is not required in the RIB to advertise a default route.

In Figure 9-13, two IS-IS routing domains are provided. XR2 and R3 will advertise the default route to their neighboring L2 router.



**Figure 9-13** IS-IS Default Route Advertisement Topology

Example 9-27 provides XR2 and R3's relevant configuration of the default route advertisement.

### Example 9-27 XR2 and R3 Configuration of Default Route Advertisement

[Click here to view code image](#)

#### XR2

```
router isis CISCO
 net 49.0002.0000.0000.0002.00
 address-family ipv4 unicast
 default-information originate
```

#### R3

```
router isis
 net 49.0003.0000.0000.0003.00
 default-information originate
```

Example 9-28 verifies that XR1 and R4 receive the default route from XR2 and R3. Notice that the default route is an L2 route.

### Example 9-28 XR1 Route Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route isis
```

```
i*L2 0.0.0.0/0 [115/10] via 10.12.1.2, 00:03:19, GigabitEthernet0/0/0/0
i L2 10.23.1.0/24 [115/20] via 10.12.1.2, 00:03:19, GigabitEthernet0/0/0/0
i L2 192.168.2.2/32 [115/20] via 10.12.1.2, 00:03:19, GigabitEthernet0/0/0/0
```

```
R4#show ip route isis
```

```
! Output omitted for brevity
```

```
Gateway of last resort is 10.34.1.3 to network 0.0.0.0
```

```
i*L2 0.0.0.0/0 [115/10] via 10.34.1.3, 00:07:29, GigabitEthernet0/1
    192.168.3.0/32 is subnetted, 1 subnets
i L2 192.168.3.3 [115/20] via 10.34.1.3, 00:06:51, GigabitEthernet0/1
```



## PREFIX SUPPRESSION

IS-IS advertises every network prefix in an IS-IS topology into the routing domain. Because the IS-IS protocol operates at the second layer of the OSI model, the transit network prefixes in the IP Reachability TLVs are not needed to build the IS-IS topology. The IS-IS topology is still built with the IS Neighbors TLVs.

Service provider networks typically have 500,000+ routes in their network. Border Gateway Protocol (BGP) is commonly used for advertisement of network segments with a prefix length smaller than /30, allowing for the IGP to only include transit networks (/30 and /31) and loopback interfaces (/32). Unless connectivity to a transit network (/30 and /31) is required, leaving the transit network IP prefixes in the LSPDB can cause SPF calculations to take longer and can consume more router memory than necessary. Removing the transit networks can reduce the number of prefixes significantly while leaving loopback addresses for establishing BGP sessions as demonstrated in the following example.

IS-IS supports prefix suppression to restrict the IP prefixes included in the LSP. IOS nodes can suppress the advertisement of an interface's IP addresses with the interface parameter command **no isis advertise-prefix**. If a router has multiple interfaces, it might be faster to enable the IS-IS router configuration command **advertise-passive only**, which will advertise only the passive interfaces in the LSPs. IOS XR nodes suppress network advertisements with the command **suppressed** under the interface in the IS-IS configuration.

In [Figure 9-14](#), the networks between XR1, XR2, R3, and R4 are transit networks. Only connectivity to the Loopback 0 interfaces is required.

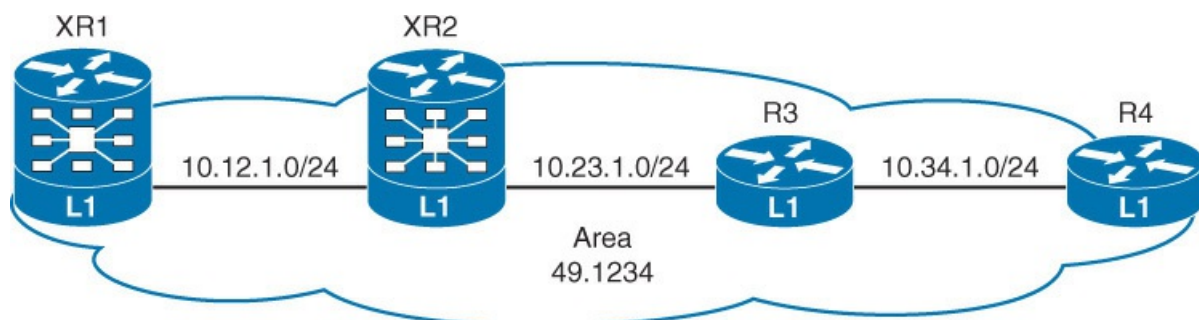


Figure 9-14 IS-IS Prefix Suppression

[Example 9-29](#) provides the routing table for XR1. Notice that the transit networks 10.23.1.0/24 and 10.34.1.0/24 are included.

### Example 9-29 XR1 Route Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route isis
```

```
i L1 10.23.1.0/24 [115/20] via 10.12.1.2, 00:00:23, GigabitEthernet0/0/0/0
i L1 10.34.1.0/24 [115/30] via 10.12.1.2, 00:00:14, GigabitEthernet0/0/0/0
i L1 192.168.2.2/32 [115/20] via 10.12.1.2, 00:00:23, GigabitEthernet0/0/0/0
i L1 192.168.3.3/32 [115/30] via 10.12.1.2, 00:00:14, GigabitEthernet0/0/0/0
i L1 192.168.4.4/32 [115/40] via 10.12.1.2, 00:00:14, GigabitEthernet0/0/0/0
```

Example 9-30 shows the IS-IS database for the Area 49.1234. Notice that six of the IP Reachability entries are for the transit network, compared to the four entries for each router's loopback address.

### **Example 9-30** XR1 Route Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis database detail | i IP
  IP Address: 192.168.1.1
  Metric: 10      IP 10.12.1.0/24
  Metric: 10      IP 192.168.1.1/32
  IP Address: 192.168.2.2
  Metric: 10      IP 10.12.1.0/24
  Metric: 10      IP 10.23.1.0/24
  Metric: 10      IP 192.168.2.2/32
  IP Address: 192.168.3.3
  Metric: 10      IP 192.168.3.3/32
  Metric: 10      IP 10.23.1.0/24
  Metric: 10      IP 10.34.1.0/24
  IP Address: 192.168.4.4
  Metric: 10      IP 192.168.4.4/32
  Metric: 10      IP 10.34.1.0/24
```

Example 9-31 provides the configuration for XR1, XR2, R3, and R4. Notice that R3 has set the Loopback 0 interface as passive and is advertising only the passive interfaces.

### **Example 9-31** XR1, XR2, R3, and R4 Prefix Suppression Configuration

[Click here to view code image](#)

```
XR1
router isis CISCO
 net 49.1234.0000.0000.0001.00
 interface Loopback0
  address-family ipv4 unicast
  !
 interface GigabitEthernet0/0/0/0
  suppressed
  address-family ipv4 unicast
  !
```

**XR2**

```
router isis CISCO
 net 49.1234.0000.0000.0002.00
 interface Loopback0
  address-family ipv4 unicast
  !
  !
 interface GigabitEthernet0/0/0/0
  suppressed
  address-family ipv4 unicast
  !
  !
 interface GigabitEthernet0/0/0/2
  suppressed
  address-family ipv4 unicast
  !
```

**R3**

```
interface Loopback0
 ip address 192.168.3.3 255.255.255.255
 !
interface GigabitEthernet0/0
 ip address 10.23.1.3 255.255.255.0
 ip router isis
 !
interface GigabitEthernet0/1
 ip address 10.34.1.3 255.255.255.0
 ip router isis
 !
router isis
 net 49.1234.0000.0000.0003.00
 advertise passive-only
 passive-interface Loopback0
```

**R4**

```
interface Loopback0
 ip address 192.168.4.4 255.255.255.255
 ip router isis
 !
interface GigabitEthernet0/1
 ip address 10.34.1.4 255.255.255.0
 ip router isis
 no isis advertise prefix

router isis
 net 49.1234.0000.0000.0004.00
```

**Example 9-32** verifies that XR1 no longer sees the 10.23.1.0/24 and 10.34.1.0/24 transit networks in the RIB.

**Example 9-32 XR1 Route Table**

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route isis
```

```
i L1 192.168.2.2/32 [115/20] via 10.12.1.2, 00:00:23, GigabitEthernet0/0/0/0
i L1 192.168.3.3/32 [115/30] via 10.12.1.2, 00:00:14, GigabitEthernet0/0/0/0
i L1 192.168.4.4/32 [115/40] via 10.12.1.2, 00:00:14, GigabitEthernet0/0/0/0
```

[Example 9-33](#) shows the IS-IS database, which does not include the transit networks. IP reachability information was reduced by 60 percent by suppressing the transit network prefixes.

### Example 9-33 XR1 Route Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show isis database detail | i IP
```

```
IP Address: 192.168.1.1
Metric: 10      IP 192.168.1.1/32
IP Address: 192.168.2.2
Metric: 10      IP 192.168.2.2/32
IP Address: 192.168.3.3
Metric: 0       IP 192.168.3.3/32
IP Address: 192.168.4.4
Metric: 10      IP 192.168.4.4/32
```

[Example 9-34](#) demonstrates that R4 can connect to XR1. Notice that the packet is sourced from R4's Loopback 0 interface to ensure that XR1 will be able to reply.

### Example 9-34 XR1 Route Table

[Click here to view code image](#)

```
R4#ping 192.168.1.1 source loopback 0
```

```
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
```

```
Packet sent with a source address of 192.168.4.4
```

```
!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 5/5/6 ms
```

[Example 9-35](#) demonstrates that the transit networks are still visible in the traceroute, although they are not present in the IS-IS LSPDB.

### Example 9-35 XR1 Route Table

[Click here to view code image](#)

```
R4#traceroute 192.168.1.1 source loopback 0
```

```
Type escape sequence to abort.
```

```
Tracing the route to 192.168.1.1
```

```
VRF info: (vrf in name/id, vrf out name/id)
```

```
1 10.34.1.3 0 msec 1 msec 1 msec
```

```
2 10.23.1.2 6 msec 3 msec 3 msec
```

```
3 10.12.1.1 9 msec * 7 msec
```

## SUMMARY

IS-IS uses a two-tier hierarchical architecture, where Level 2 is the backbone area and can cross through multiple areas. L1-L2 routers advertise L1 routes to other L2 routers and provide connectivity to L1 routers by setting the attached bit in the L1 LSP. The L2 backbone should be contiguous to prevent any traffic loss between areas.

Route summarization occurs only when prefixes are advertised into IS-IS levels because the LSPDB is identical between all routers in a level. Summarization reduces the memory consumption for the routing table and provides fault isolation because visibility is removed for prefixes that are more specific. Prefix suppression of transit network prefixes provide an additional method to reduce the size of the LSPDB.

The default IS-IS interface metrics can introduce suboptimal routing. The default narrow metrics does not scale well for topologies that use drastically different bandwidth speeds. Using the IS-IS wide metrics provides scalability when interface speeds can vary drastically (for example, 1 Mbps to 100 Gbps).

## REFERENCES IN THIS CHAPTER

ISO, "Intermediate System to Intermediate System Intra-Domain Routing Information Exchange Protocol for Use in Conjunction with the Protocol for Providing the Connectionless-mode Network Service (ISO 8473)," International Standard 10589, 1992.

Li, Tony, et al., RFC 2966, *Domain-wide Prefix Distribution with Two-Level IS-IS*, IETF, <http://www.ietf.org/rfc/rfc2966.txt>, October 2000.

Li, Tony and Henk Smit. RFC 3784, *Intermediate System to Intermediate System (IS-IS) Extensions for Traffic Engineering (TE)*, IETF, <https://tools.ietf.org/html/rfc3784>, June 2004.

Doyle, Jeff, Carrol, Jennifer. *Routing TCP/IP Volume I, 2nd Edition*. Indianapolis: Cisco Press, 2006.

Martey, Aber. *IS-IS Network Design Solutions*. Indianapolis: Cisco Press, 2002.

Cisco. Cisco IOS Software Configuration Guides. <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides. <http://www.cisco.com>

## Chapter 10. Border Gateway Protocol (BGP)

This chapter covers the following topics:

- BGP fundamentals
- BGP inter-router communication
- Basic BGP configuration
- iBGP and eBGP behaviors
- iBGP scalability issues and solutions
- BGP security

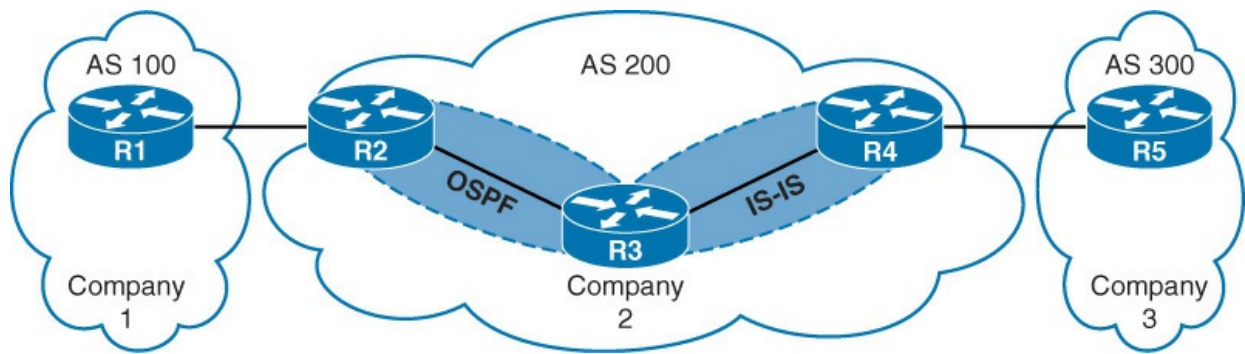
RFC 4271 defines Border Gateway Protocol (BGP) as a standardized path vector routing protocol that provides scalability, flexibility, and network stability. When BGP was created, the primary design consideration was for IPv4 interorganization connectivity on public networks, like the Internet, or private dedicated networks. BGP is the only protocol used to exchange networks on the Internet, which has more than 500,000 IPv4 routes at the time of this writing. BGP does not advertise incremental updates or refresh network advertisements like Open Shortest Path First (OSPF) Protocol or Intermediate System-to-Intermediate System (IS-IS) Protocol because the large size of the BGP tables. BGP prefers stability within the network because a link flap could result in route-computation for thousands of routes.

In interior gateway protocols (IGPs), all routers use a common logic within the routing domain to find the shortest path to reach a destination. Exterior gateway protocols (EGPs) need the flexibility in the routing policy because routing between multiple organizations could require different traffic patterns based on interorganizational agreements.

### **BGP FUNDAMENTALS**

From the perspective of BGP, an autonomous system is a collection of routers under a single organization's control, using one or more IGPs and common metrics to route packets within the autonomous system. If multiple IGPs or metrics are used within an autonomous system, the autonomous system must appear consistent to external autonomous systems in routing policy. An IGP is not required within an autonomous system and could use BGP as the only routing protocol in it, too.

In Figure 10-1, AS100 and AS300 are not aware that AS200 is using OSPF and IS-IS as the IGP because the routing policy appears consistent.



**Figure 10-1** *Autonomous Systems and IGP*

### Autonomous System Numbers

Autonomous system numbers (ASNs) were originally 2 bytes (16 bits), providing 65,535 ASNs. Due to exhaustion, RFC 4893 expanded the ASN field to accommodate 4 bytes (32 bits). This allows for 4,294,967,295 unique ASNs, providing quite a leap from the original 65,535 ASNs.

Two blocks of private ASNs are available for any organization to use as long as they are never exchanged publicly on the Internet. ASNs 64,512 through 65,534 are private ASNs within the 16-bit ASN range, and 4,200,000,000 through 4,294,967,294 are private ASNs within the extended 32-bit range. Private ASNs are similar to RFC 1918 IPv4 address space and are exclusively used for private routing.

The Internet Assigned Numbers Authority (IANA) is responsible for assigning all public ASNs to ensure that they are globally unique. IANA requires the following items when requesting a public ASN:

- Proof of a publicly allocated network range
- Proof that Internet connectivity is provided through multiple connections
- Need for a unique route policy from your providers

#### Note

It is imperative that you use only private ASNs or the ASN assigned by IANA or your service provider. Using another organization's ASN without permission could result in traffic loss, and cause havoc on the Internet.

### Path Attributes

BGP attaches path attributes (PAs) associated with each network path. The path attributes provide BGP with granularity and control of routing policies within BGP. The BGP prefix attributes are classified as follows:

- Well-known mandatory
- Well-known discretionary
- Optional transitive
- Optional nontransitive

Per RFC 4271, well-known attributes must be recognized by all BGP implementations. Well-

known mandatory attributes must be included with every prefix advertisement; whereas well-known discretionary attributes may or may not be included with the prefix advertisement.

Optional attributes do not have to be recognized by all BGP implementations. Optional attributes can be set so that they are transitive and stay with the route advertisement from autonomous system to autonomous system. Other PAs are *nontransitive* and cannot be shared from autonomous system to autonomous system.

In BGP, the network layer reachability information (NLRI) is the routing update that consists of the network prefix, prefix-length, and any BGP path attributes for that specific route.

### Loop Prevention

BGP is a path vector routing protocol and does not contain a complete topology of the network (like link-state routing protocols do). BGP behaves similar to distance vector protocols to ensure a path is loop-free path.

The BGP attribute `AS_PATH`, typically written as `AS_Path`, is a well-known mandatory attribute and includes a complete listing of all the ASNs that the prefix advertisement has traversed from its source autonomous system. The `AS_Path` is used as a loop-prevention mechanism in the BGP protocol. If a BGP router receives a prefix advertisement with its autonomous system listed in the `AS_Path`, it discards the prefix because the router thinks the advertisement forms a loop.

Figure 10-2 demonstrates the loop-prevention mechanism:

- AS100 advertises the 172.16.1.0/24 prefix to AS200.
- AS200 then advertises the prefix to AS400, which then advertises the prefix to AS300.
- AS300 advertises the prefix back to AS100 with an `AS_Path` of 300 400 200 100. AS100 sets itself in the `AS_Path` and discards the prefix.

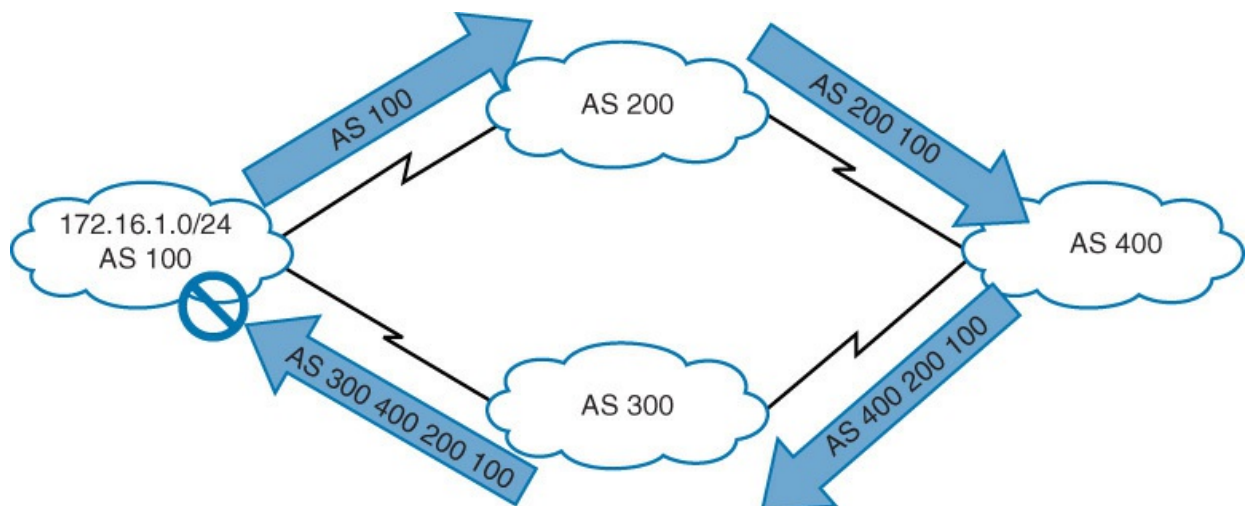


Figure 10-2 Path Vector Loop Prevention



## Address Families

RFC 2858 added multiprotocol BGP (MBGP) capability by adding extensions called *address family identifiers* (AFIs). An address family correlates to a specific network protocol such as IPv4, IPv6, VPN, and so on; and additional granularity is provided by a subsequent address family identifier (SAFI), such as unicast, labeled unicast, and multicast. BGP includes an AFI and SAFI with every route advertisement to differentiate between the protocols.

Every address family maintains a separate database and configuration in BGP. This allows for a routing policy in one address family to differ from a routing policy in a different address family even though the router uses the same BGP session to the other router.

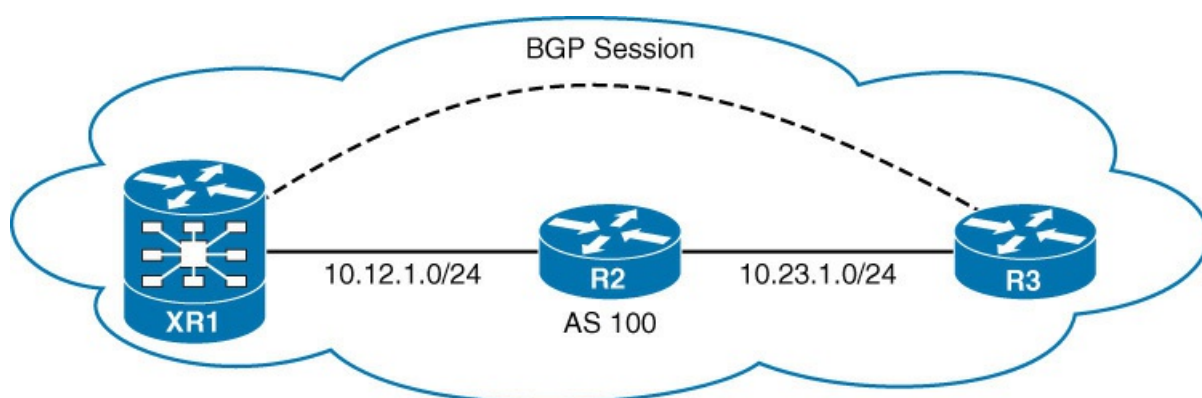
## INTER-ROUTER COMMUNICATION

BGP does not use hello packets to discover neighbors like IGP protocols do, and it does not discover neighbors dynamically. BGP was designed as an interautonomous routing protocol, implying that neighbor adjacencies should not change frequently and are coordinated. BGP neighbors are defined by IP address.

BGP uses TCP port 179 to communicate with other routers. TCP allows for handling of fragmentation, sequencing, and reliability (acknowledgment and retransmission) of communication packets.

IGP protocols follow the physical topology because the sessions are formed with hellos that cannot cross network boundaries. BGP uses TCP, which is capable of crossing network boundaries. Although BGP can form neighbor adjacencies that are directly connected, it can also form adjacencies that are multiple hops away. Multihop sessions require that the router use an underlying route installed in the Routing Information Base (RIB) (static or from any routing protocol) to establish the TCP session with the remote endpoint.

In [Figure 10-3](#), XR1 is able to establish a BGP session with R3 even though it passes through R2. XR1 uses a static route to reach the 10.23.1.0/24 network, and R3 has a static route to reach the 10.12.1.0/24 network. R2 is unaware that XR1 and R3 have established a BGP session even though the packets flow through R2.



**Figure 10-3** BGP Multihop Sessions

### Note

BGP neighbors connected via the same network use the Address Resolution Protocol (ARP) table to locate the IP address of the peer. Multihop BGP sessions require route table information to determine reachability of the peer's IP address. It is common to have a static route or IGP running between BGP neighbors for providing the topology path information for establishing the BGP TCP session.

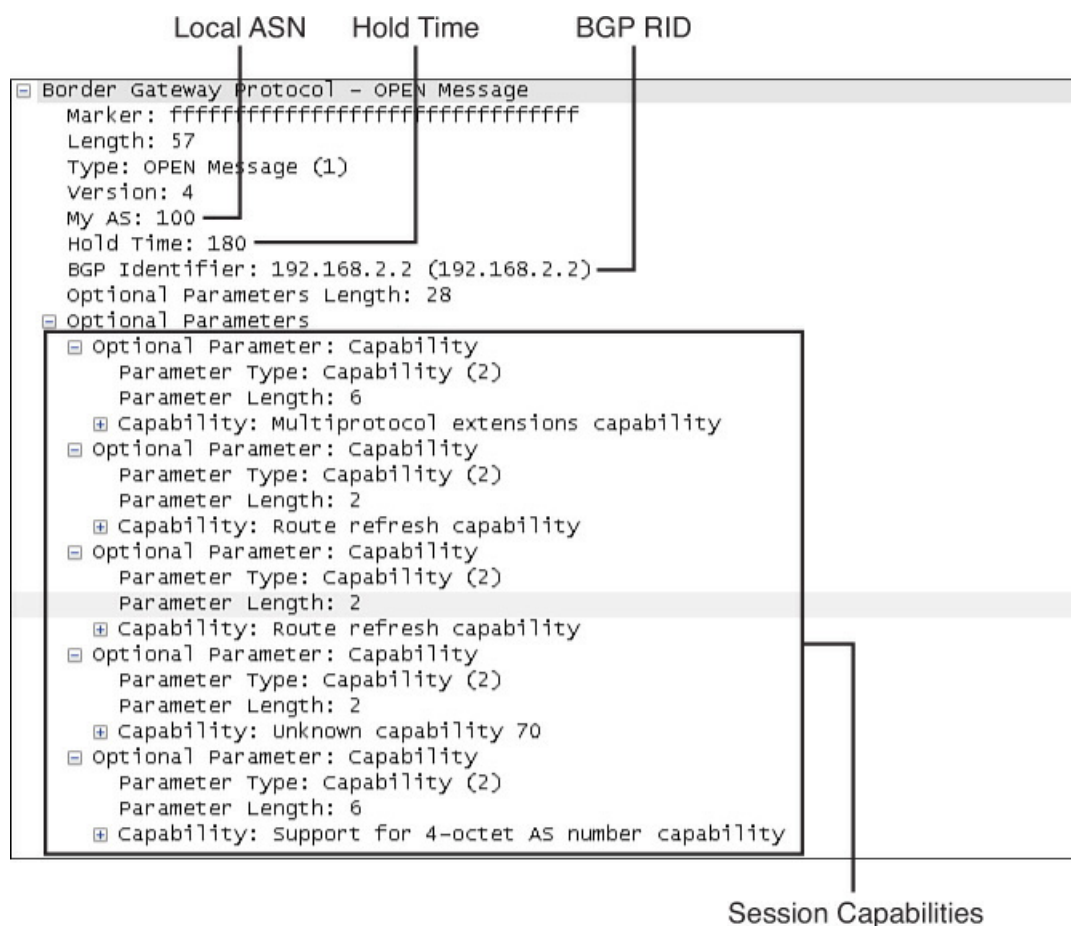
BGP communication uses four message types, as shown in [Table 10-1](#).

Type	Name	Functional Overview
1	Open	Sets up and establishes BGP adjacency
2	Update	Advertises, updates, or withdraws routes
3	Notification	Indicates an error condition to a BGP neighbor
4	Keepalive	Ensures that BGP neighbors are still alive

**Table 10-1** BGP Packet Types

### Open Messages

The open message is used to establish a BGP adjacency. Both sides negotiate session capabilities before a BGP peering establishes. The open message contains the BGP version number, ASN of the originating router, hold time, BGP identifier, and other optional parameters that establish the session capabilities. [Figure 10-4](#) displays a packet capture of an open message with the fields mentioned earlier highlighted.



**Figure 10-4** BGP Packet Capture of Open Message

### Hold Time

The holdtime attribute sets the hold timer in seconds for each BGP neighbor. Upon receipt of an update or keepalive, the hold timer resets to the initial value. If the hold timer reaches 0, the BGP session is torn down, routes from that neighbor are removed, and an appropriate update route withdraw message is sent to other BGP neighbors for the impacted prefixes. The hold time is a heartbeat mechanism for BGP neighbors to ensure that the neighbor is healthy and alive.

When establishing a BGP session, the routers use the smaller hold time value contained in the two routers' open messages. The hold time value must be at least 3 seconds, or 0. For Cisco

routers, the default hold timer is 180 seconds.

### **BGP Identifier**

The BGP router ID (RID) is a 32-bit unique number that identifies the BGP router in the advertised prefixes as the BGP identifier. The RID can be used as a loop-prevention mechanism for routers advertised within an autonomous system. The RID can be set manually or dynamically for BGP. A non-0 value must be set for routers to become neighbors. The dynamic RID allocation logic varies from IOS to IOS XR.

- **IOS:** IOS nodes use the highest IP address of the any up loopback interfaces. If there is not an up loopback interface, the highest IP address of any active up interfaces becomes the RID when the BGP process initializes.

- **IOS XR:** IOS XR nodes use the IP address of the lowest up loopback interface. If there are no up loopback interfaces, a value of 0 (0.0.0.0) is used and will prevent any BGP adjacencies from forming.

Router IDs typically represent an IPv4 address that resides on the router, such as a loopback address. Any IPv4 address can be used, including IP addresses not configured on the router. For IOS and IOS XR, the command **bgp router-id** *router-id* under BGP router configuration mode statically assigns the BGP RID under the BGP router configuration. Upon changing the router ID, all BGP sessions will reset and need to reestablish.

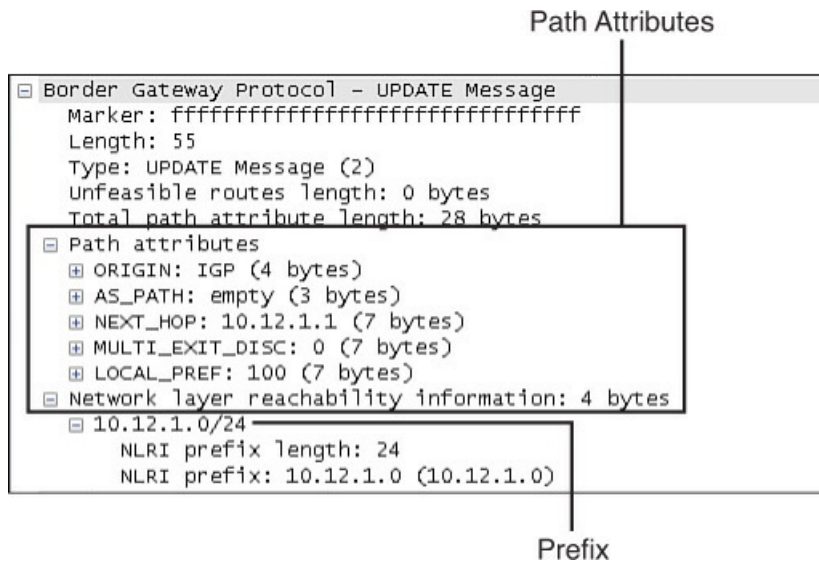
### **Keepalive Messages**

BGP does not rely on the TCP connection state to ensure that the neighbors are still alive. Keepalive messages are exchanged every one-third of the hold timer that is agreed upon between the two BGP routers. Cisco devices have a default hold time of 180 seconds, so the default keepalive interval is 60 seconds. If the hold time is set for 0, no keep alive messages are sent between the BGP neighbors.

### **Update Messages**

The update message advertises any feasible routes, withdraws previously advertised routes, or can do both. The update message includes the network layer reachability information (NLRI) that includes the prefix and associated BGP path attributes when advertising prefixes. Withdrawn NLRIs include only the prefix.

An update message can act as a keepalive to reduce unnecessary traffic. [Figure 10-5](#) displays a packet capture of an update message.



**Figure 10-5** Packet Capture of BGP Update

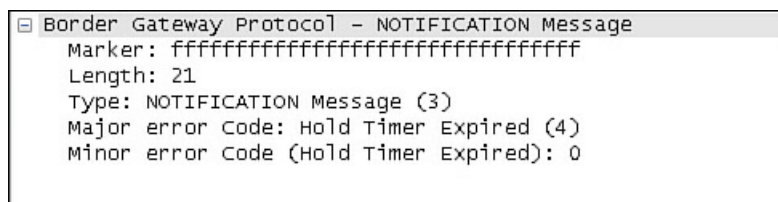
**Note**

BGP tries to use the largest size packet to improve network convergence time. The update may include NLR's prefixes in the message. BGP relies on the TCP maximum segment size (MSS) to define the maximum size of the data payload, and normally bases the calculation on the outbound interface maximum transmission unit (MTU). Multihop BGP sessions may encounter problems if a smaller MTU link is present in the path. BGP includes a MTU path discovery mechanism to detect this issue but might encounter problems with asymmetric routing or firewalls.

The MSS can be set to a value within tolerance of a transit router with the global configuration command `ip tcp mss 500-1460` on IOS nodes, and with the equivalent command `tcp mss 68-10000` for IOS XR routers.

**Notification Messages**

A notification message is sent when an error is detected with the BGP session such as a hold timer expiring, neighbor capabilities changing, or a BGP session reset is requested. This will cause the BGP connection to close. [Figure 10-6](#) shows a BGP notification message for a holder timer expiration.



**Figure 10-6** Packet Capture of BGP Notification

**BGP Sessions**

A BGP session refers to the established adjacency between two BGP routers. BGP sessions are always point-to-point (P2P) and are categorized into two types:

- **Internal BGP (iBGP):** Sessions established with an iBGP router that are in the same autonomous system or participates in the same BGP confederation. iBGP sessions are assumed to be more secure and some of BGP's security measures are lowered in comparison to eBGP sessions. iBGP prefixes are assigned an AD of 200 upon installing into the router's RIB.
- **External BGP (eBGP):** Sessions established with a BGP router that are in a different autonomous system. eBGP prefixes are assigned an administrative distance (AD) of 20 upon installing into the router's RIB.

## BGP NEIGHBOR STATES

BGP forms a TCP session with neighbor routers called peers. BGP uses a process called the Finite State Machine (FSM) to maintain a table of all BGP peers and their operational status. The session may report in the following state:

- Idle
- Connect
- Active
- OpenSent
- OpenConfirm
- Established

### Idle State

This is the first state of the BGP FSM. BGP detects a start event and tries to initiate a TCP connection to the BGP peer and listens for a new connect from a peer router.

If an error causes BGP to go back to the idle state for a second time, the `ConnectRetryTimer` is set to 60 seconds and must decrement to 0 before the connection can be initiated again. Further failures to leave the idle state result in the `ConnectRetryTimer` doubling in length from the previous time.

### Connect State

In this state, BGP initiates the TCP connection. If the three-way TCP handshake completes, the established BGP session BGP process resets the `ConnectRetryTimer` and sends the open message to the neighbor and changes to the `OpenSent` state.

If the `ConnectRetry` timer depletes before this stage is complete, a new TCP connection is attempted, the `ConnectRetry` timer is reset, and the state is moved to active. If any other input is received, the state is changed to idle.

During this stage, the neighbor with the higher IP address manages the connection. The router initiating the request uses a dynamic source port, but the destination port is always 179. [Example 10-1](#) shows an established BGP session using the command `show tcp brief` to display the active TCP sessions between a router

### Example 10-1 Established BGP Session

[Click here to view code image](#)

```
RP/0/0/CPU0:R1#show tcp brief | exc "LISTEN|CLOSED"
```

PCB	VRF-ID	Recv-Q	Send-Q	Local Address	Foreign Address	State
0x088bcbb8	0x60000000	0	0	10.12.1.1:179	10.12.1.2:59884	ESTAB

```
R2#show tcp brief
```

TCB	Local Address	Foreign Address	(state)
EF153B88	10.12.1.2.59884	10.12.1.1.179	ESTAB

Note

Most service providers typically assign their customers the higher IP address because this helps with troubleshooting issues caused by access control lists (ACLs) or firewall rules.

### Active State

In this state, BGP starts a new three-way TCP handshake. If a connection is established, an open message is sent, the hold timer is set to 4 minutes, and the state moves to OpenSent. If this attempt for TCP connection fails, the state moves back to the connect state and resets the ConnectRetryTimer.

### OpenSent State

In this state, an open message has been sent from the originating router and is awaiting an open message from the other router. Once the originating router receives the open message from the other router, both open messages are checked for errors. The following items are compared:

- BGP versions must match.
- The source IP address of the open message must match IP address that is configured for the neighbor.
- ASN in the open message must match what is configured for the neighbor.
- BGP identifiers (RID) must be unique. If a RID does not exist, this condition is not met.
- Security parameters (password, TTL, and so on).

If the open messages do not have any errors, the hold time is negotiated (using the lower value), and a keepalive message is sent (assuming the value is not set to 0). The connection state is then moved to OpenConfirm. If an error is found in the open message, a notification message is sent, and the state is moved back to idle.

If TCP receives a disconnect message, BGP closes the connection, resets the ConnectRetryTimer, and sets the state to active. Any other input in this process results in the state moving to idle.

### OpenConfirm State

In this state, BGP waits for a keepalive or notification message. Upon receipt of a neighbor's keepalive, the state is moved to established. If the hold timer expires, a stop event occurs, or a notification message is received, the state is moved to idle.

### Established State

In this state, the BGP session is established. BGP neighbors exchange routes via update messages. As update and keepalive messages are received, the hold timer is reset. If the hold timer expires, an error is detected BGP moves the neighbor back to the idle state.

Figure 10-7 illustrates the BGP neighbor adjacency process with the corresponding FSM state.

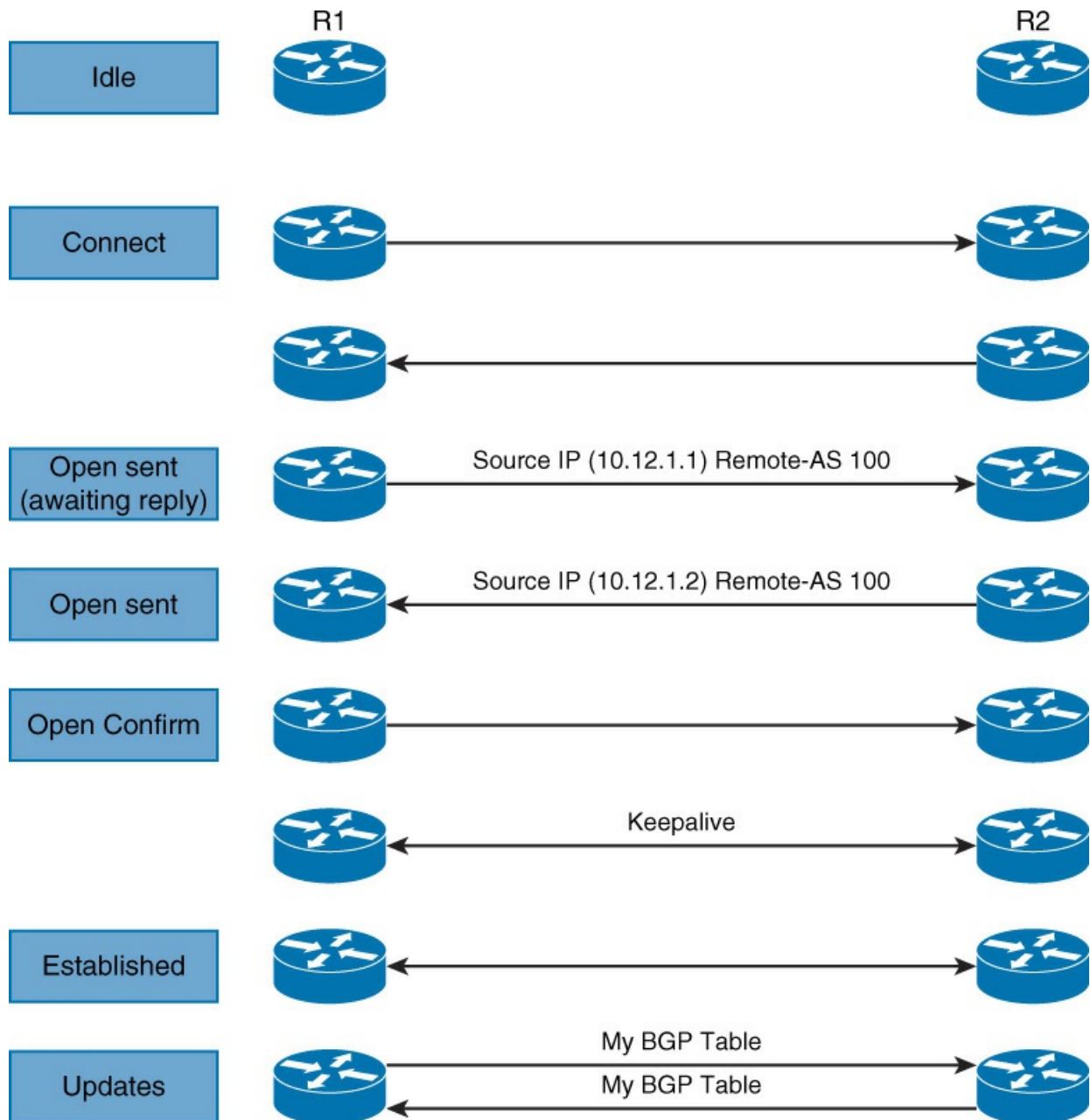


Figure 10-7 BGP Neighbor Adjacency

## BASIC BGP CONFIGURATION

When configuring BGP, it is best to think of the configuration from a modular perspective. BGP router configuration requires the following components:

- BGP session parameters:** BGP session parameters provide settings that involve establishing communication to the remote BGP neighbor. Session settings include the ASN of the BGP peer, authentication, and keepalive timers.
- Address family initialization:** The address family is initialized under the BGP router configuration mode. Networks advertisement and summarization occur within the address family.
- Activate the address family on the BGP peer:** Activate the address family on the BGP peer. For a session to initiate, one address family for that neighbor must be activated. The router's IP address is added to the neighbor table, and BGP will attempt to establish a BGP

session or will accept a BGP session initiated from the peer router.

## IOS

The steps for configuring BGP on an IOS router follow. IOS activates the IPv4 address family by default. This can simplify the configuration in an IPv4 environment because Steps 3 and 4 are optional but may cause confusion when working with other address families. The BGP router configuration command **no bgp default ip4-unicast** disables the automatic activation of the IPv4 AFI so that Steps 3 and 4 are required.

### Step 1. Create the BGP routing process.

Initialize the BGP process with the global command **router bgp as-number**.

### Step 2. Identify the BGP neighbor's IP address and ASN.

Identify the BGP neighbor's IP address and autonomous system number with the BGP router configuration command **neighbor ip-address remote-as as-number**.

### Step 3. Initialize the address family.

Initialize the address family with the BGP router configuration command **address-family address-family address-family-modifier**.

### Step 4. Activate the address family for the BGP neighbor.

Activate the address family for the BGP neighbor with the BGP address family configuration command **neighbor ip-address activate**.

#### Note

On IOS routers, the default address family modifier for the IPv4 and IPv6 address families is unicast and is optional. The address family modifier is required on IOS XR nodes.

Figure 10-8 illustrates a topology for a simple BGP configuration.

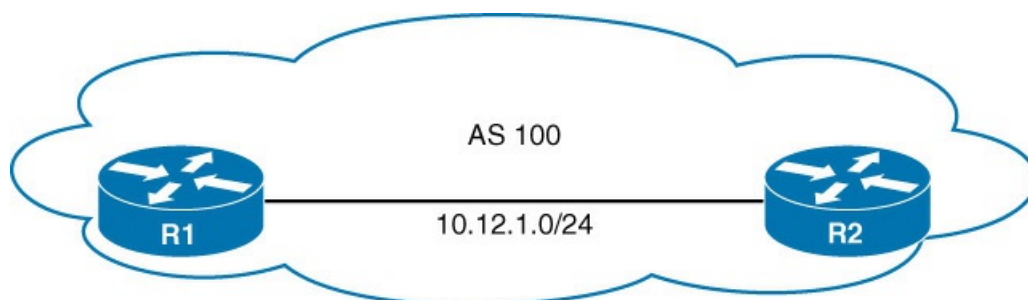


Figure 10-8 Simple BGP Topology

Example 10-2 demonstrates how to configure R1 and R2 using the IOS default and optional IPv4 AFI modifier command-line interface (CLI) syntax. R1 is configured using the default IPv4 address family enabled, and R2 disables IOS's default IPv4 address family and manually activates it for the specific neighbor 10.12.1.1.

### Example 10-2 IOS Basic BGP Configuration



[Click here to view code image](#)

#### R1 (Default IPv4 Address-Family Enabled)

```
router bgp 100
 neighbor 10.12.1.2 remote-as 100
```

#### R2 (Default IPv4 Address-Family Disabled)

```
router bgp 100
 no bgp default ipv4-unicast
 neighbor 10.12.1.1 remote-as 100
 !
 address-family ipv4
  neighbor 10.12.1.1 activate
 exit-address-family
```

## IOS XR

IOS XR's BGP configuration structure is based on the neighbor's peering IP address. All the session and address family configuration is nested under each neighbor. The steps for configuring BGP on an IOS XR router follow:

### Step 1. Create the BGP routing process.

Initialize the BGP process with the global configuration command **router bgp** *as-number*.

### Step 2. Initialize the address family.

Initialize the address-family with the BGP router configuration command **address-family** *address-family address-family-modifier* so that it can be associated to a BGP neighbor.

### Step 3. Identify the BGP neighbor's IP address.

Identify the BGP neighbor's IP address with the BGP router configuration command **neighbor** *ip-address*.

### Step 4. Identify the BGP neighbor's ASN.

Identify the BGP neighbor's ASN with the BGP neighbor configuration command **remote-as** *as-number*.

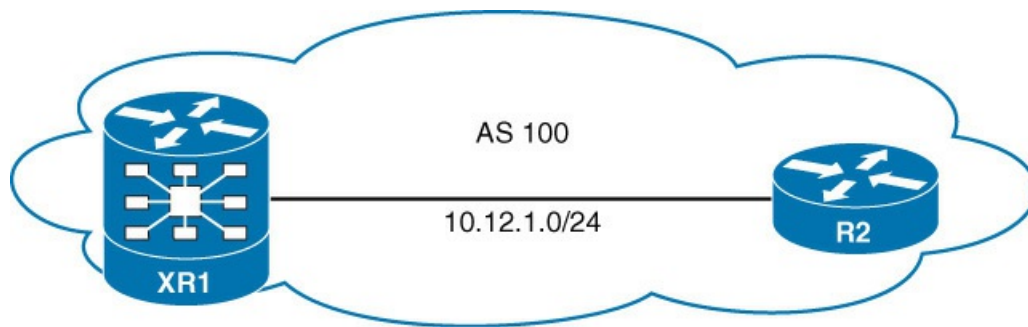
### Step 5. Activate the address family for the BGP neighbor.

Activate the address family for the BGP neighbor with the BGP neighbor configuration command **address-family** *address-family address-family-modifier*.

#### Note

IOS XR nodes will not establish a BGP session if the RID is set to 0 because dynamic RID allocation did not find any up loopback interfaces. The RID will need to be set manually with the BGP router configuration command **bgp router-id** *router-id*.

Figure 10-9 illustrates a topology example with XR1 establishing a BGP session with R2.



**Figure 10-9** IOS and IOS XR Topology Example

**Example 10-3** demonstrates the BGP configuration of XR1 and R2. The RID is set on XR1 because that router does not have any loopback interfaces.

### **Example 10-3** IOS XR and IOS BGP Configuration

[Click here to view code image](#)

#### **XR1**

```
router bgp 100
  bgp router-id 192.168.1.1
  address-family ipv4 unicast
  !
  neighbor 10.12.1.2
    remote-as 100
    address-family ipv4 unicast
  !
```

#### **R2**

```
router bgp 100
  bgp log-neighbor-changes
  no bgp default ipv4-unicast
  neighbor 10.12.1.1 remote-as 100
  !
  address-family ipv4
    neighbor 10.12.1.1 activate
  exit-address-family
```

### **Verification of BGP Sessions**

The BGP session is verified with the command **show bgp address-family address-family-modifier summary** on IOS and IOS XR nodes.

**Example 10-4** displays the IPv4 BGP unicast summary. Notice that the BGP RID and table version are the first components shown. The *Up/Down* column reflects that the BGP session is up for over 5 minutes.

### **Example 10-4** BGP IPv4 Session Summary Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast summary
```

```
! Output omitted for brevity
```

```
BGP router identifier 192.168.1.1, local AS number 100
```

```
BGP main routing table version 4
```

Process	RcvTblVer	bRIB/RIB	LabelVer	ImportVer	SendTblVer	StandbyVer
Speaker	4	4	4	4	4	4

Neighbor	Spk	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	St/PfxRcd
10.12.1.2	0	100	8	7	4	0	0	00:05:23	0

```
R2#show bgp ipv4 unicast summary
```

```
! Output omitted for brevity
```

```
BGP router identifier 192.168.2.2, local AS number 100
```

```
BGP table version is 1, main routing table version 1
```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
10.12.1.1	4	100	8	9	1	0	0	00:05:23	0

Table 10-2 explains the fields of output when displaying the BGP table.

Field	Description
Table Version	This tracks the local BGP table version. Any time the BGP best path executes, the table version increments.
Neighbor	IP address of the BGP peer.
V	BGP version spoken by BGP peer (IOS only).
Spk	Speaker process for that neighbor (always 0) (IOS XR only).
AS	Autonomous system number of BGP peer.
MsgRcvd	Count of messages received from the BGP peer.
MsgSent	Count of messages sent to the BGP peer.
TblVer	Last version of the local router's BGP database that is advertised to that specific peer.
InQ	Number of messages queued to be processed from the peer.
OutQ	Number of messages queued to be sent to the peer.
Up/Down	Length of time the BGP session is established, or the current status if the session is not in established state.
State/PfxRcd	Current state of BGP peer or the number of prefixes received from the peer.

Table 10-2 BGP Summary Fields

Note

Other commands, such as `show bgp summary` and `show ip bgp summary`, came out before MBGP and do not provide a structure for the current multiprotocol capabilities within BGP.

BGP neighbor session state, timers, and other essential peering information are shown with the command **show bgp address-family address-family-modifier neighbors ip-address**, as demonstrated in [Example 10-5](#).

### Example 10-5 BGP IPv4 Neighbor Output

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast neighbors 10.12.1.1
! Output omitted for brevity

! The first section provides the neighbor's IP address, remote-as, indicates if
! the neighbor is 'internal' or 'external', the neighbor's BGP version, RID,
! session state, and timers.
BGP neighbor is 10.12.1.1, remote AS 100, internal link
  BGP version 4, remote router ID 192.168.1.1
  BGP state = Established, up for 00:01:04
  Last read 00:00:10, last write 00:00:09, hold time is 180, keepalive interval is
60 seconds
  Neighbor sessions:
    1 active, is not multisession capable (disabled)
! This second section indicates the capabilities of the BGP neighbor and
! address-families configured on the neighbor.
Neighbor capabilities:
  Route refresh: advertised and received(new)
  Four-octets ASN Capability: advertised and received
  Address family IPv4 Unicast: advertised and received
  Enhanced Refresh Capability: advertised
  Multisession Capability:
  Stateful switchover support enabled: NO for session 1
Message statistics:
  InQ depth is 0
  OutQ depth is 0

! This section provides a list of the BGP packet types that have been received
! or sent to the neighbor router.
Sent      Rcvd
Opens:           1          1
  Notifications:      0          0
  Updates:             0          0
  Keepalives:         2          2
  Route Refresh:      0          0
  Total:              4          3
Default minimum time between advertisement runs is 0 seconds

! This section provides the BGP table version of the IPv4 Unicast address-family.
! The table version is not a 1-to-1 correlation with routes as multiple route
! change can occur during a revision change. Notice the Prefix Activity columns
! in this section.
For address family: IPv4 Unicast
  Session: 10.12.1.1
  BGP table version 1, neighbor version 1/0
  Output queue size : 0
  Index 1, Advertise bit 0

Prefix activity:
  Prefixes Current:      0          0
```

```

Prefixes Total:          0          0
Implicit Withdraw:      0          0
Explicit Withdraw:     0          0
Used as bestpath:      n/a        0
Used as multipath:     n/a        0

```

```

                                Outbound  Inbound
Local Policy Denied Prefixes:  -----  -----
Total:                          0          0
Number of NLRI in the update sent: max 0, min 0

```

```

! This section indicates that a valid route exists in the RIB to the BGP peer IP
! address, provides the number of times that the connection has established and
! dropped, time since the last reset, the reason for the reset, if
! path-mtu-discovery is enabled, and ports used for the BGP session.
Address tracking is enabled, the RIB does have a route to 10.12.1.1
Connections established 2; dropped 1
Last reset 00:01:40, due to Peer closed the session
Transport(tcp) path-mtu-discovery is enabled
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Minimum incoming TTL 0, Outgoing TTL 255
Local host: 10.12.1.2, Local port: 179
Foreign host: 10.12.1.1, Foreign port: 56824

```

## Prefix Advertisement

BGP **network** statements do not enable BGP for a specific interface; instead, they identify a specific network prefix to be installed into the BGP table, known as the *Loc-RIB table*.

After configuring a BGP **network** statement, the BGP process searches the global RIB for an exact network prefix match. The network prefix can be a connected network, secondary connected network, or any route from a routing protocol. After verifying that the **network** statement matches a prefix in the global RIB, the prefix installs into the BGP Loc-RIB table. As the BGP prefix installs into the Loc-RIB, the following BGP PAs are set depending on the RIB prefix type:

- Connected network
  - The next-hop BGP attribute is set to 0.0.0.0.
  - The BGP origin attribute is set to *i* (IGP).
  - BGP weight is set to 32,768.
- Static route or routing protocol
  - The next-hop BGP attribute is set to the next-hop IP address in the RIB.
  - The BGP origin attribute is set to *i* (IGP).
  - BGP weight is set to 32,768.
  - MED is set to the IGP metric.

Not every route in the Loc-RIB is advertised to a BGP peer. All routes in the Loc-RIB follow this process for advertisement to BGP peers:

### Step 1. Pass a validity check:

Verify that the NLRI is valid, and that the next-hop address is resolvable in the global RIB. If the NLRI fails either check, the NLRI remains but does not process further.

### Step 2. Process outbound neighbor route policies:

Process through any specific outbound neighbor policies. After processing, if the route was not denied by the outbound policies, the route is maintained in the Adj-RIB-Out table for later reference.

### Step 3. Advertise the NLRI to BGP peers:

Advertise the NLRI to BGP peers. If the NLRI's next-hop BGP PA is 0.0.0.0, the next-hop address is changed to the IP address of the BGP session.

Figure 10-10 illustrates the concept of installing the network prefix from localized BGP network advertisements to the BGP table.

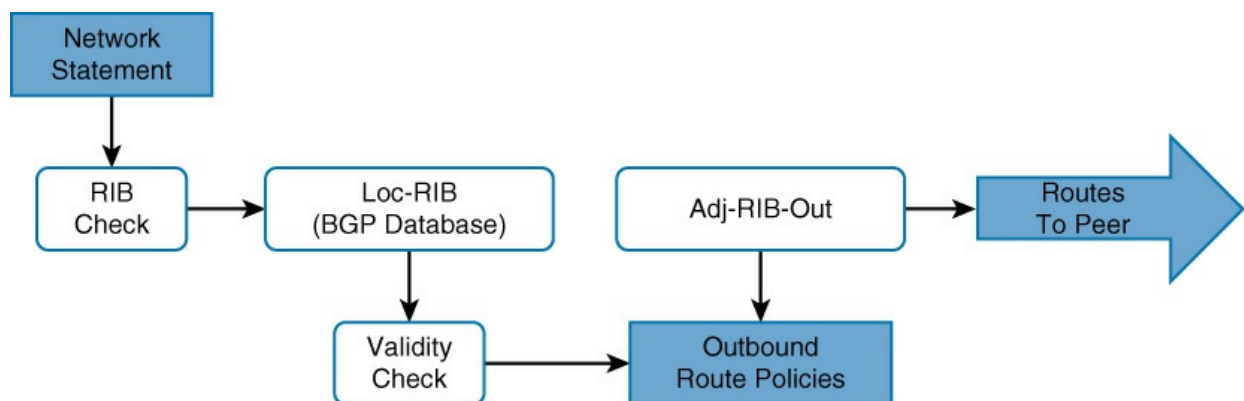


Figure 10-10 BGP Database Processing of Local Route Advertisements

#### Note

BGP advertises only the best-path to other BGP peers regardless of the number of routes (NLRI) in the BGP Loc-RIB.

The **network** statement resides under the appropriate address family within the BGP router configuration. The command **network network mask subnet-mask [route-map route-map-name]** is used for advertising IPv4 networks on IOS routers, and the command **network network/prefix-length [route-policy route-policy-name]** is used for IOS XR routers. The optional **route-map** or **route-policy** parameter provides a method to set specific BGP PAs when the prefix installs into the Loc-RIB.

Figure 10-9 illustrated XR1 and R2 connected via the 10.12.1.0/24 network. Both routers will advertise the Loopback 0 interfaces (192.168.1.1/32 and 192.168.2.2/32 respectively) and the 10.12.1.0/24 network into BGP.

Example 10-6 demonstrates the BGP network advertisement configuration.

### Example 10-6 BGP Network Advertisement

[Click here to view code image](#)

```
XR1  
router bgp 100  
  bgp router-id 192.168.1.1  
  address-family ipv4 unicast  
    network 10.12.1.0/24  
    network 192.168.1.1/32  
  !  
  neighbor 10.12.1.2  
    remote-as 100  
    address-family ipv4 unicast
```

```
R2  
router bgp 100  
  bgp log-neighbor-changes  
  no bgp default ipv4-unicast  
  neighbor 10.12.1.1 remote-as 100  
  !  
  address-family ipv4  
    network 10.12.1.0 mask 255.255.255.0  
    network 192.168.2.2 mask 255.255.255.255  
  neighbor 10.12.1.1 activate  
  exit-address-family
```

#### Receiving and Viewing Routes

Not every prefix in the Loc-RIB is advertised to a BGP peer or installed into the global RIB when received from a BGP peer. BGP processes received route advertisements in the following steps:

##### **Step 1. Store the route in Adj-RIB-In and process inbound route policies.**

The route is stored in the Adj-RIB-In table in original state. The inbound route policy is applied based on the neighbor the route was received.

##### **Step 2. Update the Loc-RIB.**

The BGP Loc-RIB database is updated with the latest entry. The Adj-RIB-In is cleared to save memory.

##### **Step 3. Pass a validity check.**

Verify that the route is valid and that the next-hop address is resolvable in the global RIB. If the route fails, the route remains in the Loc-RIB table but does not process further.

##### **Step 4. Compute the BGP best path.**

Identify the BGP best path and pass only the best path and its path attributes to Step 5. The BGP

best path selection process is explained in Chapter 15, “BGP Best Path Selection.”

### Step 5. Install the BGP best path into global RIB and advertise to peers.

Install the route into the global RIB, and process outbound route policy, store the nondiscarded routes in the Adj-RIB-Out, and advertise to BGP peers.

Figure 10-11 shows the complete BGP route processing logic. It now includes the receipt of a route from a BGP peers and the BGP best path algorithm.

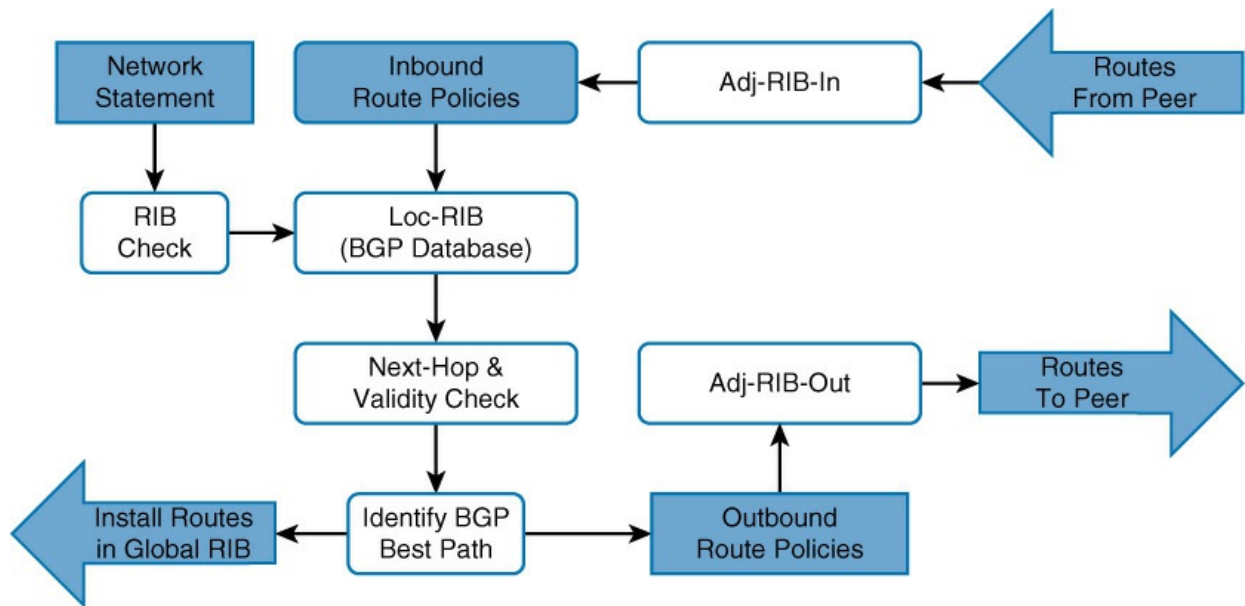


Figure 10-11 BGP Database Processing for All Routes

The command **show bgp address-family address-family-modifier** displays the contents of the BGP database (Loc-RIB) on IOS and IOS XR nodes. Every entry in the BGP Loc-RIB table contains at least one route, but could contain multiple routes for the same network prefix.

Example 10-7 displays the BGP table for XR1 and R2. The BGP table contains received routes and locally generated routes.

### Example 10-7 Display of BGP Table

[Click here to view code image](#)



```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

```
BGP main routing table version 6
```

```
Status codes: s suppressed, d damped, h history, * valid, > best  
i - internal, r RIB-failure, S stale
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 10.12.1.0/24	0.0.0.0	0		32768	i
* i	10.12.1.2	0	100	0	i
*> 192.168.1.1/32	0.0.0.0	0		32768	i
*>i192.168.2.2/32	10.12.1.2	0	100	0	i

Processed 2 prefixes, 2 paths

```
R2#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

```
BGP table version is 3, local router ID is 192.168.2.2
```

```
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,  
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,  
x best-external, a additional-path, c RIB-compressed,
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete
```

```
RPKI validation codes: V valid, I invalid, N Not found
```

Network	Next Hop	Metric	LocPrf	Weight	Path
* i10.12.1.0/24	10.12.1.1	0	100	0	i
*>	0.0.0.0	0		32768	i
*>i192.168.1.1/32	10.12.1.1	0	100	0	i
*> 192.168.2.2/32	0.0.0.0	0		32768	i

**Table 10-3** explains the fields of output when displaying the BGP Table.

Field	Description
Network	List of the network prefixes installed in BGP. If multiple NLRI exist for the same prefix, only the first prefix is identified, and others will leave a blank space. Valid NLRI are indicated by the asterisk (*). The NLRI selected as the best path is indicated by an angle bracket (>). iBGP routes are indicated by the <i>i</i> next to the prefix.
Next Hop	Next hop: A well-known mandatory BGP path attribute that defines the IP address for the next hop for that specific NLRI.
Metric	Multiple-Exit Discriminator (MED): An optional nontransitive BGP path attribute used in BGP algorithm for that specific NLRI.
LocPrf	Local preference: A well-known discretionary BGP path attribute used in the BGP best path algorithm for that specific NLRI.
Weight	Locally significant Cisco-defined attribute used in the BGP best path algorithm for that specific NLRI.
Path and Origin	AS_PATH (AS_Path): A well-known mandatory BGP path attribute used for loop-prevention and in the BGP best path algorithm for that specific NLRI. Origin: A well-known mandatory BGP path attribute used in the BGP best path algorithm. A value of <i>i</i> represents an IGP, <i>e</i> for EGP, and ? for a route that was redistributed into BGP.

**Table 10-3** BGP Table Fields

The Adj-RIB-Out table maintains a unique table for each BGP peer. This enables a network engineer to view routes advertised to a specific router. The command **show bgp address-family address-family-modifier neighbor ip-address advertised routes** displays the contents of the Adj-RIB-out table for that neighbor.

**Example 10-8** displays the Adj-RIB-Out entries specific to each neighbor. Notice that the next-hop address reflects the local router and will be changed as the router advertised to the peer.

### **Example 10-8** Neighbor-Specific View of the Adj-RIB-Out

**Click here to view code image**

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast neighbors 10.12.1.2 advertised-routes
Network          Next Hop        From           AS Path
10.12.1.0/24     10.12.1.1      Local          i
192.168.1.1/32   10.12.1.1      Local          i

Processed 2 prefixes, 2 paths
```

```
R2#show bgp ipv4 unicast neighbors 10.12.1.1 advertised-routes
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 10.12.1.0/24	0.0.0.0	0		32768	i
*> 192.168.2.2/32	0.0.0.0	0		32768	i

```
Total number of prefixes 2
```

Note

Only the prefix, next hop, metric, local preference, weight, AS\_Path, and origin path attributes are visible when viewing the advertised routes.

You can also use the **show bgp ipv4 unicast summary** command to verify the exchange of NLRIs between nodes, as shown in [Example 10-9](#).

### Example 10-9 BGP Summary with Prefixes

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast summary
```

```
! Output omitted for brevity
```

Neighbor	Spk	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	St/PfxRcd
10.12.1.2	0	100	10	12	14	0	0	00:05:13	2

```
R2#show bgp ipv4 unicast summary
```

```
! Output omitted for brevity
```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
10.12.1.1	4	100	11	10	9	0	0	00:04:56	2

The BGP routes in the global IP routing table (RIB) are displayed with the command **show ip route bgp** on IOS nodes, and with the command **show route bgp** on IOS XR nodes. [Example 10-10](#) demonstrates these commands in the topology example. The prefixes are from an iBGP session and have an AD of 200, and no metric is present.

### Example 10-10 Displaying of BPG Routes in IP Routing Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route bgp
```

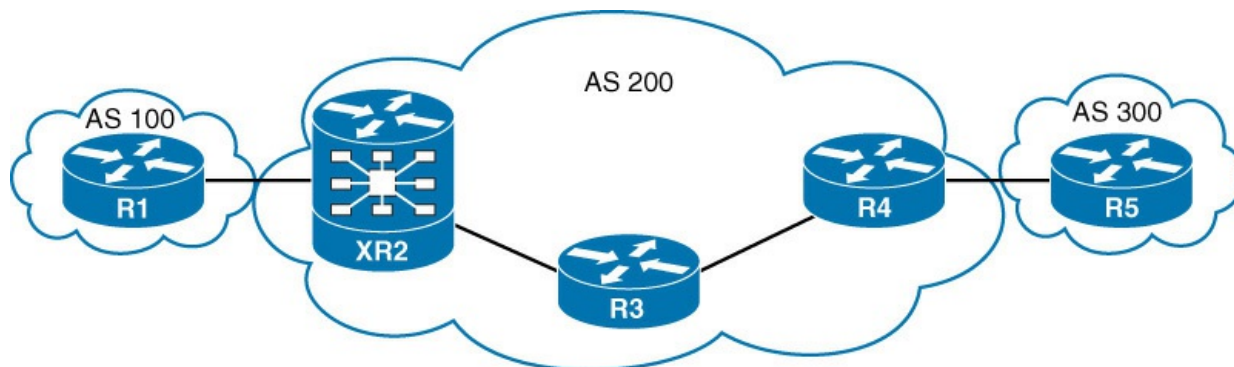
```
B 192.168.2.2/32 [200/0] via 10.12.1.2, 00:02:16
```

```
R2#show ip route bgp
```

```
192.168.1.0/32 is subnetted, 1 subnets  
B 192.168.1.1 [200/0] via 10.12.1.1, 00:02:43
```

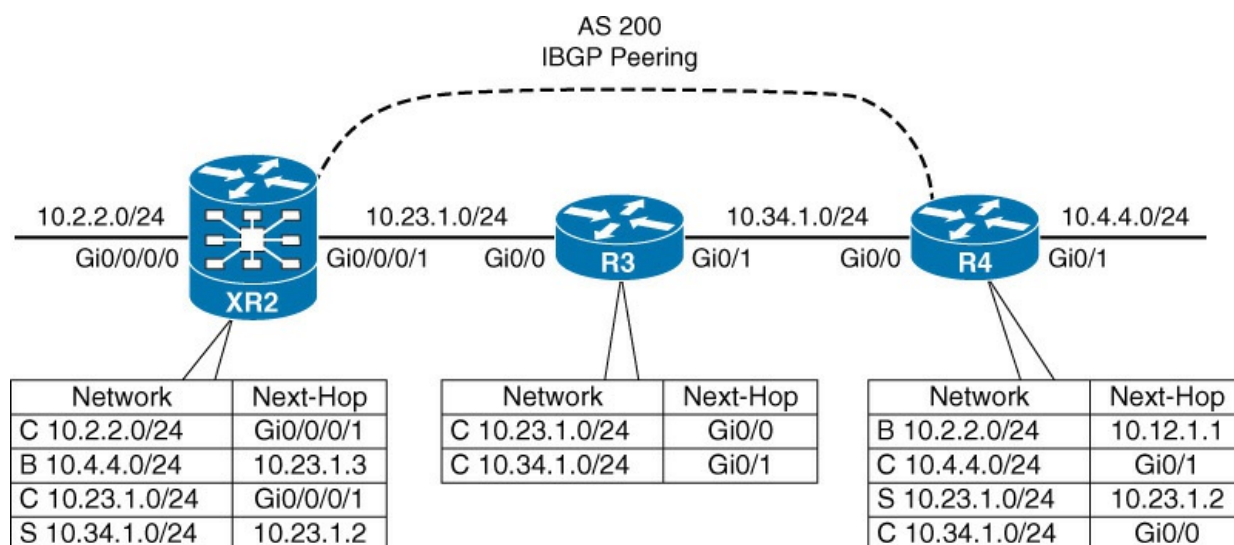
## IBGP

The need for BGP within an autonomous system typically occurs when the need for multiple routing policies exist, or when transit connectivity is provided between autonomous systems. In [Figure 10-12](#), AS200 provides transit connectivity to AS100 and AS300. AS100 connects to AS200 at XR2, and connects AS300 to AS200 at R4.



**Figure 10-12** AS200 Provides Transit Connectivity

[Figure 10-13](#) displays AS200's topology in greater detail. XR2 could form a session directly with R4, but R3 would not know where to route traffic from AS100 or AS300 when it must transit AS200. The drawing illustrates that R3 does not have the appropriate route forwarding information even for AS200's network. R3's routing table must be complete to route traffic that transits AS200.



**Figure 10-13** iBGP Prefix Advertisement Behavior

Advertising the full BGP table into an IGP is not a viable solution for the following reasons:

- **Scalability:** At the time of this writing, the Internet has 500,000+ IPv4 networks and continues to increase in size. IGPs cannot scale to that level of routes.
- **Custom routing:** Link-state protocols and distance vector routing protocols use metric as the primary method for route selection. IGPs will always use this routing pattern for path selection. BGP uses multiple steps to identify the best path and allows for BGP path attributes to manipulate the path for a specific prefix (NLRI). BGP could select a best path that has more hops, which would normally be deemed suboptimal from an IGP's perspective.

■ **Path attributes:** All the BGP path attributes cannot be maintained within IGP protocols. Only BGP is capable of maintaining the path attribute as the prefix is advertised from one edge of the autonomous system to the other edge.

### iBGP Full-Mesh Requirement

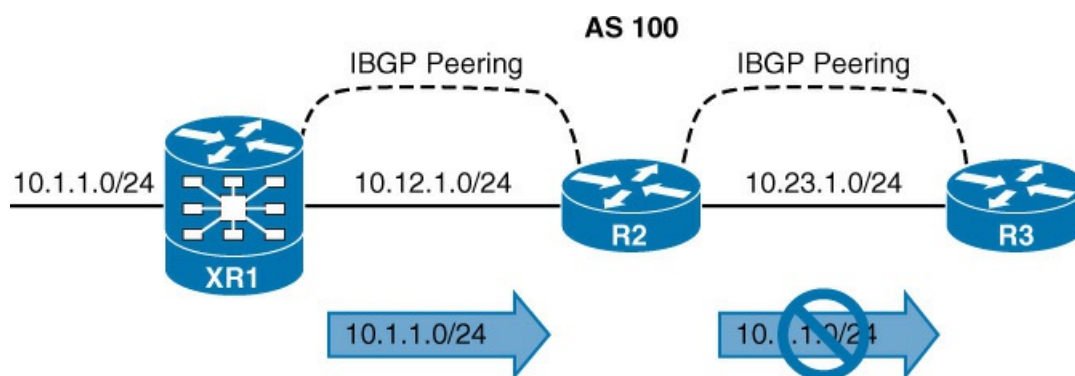
Earlier in this chapter, the AS\_Path was explained as a loop-prevention mechanism because the ASN is prepended when advertising to an eBGP neighbor. iBGP peers do not prepend their ASN to the AS\_Path because the NLRIs would fail the validity check and would not install the prefix into the IP routing table.

No other method exists to detect loops with iBGP sessions, and RFC 4271 prohibits the advertisement of an NLRI received from an iBGP peer to another iBGP peer. RFC 4271 states that all BGP routers within a single autonomous system must be fully meshed to provide a complete loop-free routing table and prevent traffic blackholing.

#### Note

At times during high route convergence, a router may advertise an iBGP route withdraw to another iBGP peer as a loop-avoidance mechanism that is within RFC compliance.

In [Figure 10-14](#), XR1, R2, and R3 are all within AS100. XR1 has an iBGP session with R2, and R2 has an iBGP session with R3. XR1 advertises the 10.1.1.0/24 prefix to R2, which is processed and inserted into R2's BGP table. R2 does not advertise the 10.1.1.0/24 NLRI to R3 because it received the prefix from an iBGP peer.



**Figure 10-14** iBGP Prefix Advertisement Behavior

In [Figure 10-15](#), XR1 and R3 form a multihop iBGP session so that R3 can receive the 10.1.1.0/24 prefix. XR1 connects to R3's 10.23.1.3 IP address and R3 connects to XR1's 10.12.1.1 IP address. XR1 and R3 will need a static route to the remote transit link, or R2 will need to advertise the 10.12.1.0/24 and 10.23.1.0/24 network into BGP.

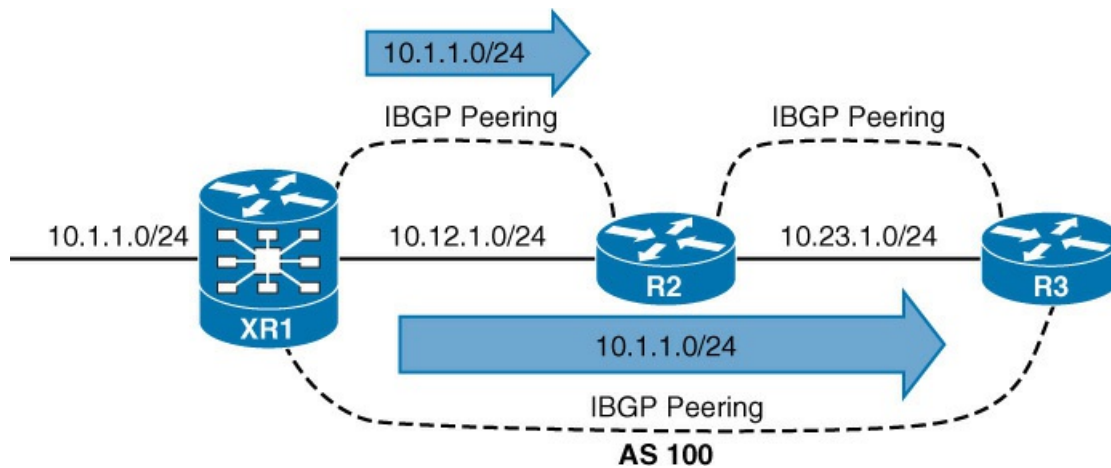


Figure 10-15 iBGP Full-Mesh Topology

### Peering via Loopback Addresses

BGP sessions are sourced by the primary IP address of the outbound interface toward the BGP peer by default. In Figure 10-16, XR1, R2, and R3 are a full mesh of iBGP sessions peered by transit links.

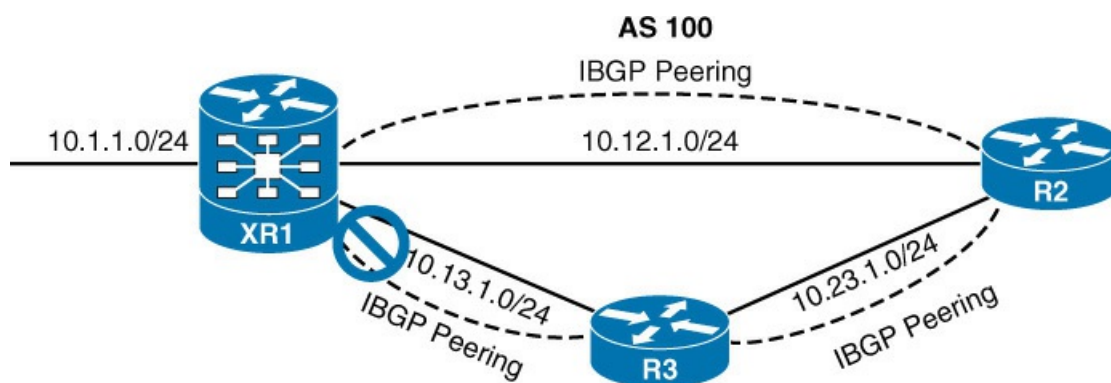


Figure 10-16 Link Failure on Full-Mesh iBGP Topology

In the event of a transit link failure on the 10.13.1.0/24 network, R3's BGP session with XR1 will time out and terminate. R3 will lose connectivity to 10.1.1.0/24 network even though a multihop path is viable through R2. The loss of connectivity occurs because iBGP does not advertise routes learned from another iBGP peer, as shown earlier.

Two solutions exist to overcome the link failure:

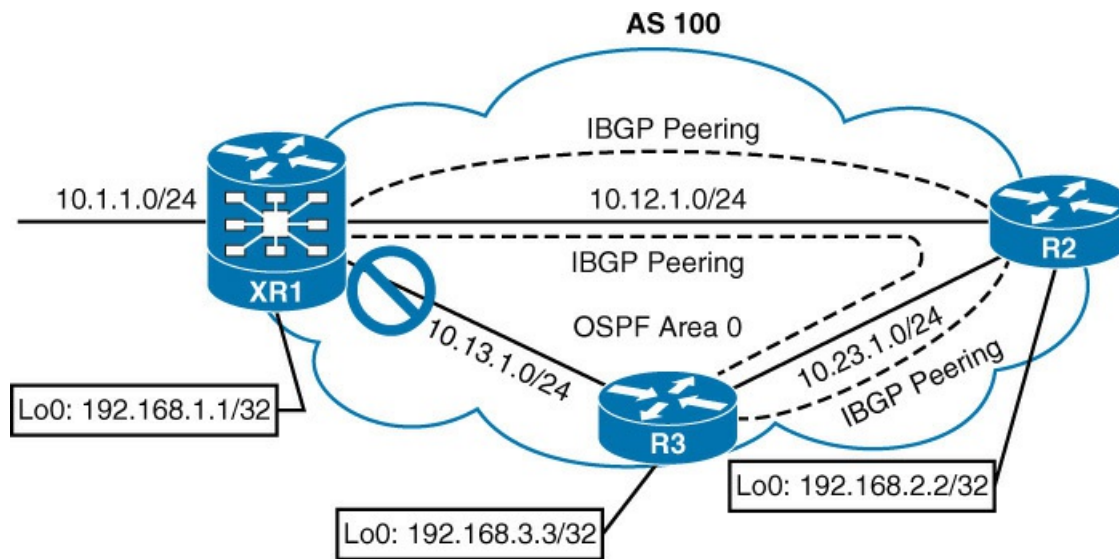
- Add a second link between all routers (three links will become six links) and establish two BGP sessions between each router.
- Configure an IGP on the routers transit links, advertise loopback interfaces into the IGP, and then configure the BGP neighbors to establish a session to the remote router's loopback address.

The second method is more efficient and preferable of the two methods

The loopback interface is virtual and will always stay up. In the event of link failure, the session stays intact while the IGP finds another path to the loopback address, and in essence turns a single-hop iBGP session into a multihop iBGP session.

Figure 10-17 illustrates the concept of peering using loopback addresses after the 10.13.1.0/24

network link fails. XR1 and R3 still maintain BGP session connectivity while routes learned from OSPF allow BGP communication traffic between the loopbacks via R2.



**Figure 10-17** Link Failure with iBGP Sessions on Loopback Interfaces

Note

BGP can be thought of as a control plane routing protocol because it allows for the exchanging routes with peers multiple hops away. BGP routers do not have to be in the data plane (path) to exchange prefixes, but all routers in the data path need to know all the routes that will be forwarded through them.

Updating the BGP configuration to set the destination of the BGP session to the remote router's loopback IP address is not enough. The source IP address of the BGP packets will still reflect the IP address of the outbound interface. When a BGP packet is received, the router correlates the source IP address of the packet to the BGP neighbor table. If the BGP packet source does not match an entry in the neighbor table, the packet cannot be associated to a neighbor, and is discarded.

The source of BGP packets can be set statically to a specific interface's primary IP address with the BGP session configuration command **neighbor ip-address update-source interface-type interface-number** on IOS nodes. IOS XR nodes use the command **update-source interface-type interface-number** under the neighbor session within the BGP router configuration.

Example 10-11 provides the BGP configuration for the topology in Figure 10-17.

**Example 10-11** BGP Configuration Source from Loopback Interfaces

[Click here to view code image](#)

**XR1**

```
router ospf 1
  area 0
    interface Loopback0
    !
    interface GigabitEthernet0/0/0/2
    !
    interface GigabitEthernet0/0/0/3
    !
    !
  !
router bgp 100
  bgp router-id 192.168.1.1
  address-family ipv4 unicast
    network 10.1.1.0/24
  !
  neighbor 192.168.2.2
    remote-as 100
    update-source Loopback0
    address-family ipv4 unicast
  !
  !
  neighbor 192.168.3.3
    remote-as 100
    update-source Loopback0
    address-family ipv4 unicast
```

**R2**

```
router ospf 1
  network 10.0.0.0 0.255.255.255 area 0
  network 192.168.2.2 0.0.0.0 area 0
  !
router bgp 100
  no bgp default ipv4-unicast
  neighbor 192.168.1.1 remote-as 100
  neighbor 192.168.1.1 update-source Loopback0
  neighbor 192.168.3.3 remote-as 100
  neighbor 192.168.3.3 update-source Loopback0
  !
  address-family ipv4
    neighbor 192.168.1.1 activate
    neighbor 192.168.3.3 activate
  exit-address-family
```



### R3

```
router ospf 1
 network 10.0.0.0 0.255.255.255 area 0
 network 192.168.3.3 0.0.0.0 area 0
!
router bgp 100
 no bgp default ipv4-unicast
 neighbor 192.168.1.1 remote-as 100
 neighbor 192.168.1.1 update-source Loopback0
 neighbor 192.168.2.2 remote-as 100
 neighbor 192.168.2.2 update-source Loopback0
!
 address-family ipv4
  neighbor 192.168.1.1 activate
  neighbor 192.168.2.2 activate
 exit-address-family
```

**Example 10-12** displays the session summary information of XR1 and R3. Notice that the neighbor IP addresses are the loopback IP addresses.

### Example 10-12 BGP IPv4 Session Summary of Loopback Interfaces

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast summary
```

```
! Output omitted for brevity
```

```
BGP router identifier 192.168.1.1, local AS number 100
```

```
BGP main routing table version 4
```

Process Speaker	RcvTblVer 4	bRIB/RIB 4	LabelVer 4	ImportVer 4	SendTblVer 4	StandbyVer 4			
Neighbor	Spk	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	St/PfxRcd
192.168.2.2	0	100	8	7	4	0	0	00:04:23	1
192.168.3.3	0	100	6	5	4	0	0	00:04:34	1

```
R3#show bgp ipv4 unicast summary
```

```
BGP router identifier 192.168.2.2, local AS number 100
```

```
BGP table version is 1, main routing table version 1
```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
192.168.1.1	4	100	8	9	1	0	0	00:05:23	1
192.168.2.2	4	100	4	5	1	0	0	00:03:22	1

**Example 10-13** displays R3's BGP table. Notice that the next-hop IP address is XR1's loopback address. When R2 and R3 forward packets to the 10.1.1.0/24 network, a recursive lookup will be performed to determine the outbound interface for the 192.168.1.1 IP address.

### Example 10-13 R3's BGP Table

[Click here to view code image](#)

```
R3#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 10.1.1.1/24	192.168.1.1	0	100	0	I

#### Note

Sourcing BGP sessions from loopback interfaces eliminates the need to recompute the BGP best path algorithm if a peering link fails, as shown in [Figure 10-17](#). It also provides automatic load balancing if there are multiple equal-cost paths via IGP to the loopback address.

## EBGP

eBGP peerings are the core component of BGP on the Internet. eBGP is the exchange of network prefixes between autonomous systems. The following behaviors are different on eBGP sessions when compared to iBGP sessions:

- Time-To-Live (TTL) on BGP packets is set to 1. BGP packets will drop in transit if a multihop BGP session is attempted. (TTL on iBGP packets is set to 255 which allows for multihop sessions.)
- The advertising router modifies the BGP next hop to the IP address sourcing the BGP connection.
- The advertising router prepends its ASN to the existing AS\_Path.
- The receiving router verifies that the AS\_Path does not contain an ASN that matches the local router's. BGP discards the NLRI if it fails the AS\_Path loop-prevention check.

The configuration for eBGP and iBGP sessions are fundamentally the same on IOS and IOS XR nodes except that the ASN in the **remote-as** statement differs from the ASN defined in the BGP process. IOS XR requires a routing policy to be associated with an eBGP peers as a security measure to ensure that routes are not accidentally accepted or advertised. If a route policy is not configured in the appropriate address family, routes are discarded upon receipt, and no routes are advertised to eBGP peers.

An inbound and outbound route policy is configured with the command **route-policy route-policy-name {in | out}** under the BGP neighbor address family configuration. Route policies are explained in detail in [Chapter 11](#), “Route Maps and Route Policy,” but a simple *PASSALL* route policy is provided in [Example 10-14](#).

#### Note

Although not required, IOS uses route maps to apply different route policies for BGP neighbors with the address family configuration command **neighbor ip-address route-map route-map-name {in | out}**. Route maps are explained in detail in [Chapter 11](#).

[Figure 10-18](#) provides a simple eBGP topology. R1 (AS100) is peering with XR2 (AS200), and XR2 is a neighbor with R3 (AS300). All three routers will advertise their Loopback 0 interfaces into BGP.

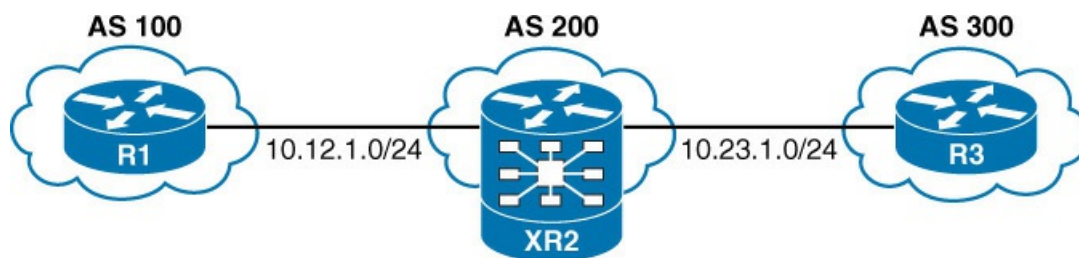


Figure 10-18 eBGP Topology

Example 10-14 demonstrates how to configure the routers for the eBGP topology. Notice the simple PASSALL route policy in XR2's configuration. R3's configuration is similar to R1's configuration.

### Example 10-14 eBGP Configuration

[Click here to view code image](#)

#### R1

```
router bgp 100
  no bgp default ipv4-unicast
  neighbor 10.12.1.2 remote-as 200
  !
  address-family ipv4
    network 192.168.1.1 mask 255.255.255.255
    neighbor 10.12.1.2 activate
  exit-address-family
```

#### XR2

```
router bgp 200
  address-family ipv4 unicast
    network 192.168.2.2/32
  !
  neighbor 10.12.1.1
    remote-as 100
  address-family ipv4 unicast
    route-policy PASSALL in
    route-policy PASSALL out
  !
  !
  neighbor 10.23.1.3
    remote-as 300
  address-family ipv4 unicast
    route-policy PASSALL in
    route-policy PASSALL out
  !
  route-policy PASSALL
    pass
  end-policy
```

Note

Different outbound (or inbound) route policies may be different from neighbor to neighbor, which allows for a dynamic routing policy within an autonomous system

**Example 10-15** displays the BGP table for all three routers. The AS\_Path of the most recent autonomous system is always prepended (the furthest to the left). From R1's perspective, the 192.168.3.3/32 prefix originates in AS300 and is reachable via AS200. Notice that R1's BGP table for the 192.168.2.2/32 and 192.168.3.3/32 prefixes have a next hop IP address of 10.12.1.2 (R2).

In the output, the ASN on the furthest right is known as the *originating autonomous system*, and the ASN on the furthest left is known as the *neighboring autonomous system*. Any ASNs in between are known as *transit autonomous systems*.

### Example 10-15 R1's, XR2's, and R3's BGP Table

[Click here to view code image](#)

```
R1#show bgp ipv4 unicast
! Output omitted for brevity
      Network          Next Hop           Metric LocPrf Weight Path
*> 192.168.1.1/32      0.0.0.0            0          32768 i
*> 192.168.2.2/32      10.12.1.2          0          0 200 i
*> 192.168.3.3/32      10.12.1.2          0          0 200 300 i
```

```
RP/0/0/CPU0:XR2# show bgp ipv4 unicast
! Output omitted for brevity
      Network          Next Hop           Metric LocPrf Weight Path
*> 192.168.1.1/32      10.12.1.1          0          0 100 i
*> 192.168.2.2/32      0.0.0.0            0          32768 i
*> 192.168.3.3/32      10.23.1.3          0          0 300 i

Processed 3 prefixes, 3 paths
```

```
R3#show bgp ipv4 unicast
! Output omitted for brevity
      Network          Next Hop           Metric LocPrf Weight Path
*> 192.168.1.1/32      10.23.1.2          0          0 200 100 i
*> 192.168.2.2/32      10.23.1.2          0          0 200 i
*> 192.168.3.3/32      0.0.0.0            0          32768 i
```

You can display the BGP attributes for all paths to a specific network prefix with the command **show bgp ipv4 unicast network** on IOS and IOS XR nodes.

**Examples 10-16** and **10-17** display the BGP path attributes for the remote prefix (192.168.3.3/32) and local prefix (192.168.1.1/32), respectively.

### Example 10-16 BGP Prefix Attributes for Remote Prefix

[Click here to view code image](#)

```
R1#show bgp ipv4 unicast 192.168.3.3
BGP routing table entry for 192.168.3.3/32, version 11
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 1
  200 300
    10.12.1.2 from 10.12.1.2 (192.168.2.2)
      Origin IGP, localpref 100, valid, external, best
```

### **Example 10-17** *BGP Prefix Attributes for Local Prefix*

[Click here to view code image](#)

```
R1#show bgp ipv4 unicast 192.168.1.1
BGP routing table entry for 192.168.1.1/32, version 7
Paths: (1 available, best #1, table default)
  Advertised to update-groups:
    2
  Refresh Epoch 1
  Local
    0.0.0.0 from 0.0.0.0 (192.168.1.1)
      Origin IGP, metric 0, localpref 100, weight 32768, valid, sourced, local, best
```

**Table 10-4** explains the output provided in the previous two examples and their correlation to BGP. Some of the BGP path attributes may change depending on the BGP features used.

Output	Description
Paths: (1 available, best #1)	Provides a count of BGP paths in the BGP Loc-RIB and identifies the path selected as the BGP best path. All the paths and BGP attributes are listed after this.
Not advertised to any peer	Identifies whether the prefix was advertised to a BGP peer. BGP neighbors are consolidated into BGP update groups. Explicit neighbors can be seen with the command <b>show bgp ipv4 unicast update-group</b> on IOS or IOS XR nodes.
200 300	This is the AS_Path for the NLRI as it was received.
10.12.1.2 from 10.12.1.2 (192.168.2.2)	The first entry lists the IP address of the eBGP edge peer. The <i>from</i> field lists the IP address of the iBGP router that received this route from the eBGP edge peer (In this case, the route was learned from an eBGP edge peer, so the address will be the eBGP edge peer.) Expect this field to change when an external route is learned from an iBGP peer. The number in parentheses is the BGP identifier (RID) for that node.
Origin IGP	The origin is the BGP well-known mandatory attribute that states the mechanism for advertising this route. In this instance, it is an internal route.
metric 0	Displays the optional nontransitive BGP attribute Multiple-Exit Discriminator (MED), also known as BGP metric.
localpref 100	Displays the well-known discretionary BGP attribute local preference.
valid	Displays the validity of this path.
External	Displays how the route was learned. It will be internal, external, or local.

Table 10-4 BGP Prefix Attributes

### eBGP and iBGP Topologies

Combining eBGP sessions with iBGP sessions can cause confusion in terminology and concepts.

Figure 10-19 provides a reference topology for clarifying the concepts presented earlier regarding eBGP and iBGP router advertisements. XR1 and XR2 form an eBGP session, R3 and R4 form an eBGP session as well, and XR2 and R3 form an iBGP session. XR2 and R3 are iBGP peers and follow the rules of iBGP advertisement even if the routes are learned from an eBGP peer.

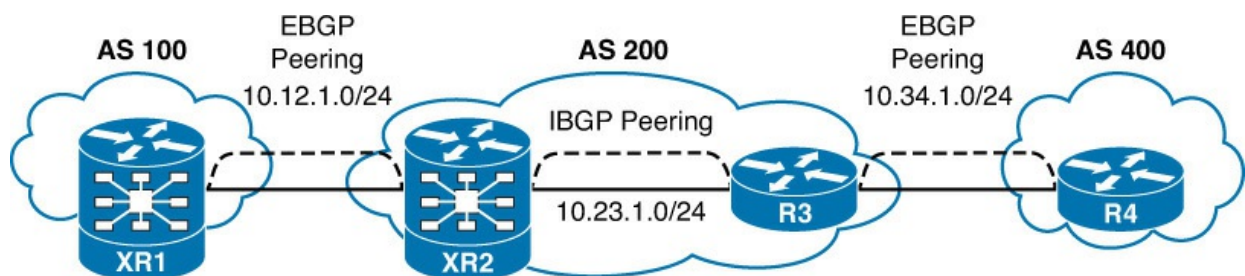


Figure 10-19 eBGP and iBGP Topology

As an eBGP prefix is advertised to an iBGP neighbor, issues may arise with the NLRI passing the validity check and next-hop reachability check, thus preventing advertisements to other BGP peers. The most common issue involves the failure of the next-hop accessibility check. iBGP

peers do not modify the next-hop address if the NLRI has a next-hop address other than 0.0.0.0. The next-hop address must be resolvable in the global RIB in order for it to be valid and advertised to other BGP peers.

To demonstrate this concept, only XR1 and R4 have advertised their loopback interfaces into BGP, 192.168.1.1/32 and 192.168.4.4/32. Figure 10-20 displays the BGP table for all four routers. Notice that the BGP best path symbol (>) is missing for the 192.168.4.4/32 prefix on XR2 and for the 192.168.1.1/32 on R3.

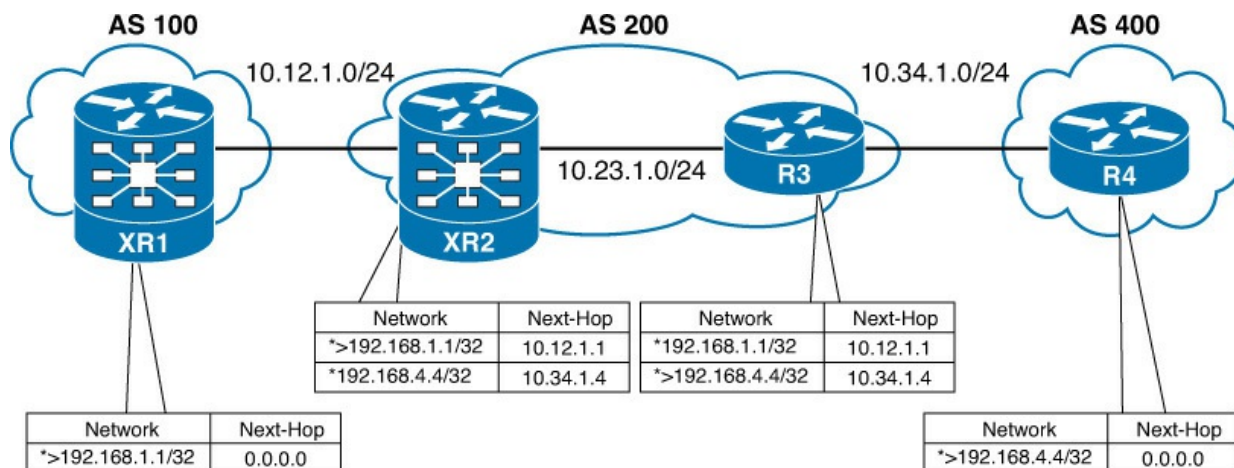


Figure 10-20 BGP Next-Hop Addresses on an eBGP and iBGP Topology

XR1's BGP table is missing the 192.168.4.4/32 prefix because XR2 did not advertise the network prefix. The prefix did not pass XR2's next-hop accessibility check, which prevented the execution of the BGP best path algorithm. R4 advertised the prefix to R3 with the next-hop address of 10.34.1.4, and R3 advertised the prefix to XR2 with a next-hop address of 10.34.1.4. XR2 does not have a route for the 10.34.1.4 IP address, and deems the next hop inaccessible. Because XR2 did not install the 192.168.4.4/32 prefix with a best path, it never advertised the route to XR1. The same logic applies to XR1's 192.168.1.1/32 prefix when advertised toward R4.

Example 10-18 shows the BGP attributes on R3 for the 192.168.1.1/32 prefix. Notice that the prefix is not advertised to any peer because the next-hop is *inaccessible*.

### Example 10-18 BGP Path Attributes for 192.168.1.1/32

[Click here to view code image](#)

```
R3#show bgp ipv4 unicast 192.168.1.1
BGP routing table entry for 192.168.1.1/32, version 2
Paths: (1 available, no best path)
  Not advertised to any peer
  Refresh Epoch 1
  100
  10.12.1.1 (inaccessible) from 10.23.1.2 (192.168.2.2.2)
  Origin IGP, metric 0, localpref 100, valid, internal
```

To correct the issue, the peering links, 10.12.1.0/24 and 10.34.1.0/24, need to be in both XR2 and R3's routing table via either technique:

- IGP advertisement. Remember to use the passive interface to prevent an adjacency from forming. IGP's do not provide a filtering capability like BGP does.
- Advertising the networks into BGP.

Both techniques allow the prefixes to pass the next-hop accessibility test.

Figure 10-21 displays the topology with both transit links advertised into BGP. Notice that this time all four prefixes are valid, with a BGP best path selected.

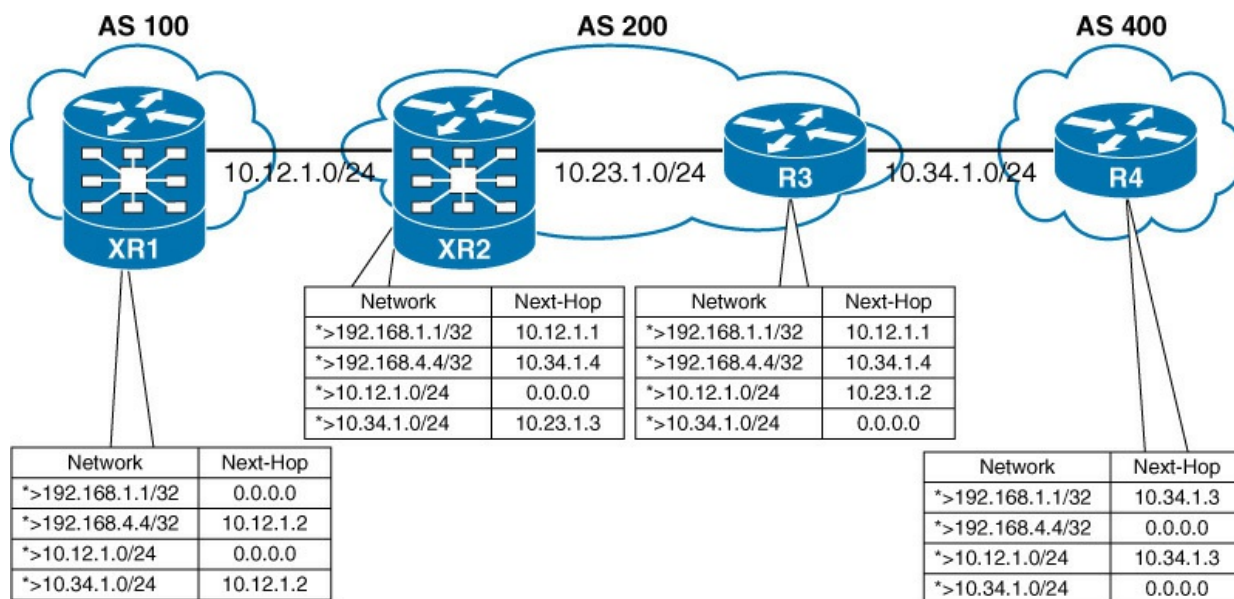


Figure 10-21 eBGP and iBGP Topology with Peer Links

Example 10-19 verifies that R4 has connectivity to XR1 after the peering links are advertised into BGP.

### Example 10-19 Verification of Connectivity Between XR1 and R4

[Click here to view code image](#)

```
R4#ping 192.168.1.1 source loopback0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.4.4
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms
```

### Next-Hop Manipulation

Imagine a service provider network with 500 routers, and every router has 200 eBGP peering sessions. To ensure that the next-hop address is reachable to the iBGP peers, the advertisement of 100,000 peering networks in BGP or IGP must be done. This consumes router resources in the entire autonomous system.

Another technique to ensure that the next-hop address check passes without advertising peering networks into a routing protocol involves the modification of the next-hop address. The next-hop IP address can be modified on inbound or outbound neighbor routing policies. Managing IP addresses by BGP routing policy can be a complicated task with a high number of eBGP peers.



The **next-hop-self** feature modifies the next-hop address in the route's path attribute for external BGP prefixes.

The command **neighbor ip-address next-hop-self [all]** is used for each neighbor under the address family configuration on IOS routers, and the command **next-hop-self** is applied under the neighbor address family configuration for IOS XR routers.

Example 10-20 demonstrates the next-hop-self configuration of XR2 and R3 from Figure 10-21. Notice that the next-hop-self is used only on the iBGP peerings.

### **Example 10-20** XR2 and R3 Next-Hop-Self

[Click here to view code image](#)

#### **XR2**

```
router bgp 200
  address-family ipv4 unicast
  !
  neighbor 10.12.1.1
    remote-as 100
    address-family ipv4 unicast
      route-policy PASSALL in
      route-policy PASSALL out
    !
  !
  neighbor 10.23.1.3
    remote-as 200
    address-family ipv4 unicast
      next-hop-self
```

#### **R3**

```
router bgp 200
  bgp log-neighbor-changes
  no bgp default ipv4-unicast
  neighbor 10.23.1.2 remote-as 200
  neighbor 10.34.1.4 remote-as 400
  !
  address-family ipv4
    network 3.3.3.3 mask 255.255.255.255
    neighbor 10.23.1.2 activate
    neighbor 10.23.1.2 next-hop-self
    neighbor 10.34.1.4 activate
  exit-address-family
```

Figure 10-22 shows the topology and BGP routing table for all four routers. Notice that XR2 and R3 advertised the eBGP routes to each other with the next-hop address as the iBGP session IP address, allowing the NLRIs to pass the next-hop accessibility check.

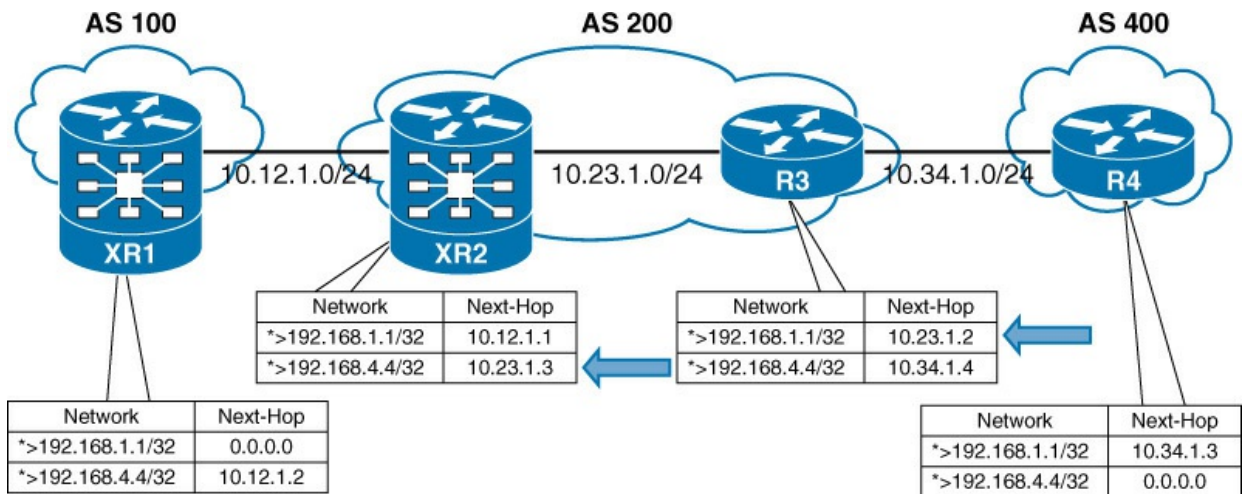


Figure 10-22 eBGP and iBGP Topology with Next-Hop-Self

Note

The **next-hop-self** feature does not modify the next-hop address for iBGP prefixes by default. IOS nodes can append the optional **all** keyword, which modifies the next-hop address on iBGP prefixes, too. IOS XR provides the BGP configuration command **ibgp policy out enforce-modifications**, which will modify iBGP routes in the same manner as eBGP routes.

## IBGP SCALABILITY

The inability for BGP to advertise a prefix learned from one iBGP peer to another iBGP peer can lead to scalability issues within an autonomous system. Figure 10-23 shows 6 routers requiring 15 sessions to maintain the iBGP full mesh.

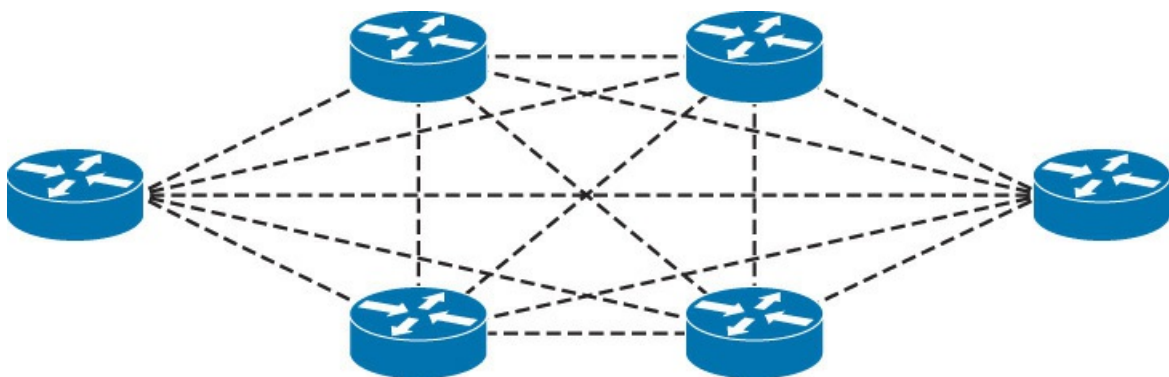


Figure 10-23 iBGP Full-Mesh Sessions

The formula  $n(n - 1) / 2$  provides the number of sessions required, where  $n$  represents the number of routers. Figure 10-24 shows the exponential growth of BGP sessions, which cannot be obtainable in an environment with hundreds of routers running BGP.

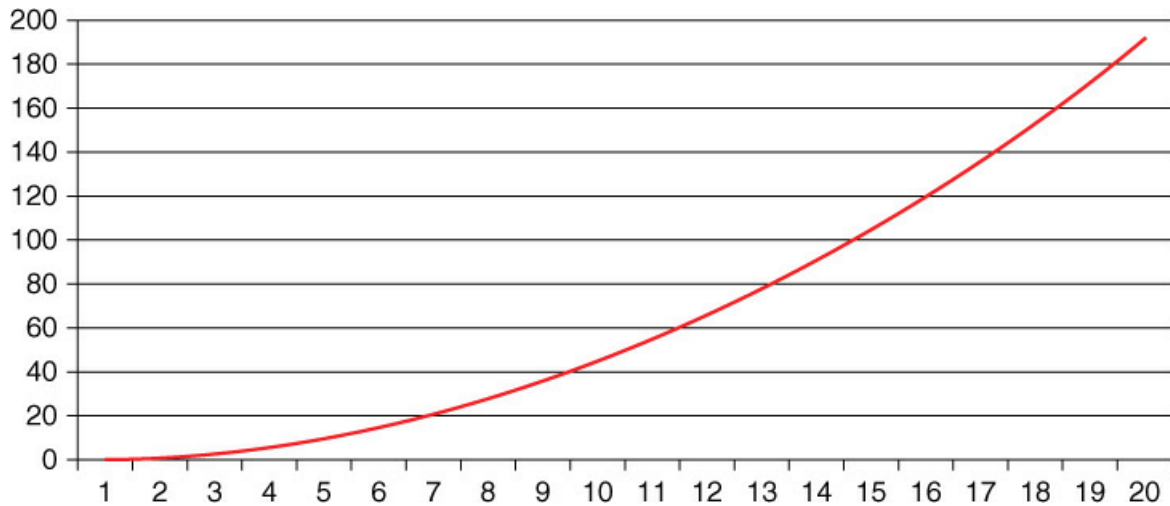


Figure 10-24 iBGP Full-Mesh Graph

### Route Reflectors

RFC 1966 introduced the concept that an iBGP peering can be configured so that it reflects routes to another iBGP peer. The router reflecting routes is known as a *route reflector* (RR), and the router receiving reflected routes is a *route reflector client*. Three basic rules involve route reflectors and route reflection:

■ **Rule 1:** If an RR receives a route from a non-RR client, the RR will advertise the route to an RR client. It will not advertise the route to a non-RR client. Figure 10-25 demonstrates rule 1 for route reflectors.

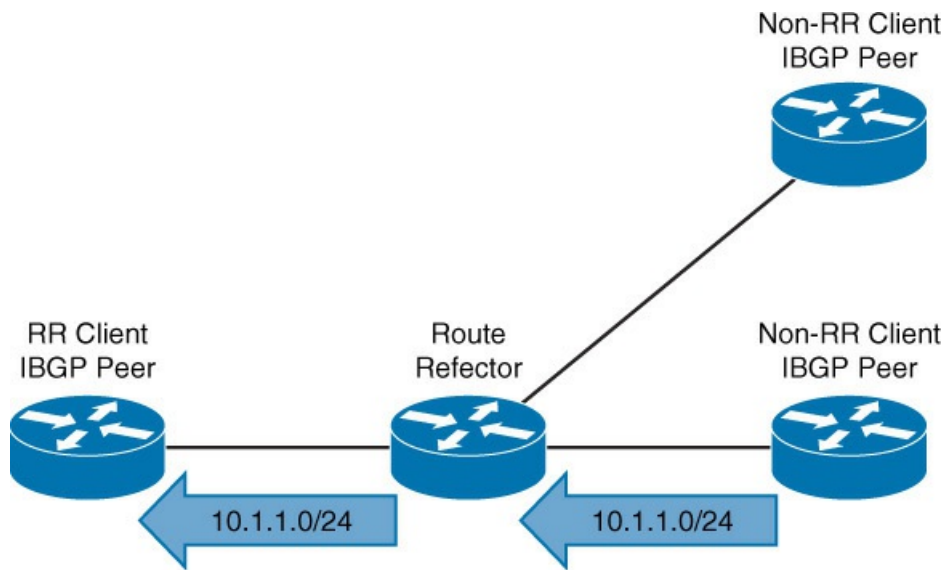


Figure 10-25 Route Received from Non-RR Client

■ **Rule 2:** If a RR receives a route from a RR client, it will advertise the route to RR clients and non-RR clients. Even the RR client that sent the advertisement will receive a copy of the route, but it discards the route because it sees itself as the route originator. Figure 10-26 demonstrates rule 2 for RRs.

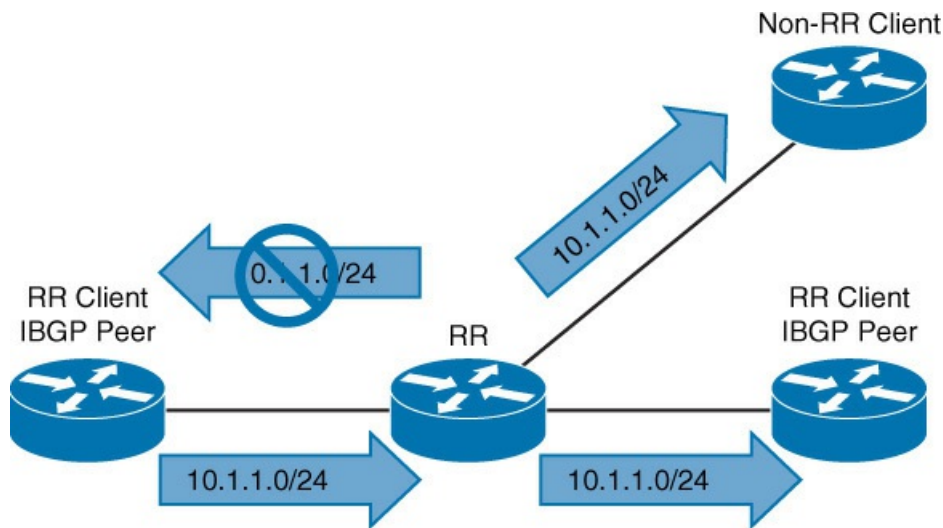


Figure 10-26 Route Received from a Route Reflector Client

■ **Rule 3:** If a RR receives a route from an eBGP peer, it will advertise the route to RR clients and non-RR clients. Figure 10-27 demonstrates rule 3 for route reflectors.

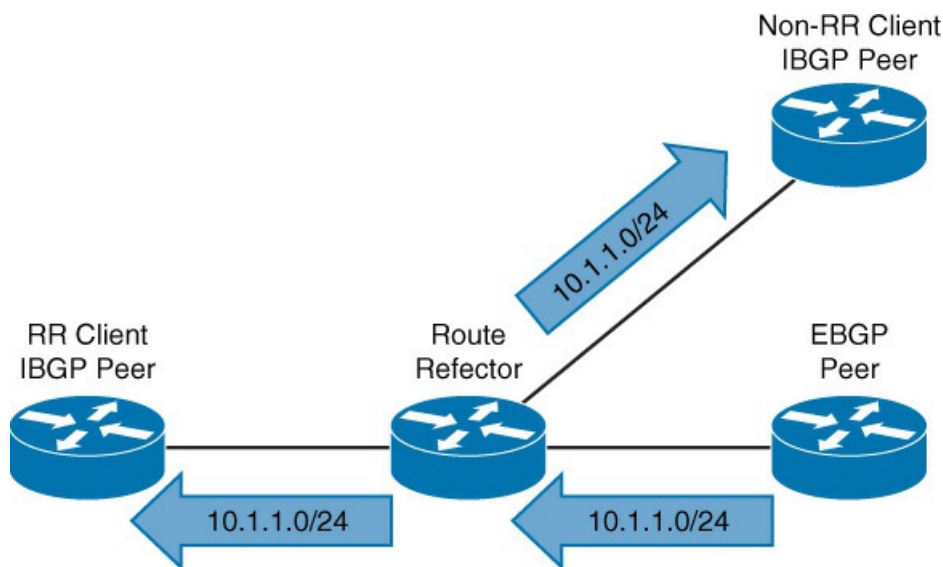


Figure 10-27 Route Received by eBGP Peer

Only RRs are aware of this change in behavior because no additional BGP configuration is performed on RR clients. BGP route reflection is specific to each address family. The command **neighbor ip-address route-reflector-client** is used on IOS nodes, and the command **route-reflector-client** is used on IOS XR nodes under the neighbor's address family configuration.

Figure 10-28 provides a simple iBGP topology to demonstrate a BGP route reflector. XR1 and R4 advertise the 10.1.1.0/24 and 10.4.4.0/24 appropriately. XR1 is a route reflector client to XR2, and R4 is an RR client to R3.

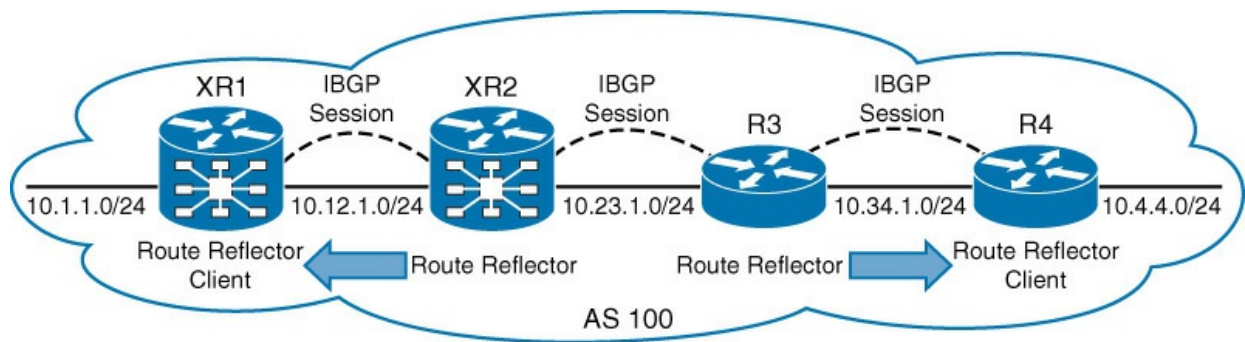


Figure 10-28 Route Reflector Topology

Example 10-21 provides the relevant BGP configuration with the RR configuration highlighted. Notice that the RR clients are configured only on XR2 and R3.

Note

You can add a description to associate a description with an IP address with the command `neighbor ip-address description description` on IOS nodes, and the command `description description` is used on the IOS XR nodes. This allows for differentiating a router by name on large configurations. Example 10-21 demonstrates the use of this command.

### Example 10-21 BGP Route Reflector Configurations

[Click here to view code image](#)

**XR1**

```
router bgp 100
 address-family ipv4 unicast
  network 10.12.1.0/24
  network 192.168.1.1/32
 !
 neighbor 10.12.1.2
  remote-as 100
  description XR2
  address-family ipv4 unicast
```

**XR2**

```
router bgp 100
 address-family ipv4 unicast
  network 10.12.1.0/24
  network 10.23.1.0/24
 !
 neighbor 10.12.1.1
  remote-as 100
  description XR1
  address-family ipv4 unicast
  route-reflector-client
 !
 !
 neighbor 10.23.1.3
  remote-as 100
 neighbor R3
  address-family ipv4 unicast
```

**R3**

```

router bgp 100
  no bgp default ipv4-unicast
  neighbor 10.23.1.2 remote-as 100
  neighbor 10.23.1.2 description XR2
  neighbor 10.34.1.4 remote-as 100
  neighbor 10.34.1.4 description R4

!
address-family ipv4
  network 10.23.1.0 mask 255.255.255.0
  network 10.34.1.0 mask 255.255.255.0
  neighbor 10.23.1.2 activate
  neighbor 10.34.1.4 activate
  neighbor 10.34.1.4 route-reflector-client
exit-address-family

```

**R4**

```

router bgp 100
  no bgp default ipv4-unicast
  neighbor 10.34.1.3 remote-as 100
  neighbor 10.34.1.3 description R3
!
address-family ipv4
  network 10.34.1.0 mask 255.255.255.0
  network 10.4.4.0 mask 255.255.255.0
  neighbor 10.34.1.3 activate
  neighbor 10.34.1.3 next-hop-self
exit-address-family

```

Figure 10-29 demonstrates the 10.1.1.0/24 network advertisement from XR1 to R4 using the appropriate route reflector rules shown in Figure 10-25, Figure 10-26, and Figure 10-27.

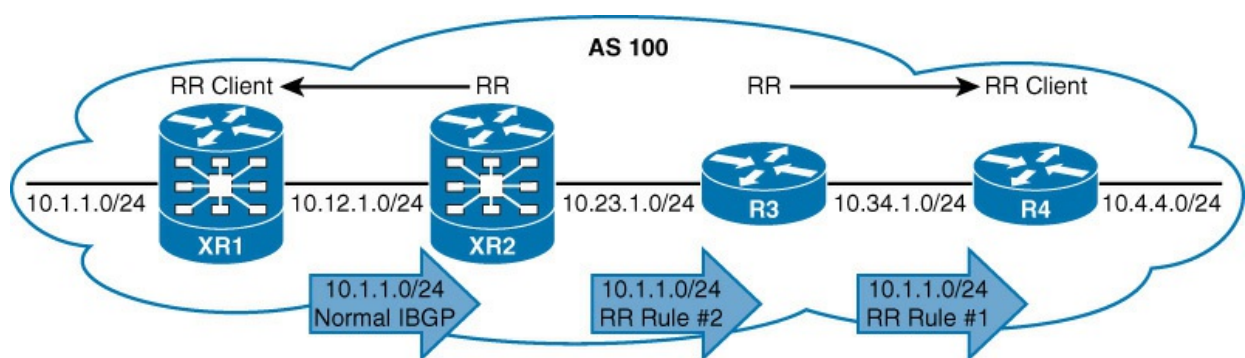


Figure 10-29 Route Reflector Rules

Example 10-22 displays XR1's and R4's BGP table. XR1 has received the 10.4.4.0/24 route and R4 has received the 10.1.1.0/24 route.

### Example 10-22 XR1's and R4's BGP Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 10.1.1.0/24	0.0.0.0	0		32768	i
*> 10.4.4.0/24	10.34.1.4	0	100	0	i
*> 10.12.1.0/24	10.12.1.2	0	100	0	i
*> 10.23.1.0/24	10.12.1.2	0	100	0	i
*> 10.34.1.0/24	10.23.1.3	0	100	0	i

```
Processed 5 prefixes, 5 paths
```

```
R4#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 10.1.1.0/24	10.12.1.1	0	100	0	i
*> 10.4.4.0/24	0.0.0.0	0		32768	i
*> 10.12.1.0/24	10.23.1.2	0	100	0	i
*> 10.23.1.0/24	10.34.1.3	0	100	0	i
r> 10.34.1.0/24	10.34.1.3	0	100	0	i

### Loop Prevention in Route Reflectors

Removing the full-mesh requirements in an iBGP topology introduces the potential for routing loops. When RFC 1966 was drafted, two other BGP RR-specific attributes were added to prevent loops:

■ **ORIGINATOR\_ID:** This optional nontransitive BGP attribute is created by the first RR and sets the value to the RID of the router that injected/advertised the route into the autonomous system. If the ORIGINATOR\_ID is already populated for a route, it should not be overwritten. If a router receives a route with its RID in the originator attribute, the route is discarded.

■ **CLUSTER\_LIST:** This nontransitive BGP attribute is updated by the RR. This attribute is appended (not overwritten) by the RR with its cluster ID. By default, this is the BGP identifier. The cluster ID can be set with the BGP configuration command **bgp cluster-id cluster-id** on IOS and IOS XR nodes. If an RR receives a route with its cluster ID in the cluster list attribute, the route is discarded.

Example 10-23 provides detailed prefix output from the RR example in Figure 10-28. Notice that the originator ID is the advertising router and that the cluster list contains both RRs listed in the order of the last RR that advertised the route.

### Example 10-23 Route Reflector Originator ID and Cluster List Attributes

[Click here to view code image](#)



```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast 10.4.4.0/24
```

```
! Output omitted for brevity
```

```
Paths: (1 available, best #1)
```

```
Local
```

```
10.34.1.4 from 10.12.1.2 (192.168.4.4)
```

```
Origin IGP, metric 0, localpref 100, valid, internal, best, group-best
```

```
Received Path ID 0, Local Path ID 1, version 7
```

```
Originator: 192.168.4.4, Cluster list: 192.168.2.2, 192.168.3.3
```

```
R4#show bgp ipv4 unicast 10.1.1.0/24
```

```
! Output omitted for brevity
```

```
Paths: (1 available, best #1, table default)
```

```
Local
```

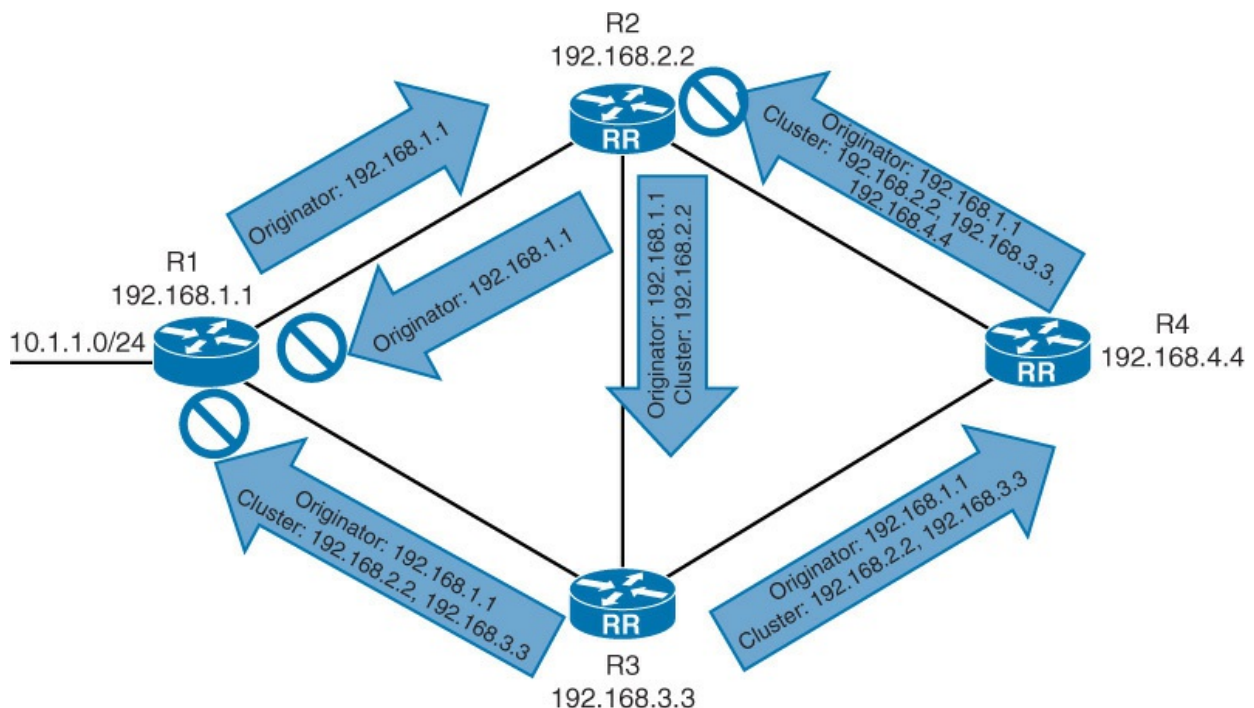
```
10.12.1.1 from 10.34.1.3 (192.168.3.3)
```

```
Origin IGP, metric 0, localpref 100, valid, internal, best
```

```
Originator: 192.168.1.1, Cluster list: 192.168.3.3, 192.168.2.2
```

**Figure 10-30** illustrates the logic of the originator ID and cluster list for preventing loops. R1 is advertising the 10.1.1.0/24 route, and R2, R3, and R4 are RRs. Only a portion of the path is shown in this illustration, but all three scenarios of loop prevention are illustrated:

- R2 advertises the original advertisement to R1, which discards the route because its RID is in the originator ID.
- R4 advertises the NLRI to R2, and R2 discards the route because its RID is in the cluster list.
- R3 advertises the NLRI to R1, and R1 discards the route because its RID is in the originator ID.



**Figure 10-30** Loop Prevention with Originator ID and Cluster List Attributes



### Out-of-Band Route Reflectors

As shown earlier, BGP can establish multihop BGP sessions and does not change the next-hop path attribute when routes are advertised to iBGP neighbors. Some large network topologies use dedicated BGP routers for route reflection that are outside of the data path. These out-of-band route reflectors provide control plane programming for the BGP routers that are in the data path and only require sufficient memory and processing power for the BGP routing table. Out-of-band route reflectors should not use the next-hop-self; otherwise, it will place the RR into the data path.

Figure 10-31 demonstrates an out-of-band route reflector, where R2 peers with AS100, R4 peers with AS300, and R1 provides route reflection to R2, R3, and R4.

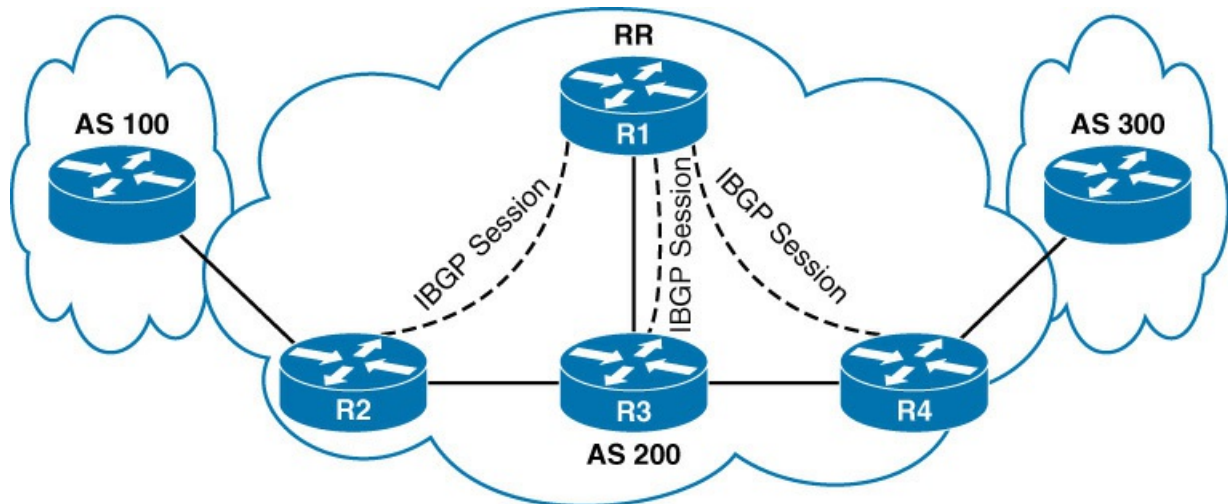


Figure 10-31 Out-of-Band Route Reflector Topology

### Confederations

RFC 3065 introduced the concept of *BGP confederations* as an alternative solution to the iBGP full-mesh scalability issues shown earlier. A confederation consists of sub-autonomous systems known as *member autonomous systems*, which combine into a larger autonomous system known as an *autonomous system confederation*. Member autonomous systems normally uses ASNs from the private ASN range (64512–65,534). eBGP peers from the confederation have no knowledge that they are peering with a confederation, and they reference the *confederation identifier* in their configuration.

Figure 10-32 demonstrates a BGP confederation with the confederation identifier of AS200. The member autonomous systems are AS65100, AS65200, and AS65300; and XR2 and R6 provide route reflection within their member autonomous system.

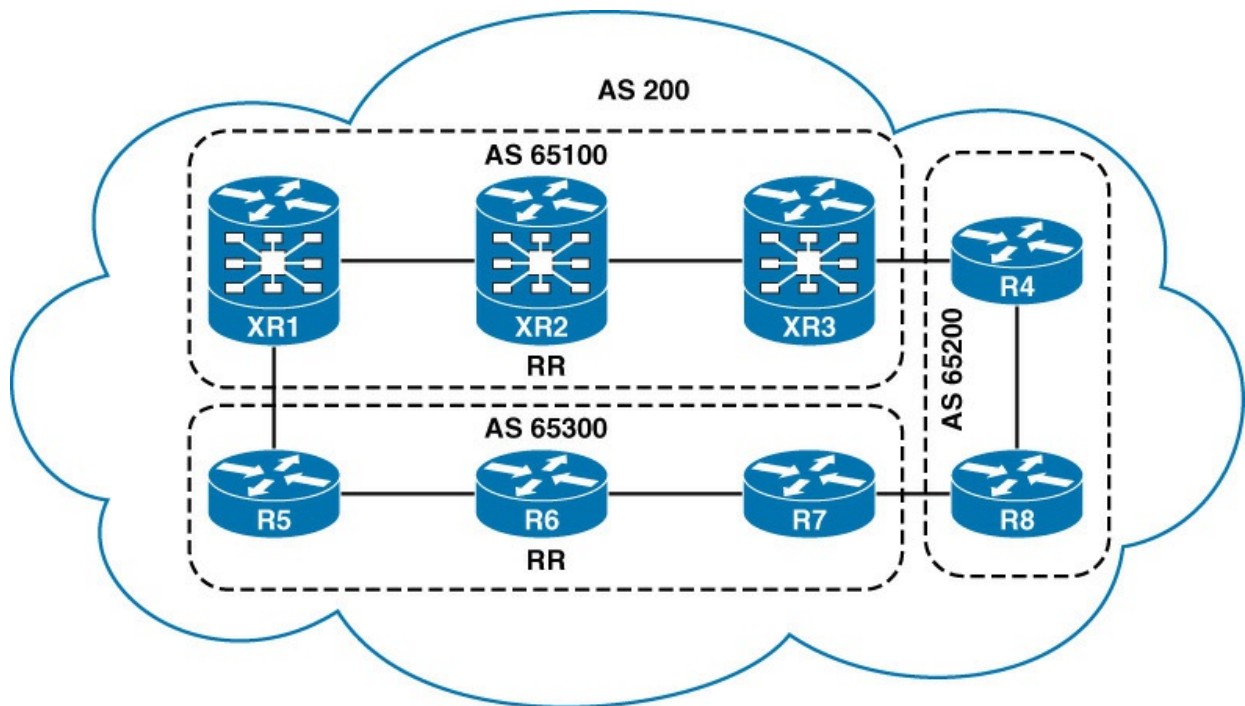


Figure 10-32 BGP Confederation Topology

Confederations share behaviors from both iBGP sessions and eBGP sessions. A list of changes is as follows:

- The AS\_Path attribute contains a subfield called AS\_CONFED\_SEQUENCE. The AS\_CONFED\_SEQUENCE is displayed in parentheses before any external ASNs in the AS\_Path. As the route passes from member autonomous system to member autonomous system, the AS\_CONFED\_SEQUENCE is appended to contain the member autonomous system ASNs. The AS\_CONFED\_SEQUENCE attribute is used to prevent loops, but is not used (counted) when choosing shortest AS\_Path.
- RRs can be used within the member autonomous system like normal iBGP peerings.
- The BGP MED attribute is transitive to all other member autonomous systems, but does not leave the confederation.
- The LOCAL\_PREF attribute is transitive to all other member autonomous systems, but does not leave the confederation.
- IOS XR nodes do not require a route policy when peering with a different member autonomous system even though the **remote-as** is different.
- The next-hop address for external confederation routes does not change as the route is exchanged between member autonomous system to member autonomous system.
- The AS\_CONFED\_SEQUENCE is removed from the AS\_Path when the route is advertised outside of the confederation.

The steps for configuring a BGP confederation are as follows:

### Step 1. Create the BGP routing process.

Initialize the BGP process with the global command **router bgp member-asn**.

### Step 2. Set the BGP confederation identifier.

Identify the BGP confederation identifier with the command **bgp confederation identifier as-number**, which is the ASN that will be viewed from routers outside of the confederation.

### Step 3. Identify peer member autonomous systems.

On routers that directly peer with another member autonomous system, identify the peering member autonomous system with the command **bgp confederation peers member-asn**.

### Step 4. Configure confederation members as normal.

Configure BGP confederation members as normal, the remaining configuration follows normal BGP configuration guidelines.

Figure 10-33 provides a sample confederation topology for AS200. XR2 and R3 provide the peering between the member autonomous systems. AS100 and AS300 are not aware that they are peering with a BGP confederation.

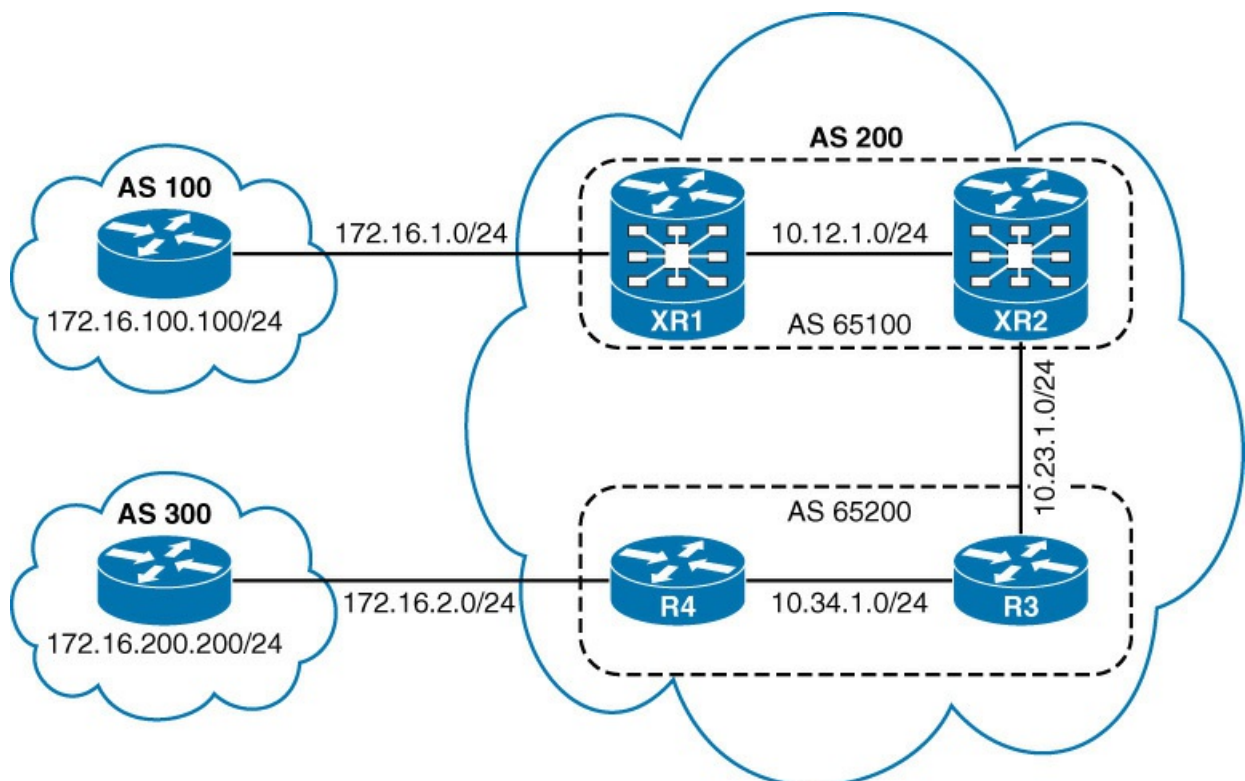


Figure 10-33 BGP Confederation Topology

Example 10-24 provides the relevant BGP configuration. Notice that XR1 and R4 do not use the **bgp confederation peers** configuration.

### Example 10-24 BGP Confederation Configuration

[Click here to view code image](#)

#### XR1

```
router bgp 65100
!
  bgp confederation identifier 200
  address-family ipv4 unicast
    network 10.12.1.0/24
    network 172.16.1.0/24
  !
  neighbor 10.12.1.2
    remote-as 65100
    address-family ipv4 unicast
    !
  neighbor 172.16.1.1
    remote-as 100
    address-family ipv4 unicast
    route-policy PASSALL in
    route-policy PASSALL out
```

#### XR2

```
router bgp 65100
  bgp confederation peers
    65200
  !
  bgp confederation identifier 200
  address-family ipv4 unicast
    network 10.12.1.0/24
    network 10.23.1.0/24
  !
  neighbor 10.12.1.1
    remote-as 65100
    address-family ipv4 unicast
    !
  !
  neighbor 10.23.1.3
    remote-as 65200
    address-family ipv4 unicast
```

### R3

```
router bgp 65200
  bgp confederation identifier 200
  bgp confederation peers 65100
  no bgp default ipv4-unicast
  neighbor 10.23.1.2 remote-as 65100
  neighbor 10.34.1.4 remote-as 65200
  !
  address-family ipv4
    network 10.23.1.0 mask 255.255.255.0
    network 10.34.1.0 mask 255.255.255.0
    neighbor 10.23.1.2 activate
    neighbor 10.34.1.4 activate
```

### R4

```
router bgp 65200
  bgp confederation identifier 200
  no bgp default ipv4-unicast
  neighbor 10.34.1.3 remote-as 65200
  neighbor 172.16.2.1 remote-as 300
  !
  address-family ipv4
    network 10.34.1.0 mask 255.255.255.0
    network 172.16.2.0 mask 255.255.255.0
    neighbor 10.34.1.3 activate
    neighbor 172.16.2.1 activate
```

**Example 10-25** displays the BGP table from AS100's perspective. Notice that XR1 removed the member ASNs from the route because it is advertised externally. AS100 is not aware that AS200 is a confederation.

### Example 10-25 AS100 BGP Table

[Click here to view code image](#)

```
AS100#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 10.12.1.0/24	172.16.1.2	0		0	200 i
*> 10.23.1.0/24	172.16.1.2			0	200 i
*> 10.34.1.0/24	172.16.1.2			0	200 i
* 172.16.1.0/24	172.16.1.2	0		0	200 i
*>	0.0.0.0	0		32768	i
*> 172.16.2.0/24	172.16.1.2			0	200 i
*> 172.16.100.100/32					
	0.0.0.0	0		32768	i
*> 172.16.200.200/32					
	0	200	300		i

**Example 10-26** displays XR'1 BGP, which participates in the member AS65100. Notice the next-hop address is not modified for the 172.16.200.200/32 route even though it passed through

multiple member autonomous systems. The AS\_CONFED\_SEQUENCE is listed in parentheses to indicate it passed through sub-AS65200 in the AS200 confederation.

### Example 10-26 R4's BGP Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast
! Output omitted for brevity
  Network          Next Hop          Metric LocPrf Weight Path
*> 10.12.1.0/24    0.0.0.0           0      32768 i
* i               10.12.1.2         0      100    0 i
*>i10.23.1.0/24    10.12.1.2         0      100    0 i
*>i10.34.1.0/24    10.23.1.3         0      100    0 (65200) i
*> 172.16.1.0/24  0.0.0.0           0      32768 i
*                172.16.1.1        0      0 100 i
*>i172.16.2.0/24  10.34.1.4         0      100    0 (65200) i
*> 172.16.100.100/32 172.16.1.1        0      0 100 i
*>i172.16.200.200/32 172.16.2.1        0      100    0 (65200) 300 i

Processed 7 prefixes, 9 paths
```

Example 10-27 displays the route information from the perspective of XR2. Notice that the route from within a confederation includes the option of *confed-internal* and *confed-external* for sources.

### Example 10-27 Confederation NLRI

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show bgp ipv4 unicast 172.16.100.100
! Output omitted for brevity
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
    10.23.1.3
  Path #1: Received by speaker 0
  Advertised to peers (in unique update groups):
    10.23.1.3
  100
  172.16.1.1 from 10.12.1.1 (192.168.1.1)
    Origin IGP, metric 0, localpref 100, valid, confed-internal, best, group-best
    Received Path ID 0, Local Path ID 1, version 8
```

```
RP/0/0/CPU0:XR2# show bgp ipv4 unicast 10.34.1.0/24
! Output omitted for brevity
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
    10.12.1.1
  Path #1: Received by speaker 0
  Advertised to peers (in unique update groups):
    10.12.1.1
    (65200)
    10.23.1.3 from 10.23.1.3 (192.168.3.3)
      Origin IGP, metric 0, localpref 100, valid, confed-external, best, group-best
      Received Path ID 0, Local Path ID 1, version 9
```

## FAILURE DETECTION

BGP relies on a stable network topology because of the size of a routing table. BGP keepalive and update messages ensure that the BGP neighbor is established. The default hold timer requires a packet be received every 3 minutes (180 seconds) to maintain the BGP session. The hold timer is negotiated when the BGP session first establishes.

By default, BGP sends a keepalive every 60 seconds to a BGP neighbor. The BGP keep-alive timer and hold timer can be set at the process or per neighbor session. The command **neighbor ip-address timers keepalive holdtime** [*minimum-holdtime*] configures the neighbor session timers on IOS nodes. The command **timers keepalive holdtime** [*minimum-holdtime*] configures the session timer for all routers on IOS and IOS XR nodes when applied to the global BGP configuration, or for a specific neighbor on IOS XR when applied to the neighbor configuration.

### Note

IOS and IOS XR support other technologies, like bidirectional forwarding detection (BFD), that are less burdensome to provide fast convergence. BFD is covered later in [Chapter 21, "High Availability."](#)

## SECURITY

eBGP sessions may be established with routers outside of an organizations control, which opens up potential security holes. The sections that follow describe the security mechanism that BGP includes to secure a router.

### eBGP Multihop

eBGP sessions use a TTL of 1 for the BGP packets as a security measure. This default behavior prevents multihop eBGP sessions that could be necessary when providers would like to peer via loopback addresses. This requires changing the TTL on BGP packets between the eBGP peers.

To change the TTL for multihop eBGP sessions, the command **neighbor ip-address ebgp-multihop hop-count** is applied under the neighbor session parameters for IOS nodes, and the command **ebgp-multihop hop-count** is used on the neighbor configuration for IOS XR nodes.

[Figure 10-34](#) demonstrates the eBGP multihop concept, as XR1 and R2 will peer via Loopback 0 interfaces. A static route is included in the configuration to provide connectivity between the routers Loopback 0 addresses.

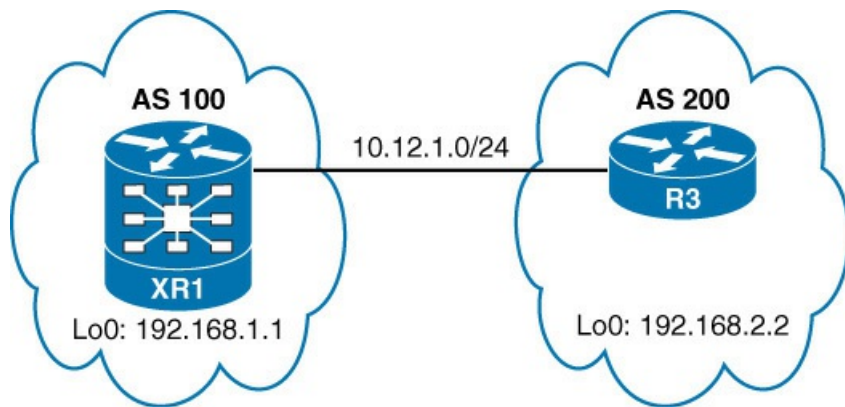


Figure 10-34 eBGP Multihop

Example 10-28 provides eBGP multihop configuration for XR1 and R2.

### Example 10-28 eBGP Multihop Configuration.

[Click here to view code image](#)

```

XR1
router static
  address-family ipv4 unicast
    192.168.2.2/32 10.12.1.2
  !
router bgp 100
  address-family ipv4 unicast
  !
  neighbor 192.168.2.2
    remote-as 200
    ebgp-multihop 2
    update-source Loopback0

  address-family ipv4 unicast
    route-policy PASSALL in
    route-policy PASSALL out

```

```

R2
ip route 192.168.1.1 255.255.255.255 10.12.1.1
!
router bgp 200
no bgp default ipv4-unicast
neighbor 192.168.1.1 remote-as 100
neighbor 192.168.1.1 ebgp-multihop 2
neighbor 192.168.1.1 update-source Loopback0
!
address-family ipv4
  network 2.2.2.2 mask 255.255.255.255
  neighbor 192.168.1.1 activate
exit-address-family

```

Confirmation of the TTL settings can be obtained with the **show bgp ipv4 unicast neighbors ip-address** command and by examining the *TTL* or *hops away*. Example 10-29 demonstrates how to view the BGP neighbor hop count after the session has established.



## Example 10-29 Confirmation of TTL

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast neighbors 192.168.2.2 | i neighbor
BGP neighbor is 192.168.2.2
  BGP neighbor version 2
  External BGP neighbor may be up to 2 hops away.
```

```
R2#show bgp ipv4 unicast neighbors 192.168.1.1 | i neighbor|TTL
BGP neighbor is 192.168.1.1, remote AS 100, external link
BGP table version 3, neighbor version 3/0
External BGP neighbor may be up to 2 hops away.
Minimum incoming TTL 0, Outgoing TTL 2
```

### TTL Security

Increasing the TTL for a BGP packet is simple but still provides a method for a hacker to send forged packets to a BGP router. [Figure 10-33](#) illustrates a hacker sending a forged packet to R3 from 12 hops away. Connectivity is maintained by incrementing the TTL high enough on the forged packet so that the TTL does not reach 0 before the packet reaches the router.

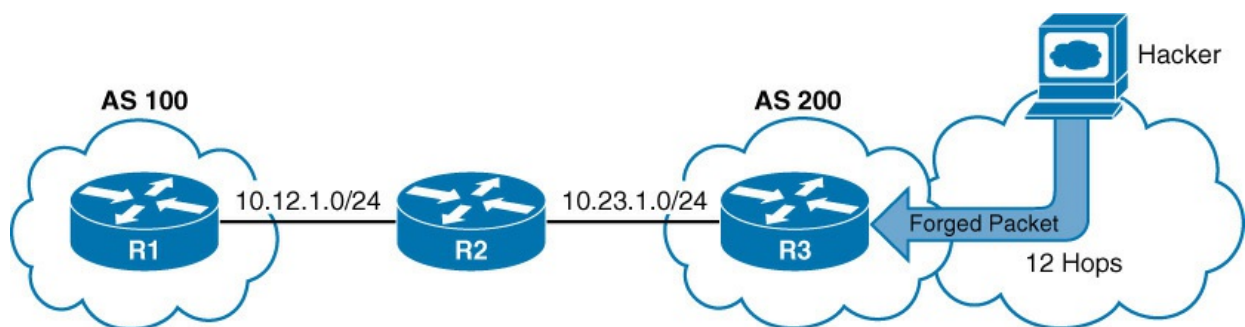


Figure 10-35 Nonsecure Multihop eBGP Session

RFC 5082 provides a security mechanism using TTL to ensure the hop count is exact. In [Figure 10-36](#), R1 is establishing a multihop eBGP session with R4.

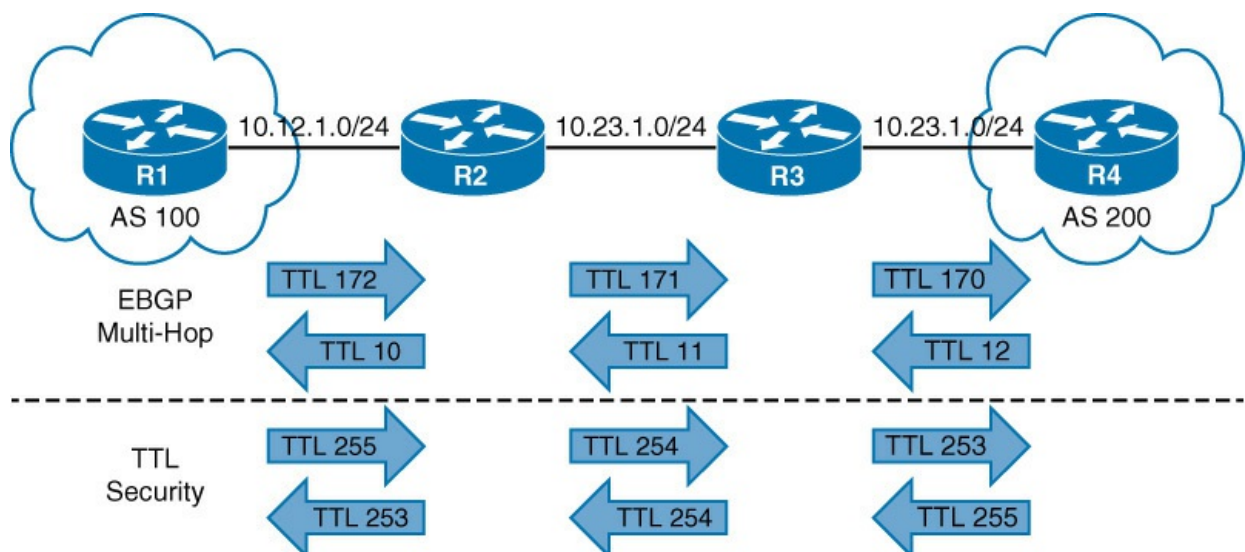


Figure 10-36 eBGP Multihop Versus TTL Security

Using regular eBGP multihop, R1 could configure eBGP multihop with 172, and the TTL would decrement to 170 upon reaching R4. R4 could configure eBGP multihop with a value of 12, and the TTL would decrement to 10 when reaching R1. eBGP multihop does not require either router to know how many hops are between them, only that the TTL be sufficient to reach the other router.

TTL security requires that each router know the number of hops to reach the peer. TTL security always sends packets with a TTL of 255. The receiving router subtracts the configured number of hops from 255 and verifies the TTL of the packet matches the expected TTL. If the TTL does not match, the packet is dropped. This tactic drastically shrinks the attack vector shown in [Figure 10-35](#).

In [Figure 10-36](#), R1 and R4 configure TTL security with a value of 3 and will accept only BGP packets with a TTL of 253.

Setting TTL security for an eBGP session uses the command **neighbor ip-address ttl-security hops hop-count** and is applied under the neighbor session parameters for IOS nodes. IOS XR only supports TTL security for directly connected neighbors with the command neighbor session configuration command **ttl-security**.

## Authentication

BGP supports authentication of BGP peers using a message digest 5 (MD5) authentication hash to prevent manipulation of BGP packets. BGP sessions that do not use authentication could potentially have spoofed updates inserted with false update messages.

To enable BGP authentication, the command **neighbor ip-address password password** is placed under the neighbor session parameters for IOS nodes, and the command **password clear password** is used on the neighbor configuration for IOS XR nodes.

To help illustrate the concept, XR1 and R2 will be configured with MD5 authentication using the password CISCO.

[Example 10-30](#) demonstrates BGP authentication configuration.

### Example 10-30 BGP Authentication Configuration

[Click here to view code image](#)

```
XR1
router bgp 100
 address-family ipv4 unicast
 !
 neighbor 10.12.1.2
  remote-as 200
  password CISCO
 address-family ipv4 unicast
```

## R2

```
router bgp 100
  no bgp default ipv4-unicast
  neighbor 10.12.1.1 remote-as 100
  neighbor 10.12.1.1 password CISCO
  !
  address-family ipv4
    neighbor 10.12.1.1 activate
```

### Note

There is not a command to verify BGP authentication between the peers outside of examining the configuration; however, a syslog message will appear if the passwords do not match, or if one neighbor is configured for authentication and the other is not.

## SUMMARY

BGP is a powerful path vector routing protocol that provides scalability and flexibility that cannot be compared to any other routing protocol. BGP uses TCP port 179 for establishing neighbors and allows TCP to handle fragmentation, sequencing, and reliability when communicating with BGP peers. Unlike most IGP, BGP is capable of forming multihop sessions that allow for a BGP router to reside outside of the data path.

Multiprotocol BGP extends functionality to multiple address families such as IPv4, IPv6, and Multiprotocol Label Switching (MPLS) virtual private networks (VPNs). Every address family maintains an independent table and routing policies. The NLRI is the routing update that consists of the network prefix, prefix length, and any BGP path attributes for that specific route.

BGP setup and deployment is simplified when considered as a modular configuration:

- **Session activation:** Identify the peering IP address, source interface, authentication, timers, and other values related to maintaining the BGP session.
- **Address family configuration:** Activate an address family for a specific BGP peer, modification of inbound/outbound route policy, and other settings specific to the advertisement/receipt of routes.

BGP uses three tables for maintaining the network prefix and path attributes:

- **Adj-RIB-In:** Contains the NLRIs in original form before inbound route policies are processed. This table results after all route policies are processed, to save memory.
- **Loc-RIB:** Contains all the NLRIs that originated locally or were received from other BGP peers. After NLRIs pass the validity and next-hop reachability check, the BGP best path algorithm selects the best NLRI for a specific prefix.
- **Adj-RIB-Out:** Contains the NLRIs after outbound route-policies have processed

After configuring the BGP **network** statement, BGP checks the global RIB for the network prefix before installing into the Loc-RIB. Depending on the entry in the global RIB, the following actions occur:

■ **Connected network:** The next-hop BGP attribute is set to 0.0.0.0, the origin attribute is set to *i* (IGP), and the BGP weight is set to 32,768.

■ **Static route or routing protocol:** The next-hop BGP attribute is set to the next-hop IP address in the RIB, the origin attribute is set to *i* (IGP), the BGP weight is set to 32,768, and the MED is set to the IGP metric.

iBGP routers do not advertise routes received from another iBGP peer unless route reflection is configured. Route reflectors use the originator ID and cluster list as a mechanism to prevent routing loops from iBGP neighbors. In additions to RRs, BGP confederations address iBGP scalability issues by breaking an autonomous system into smaller member autonomous systems.

The following behaviors differ on eBGP sessions when compared to iBGP sessions:

■ TTL on BGP packets is set to 1. BGP packets will drop in transit if a multihop BGP session is attempted. (TTL on iBGP packets is set to 255, which allows for multihop sessions.)

■ The advertising router modifies the BGP next hop to the IP address sourcing the BGP connection.

■ The advertising router prepends its ASN to the existing AS\_Path.

■ The receiving router verifies that the AS\_Path does not contain an ASN that matches the local router's. BGP discards the NLRI if it fails the AS\_Path loop-prevention check.

This chapter provides the basic fundamentals of the BGP routing protocol. In the following chapters, route maps, route policies, prefix lists, regular expressions, and access lists are explained to show you how BGP can incorporate different routing policies.

## REFERENCES IN THIS CHAPTER

Bates, T. and R. Chandra. RFC 1966, *BGP Route Reflection, An alternative to full mesh iBGP*, <http://www.ietf.org/rfc/rfc1966.txt>, June 1996

Bates, T., et al. RFC 2858, *Multiprotocol Extensions for BGP-4*, <http://www.ietf.org/rfc/rfc2858.txt>, June 2000

Traina, P., et al. RFC 3065, *Autonomous System Confederations for BGP*, <http://www.ietf.org/rfc/rfc3065.txt>, February 2001

Rekhter, Y., et al. RFC 4271, *A Border Gateway Protocol 4 (BGP-4)*, <http://www.ietf.org/rfc/rfc4271.txt>, January 2006

Vohra, Q. and E. Chen, RFC 4893, *BGP Support for Four-octet AS Number Space*, <http://www.ietf.org/rfc/rfc4893.txt>, May 2007

Gill, V., et al. RFC 5082, *The Generalized TTL Security Mechanism*, <http://www.ietf.org/rfc/rfc5082.txt>, October 2007

Cisco. Cisco IOS Software Configuration Guides. <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides. <http://www.cisco.com>

## **Part III: Advanced Routing Techniques**

## Chapter 11. Route Maps and Route Policy

This chapter covers the following topics:

- [Access control lists](#)
- [Prefix lists](#)
- [Regular expressions](#)
- [Route maps](#)
- [Routing policy language](#)

IOS route maps and IOS XR's routing policy language (RPL) provide the ability of filtering of routes, modifying route attributes, and routing behavior. These technologies use conditional match criteria to allow actions to occur based on route characteristics.

Before route maps and route policies are explained, the concepts involved with conditional matching using access control lists (ACLs), prefix lists, prefix sets, regular expressions (regex), AS\_Path (autonomous system path) ACLs, and AS path sets must be explained.

### ACCESS CONTROL LISTS

Originally, ACLs were intended to provide filtering of packets flowing into or out of a network interface, similar to the functionality of a basic firewall. Today, ACLs provide packet classification for a variety of features, such as quality of service (QoS), or for identifying networks within routing protocols.

ACLs are composed of access control entries (ACEs), which are entries in the ACL that identify the action to be taken (permit or deny) and the relevant packet classification for that action. Packet classification starts at the top (lowest sequence) and proceeds down (higher sequence) until a matching pattern is identified. Once a match is found, the appropriate action (permit or deny) is taken and processing stops. At the end of every ACLs is an implicit deny ACE, which denies all packets that did not match earlier in the ACL.

#### Note

ACE placement within an ACL is important and could lead to unintended consequences if they are out of order.

ACLs are classified into two categories:

- **Standard ACLs:** Define the packets based solely on the source network.
- **Extended ACLs:** Define the packet based on source, destination, protocol, port or combination of other packet attributes. This book is concerned with routing and limits the scope of ACLs to source, destination, and protocol.

Standard ACLs use the numbered entry 1–99, 1300–1999, or a named ACL. Extended ACLs use

the numbered entry 100–199, 2000–2699, or a named ACL. Named ACLs provide relevance to the functionality of the ACL, can be used with standard or extended ACLs, and are generally preferred.

Note

IOS XR does not use ACLs for matching traffic in a route policy, but its structure is explained for ACL-based forwarding (ABF) in [Chapter 12, “Advanced Route Manipulation.”](#)

### Standard ACLs

The process for defining a standard ACL for IOS nodes is as follows:

#### Step 1. Define the ACL.

Define the ACL with the command **ip access-list standard** {*acl-number* | *acl-name*} and the command-line interface (CLI) in ACL configuration mode.

#### Step 2. Configure ACEs.

Configure the specific ACE with the following command:

[Click here to view code image](#)

```
[sequence] {permit | deny} source source-wildcard
```

In lieu of using the *source* [*source-wildcard*], the keyword **any** replaces *0.0.0.0 0.0.0.0*, and usage of the **host** keyword refers to a /32 IP address so that the *source-wildcard* can be omitted.

The *source* and *source-wildcard* reflect a matching pattern for the network prefix that is being matched as described in the “[Wildcard Subnet Mask](#)” section in [Chapter 2, “IPv4 Addressing.”](#) [Table 11-1](#) provides sample entries that match for a specific IP address.

ACE Entry	Networks
permit any	Permits all networks
permit 172.16.0.0 0.0.255.255	Permits all networks in the 172.16.0.0 range
permit host 192.168.1.1	Permits only the 192.168.1.1/32 network

Table 11-1 Standard ACL to Network Entries

Note

The command **access-list** *acl-number* {**permit** | **deny**} *source* *source-wildcard* is considered legacy by some network engineers because it does not allow for the deletion of a specific ACE. Any iteration of this command with prefixed with the **no** keyword results in a deletion of the entire ACL and could lead to loss of access to the router if applied to an interface or other unforeseen conditions.

[Table 11-1](#) provides sample ACL entries from within the ACL configuration mode and specifies the networks that would match with a standard ACL.



Note

If a sequence is not provided, the sequence number will auto-increment by 10 based on the highest sequence number. The first entry will be 10. Sequencing allows the deletion of a specific ACE or the insertion of an ACE after the ACL is in use.

### Extended ACLs

The process for defining an extended ACL is as follows:

#### Step 1. Define the ACL.

Define the ACL with the command **ip access-list extended** {*acl-number* | *acl-name*} and place the CLI in ACL configuration mode.

#### Step 2. Configure ACEs.

Configure the specific ACE entry with the following command:

[Click here to view code image](#)

```
[sequence] {permit | deny} protocol source source-wildcard destination  
destination-wildcard
```

The behavior for selecting a network prefix with an extended ACL varies depending on whether the protocol is an interior gateway protocol (IGP) (Enhanced Interior Gateway Protocol [EIGRP], Open Shortest Path First [OSPF] Protocol, Intermediate System-to-Intermediate System [IS-IS] Protocol) or Border Gateway Protocol (BGP).

#### IGP Network Selection

When ACLs are used for the IGP network selection, the source fields of the ACL are used to identify the network, and the destination fields identify the smallest prefix length allowed in the network range. Table 11-2 provides sample ACL entries from within the ACL configuration mode and specifies the networks that would match with the extended ACL. Notice the subtle difference for the destination wildcard for the 172.16.0.0 network affects the actual network ranges that are permitted in the second and third row of the table.

ACE Entry	Networks
permit ip any any	Permits all networks
permit ip host 172.16.0.0 host 255.240.0.0	Permits all networks in the 172.16.0.0/12 range
permit ip host 172.16.0.0 host 255.255.0.0	Permits all networks in the 172.16.0.0/16 range
permit host 192.168.1.1	Permits only the 192.168.1.1/32 network

Table 11-2 Extended ACL for IGP Route Selection

#### BGP Network Selection

Extended ACLs react differently when matching BGP routes than when matching IGP routes. The source fields match against the network portion of the route, and the destination fields match against the network mask, as shown in Figure 11-1. Extended ACLs were originally the only match criteria used by IOS with BGP before the introduction of prefix lists.

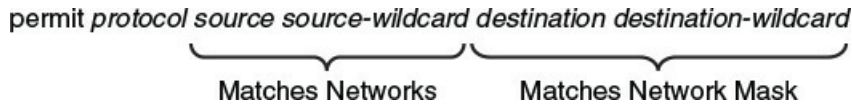


Figure 11-1 BGP Extended ACL Matches

Table 11-3 demonstrates the concept of the wildcard for the network and subnet mask.

Extended ACL	Matches These Networks
<code>permit ip 10.0.0.0 0.0.0.0 255.255.0.0 0.0.0.0</code>	Permits only the 10.0.0.0/16 network
<code>permit ip 10.0.0.0 0.0.255.0 255.255.255.0 0.0.0.0</code>	Permits any network any 10.0.x.0 network with a /24 prefix length
<code>permit ip 172.16.0.0 0.0.255.255 255.255.255.0 0.0.0.255</code>	Permits any 172.16.x.x network with a /24 – /32 prefix length
<code>permit ip 172.16.0.0 0.0.255.255 255.255.255.128 0.0.0.127</code>	Permits any 172.16.x.x network with a /25 – /32 prefix length

Table 11-3 Extended ACL for BGP Route Selection

## PREFIX MATCHING

Prefix lists and prefix sets provide another method of identifying networks in a routing protocol. They identify a specific IP address, network, or network range, and allow for the selection of multiple networks with a variety of prefix lengths by using a *prefix match specification*. This is preferred over the ACL network selection method.

The structure for a prefix match specification contains two parts: high order bit pattern and high-order bit count, which determines the high-order bits in the bit pattern that are to be matched. Some documentation refers to the high-order bit pattern as the *address* or *network*; and the high-order bit count as *length* or *mask length*.

In Figure 11-2, the prefix match specification has the high-order bit pattern of 192.168.0.0 and a high-order bit count of 16. The high-order bit pattern has been converted to binary to demonstrate where the high-order bit count lays. Because no additional matching length parameters are included, the high-order bit count is an exact match.

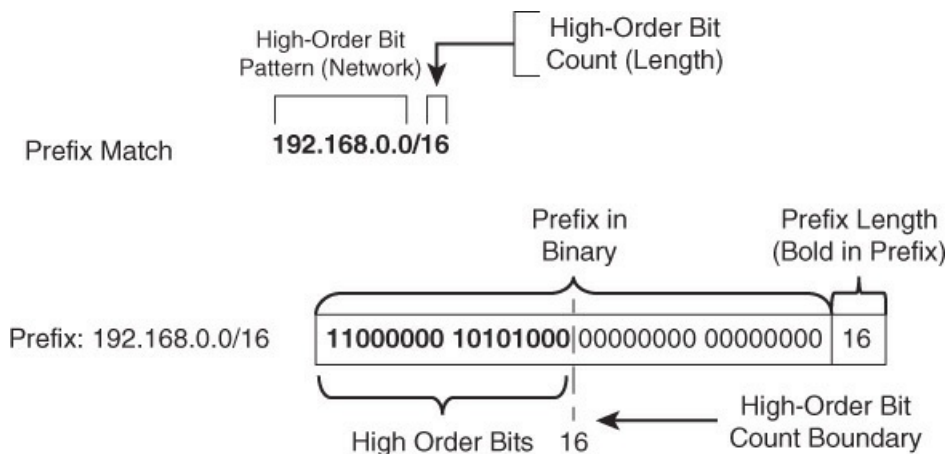


Figure 11-2 Basic Prefix Match Pattern

At this point, the prefix match specification logic looks identical to the functionality of an ACL. The true power and flexibility comes by using matching length parameters to identify multiple networks with specific prefix lengths with one statement. The matching length parameter options are as follows:

- **le** (less than or equal to <=)
- **ge** (greater than or equal to >=), or both

Figure 11-3 demonstrates the prefix match specification with a high-order bit pattern of 10.168.0.0, high-order bit count of 13, and matching length of the prefix must be greater than or equal to 24.

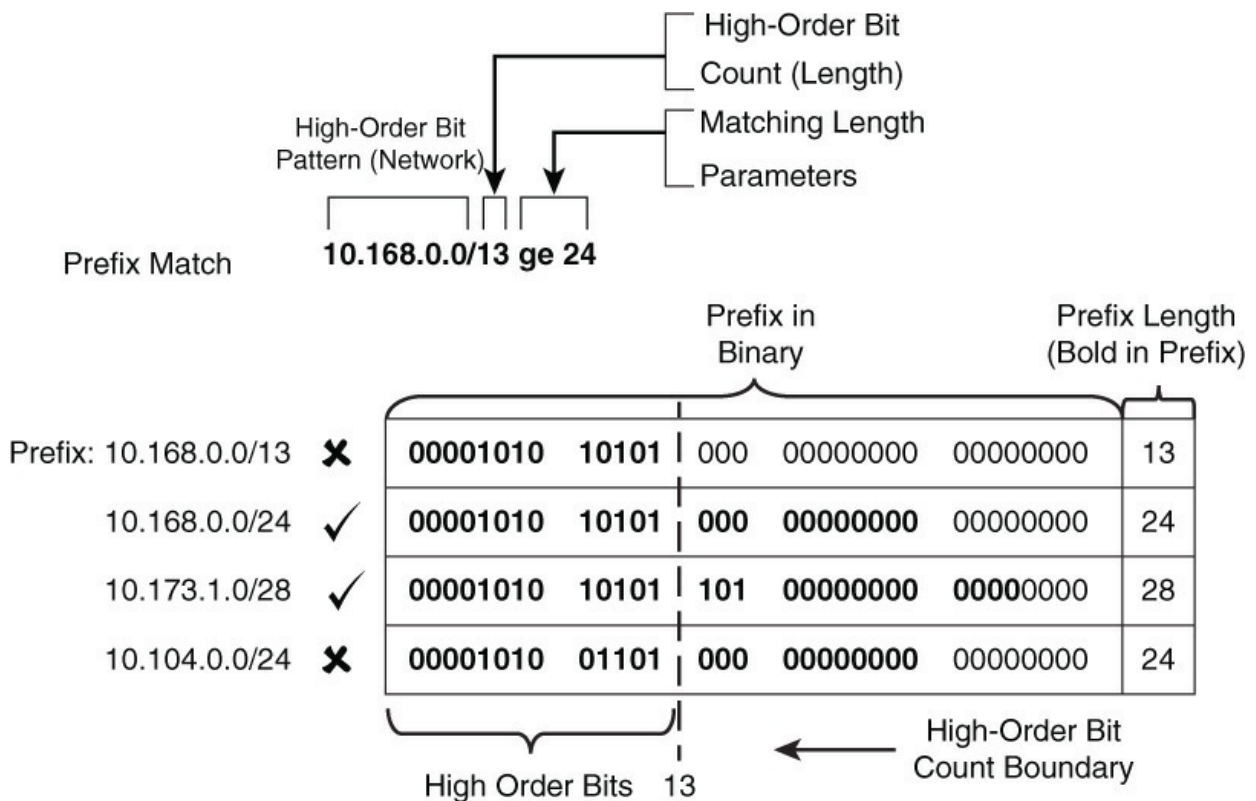


Figure 11-3 Prefix Match Pattern with Matching Length Parameters

The 10.168.0.0/13 prefix does not qualify because the prefix length is less than the minimum of 24 bits, whereas the 10.168.0.0/24 prefix does meet the matching length parameter. The 10.173.1.0/28 prefix qualifies because the first 13 bits match the high-order bit pattern and the prefix length is within the matching length parameter. The 10.104.0.0/24 prefix does not qualify because the high-order bit-pattern does not match within the high-order bit count.

Figure 11-4 demonstrates a prefix match specification with a high-order bit pattern of 10.0.0.0, high-order bit count of 8, and matching length must be between 22 and 26.

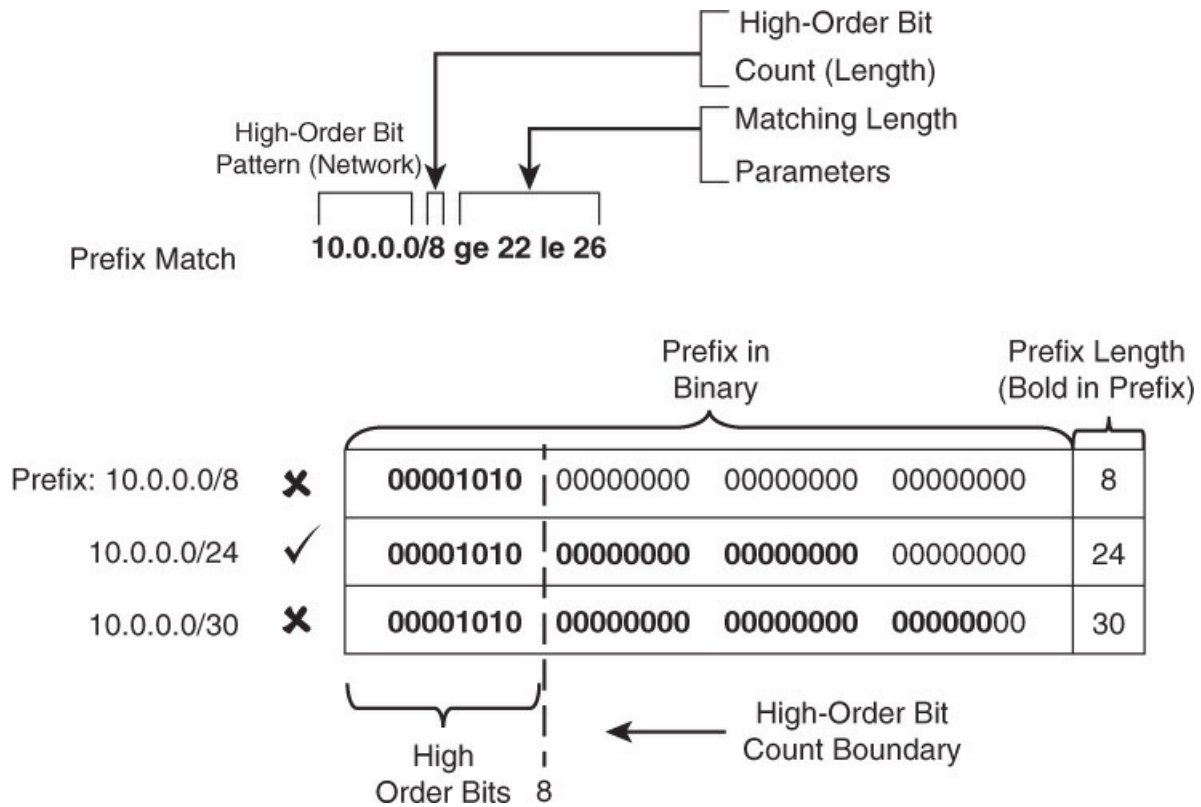


Figure 11-4 Prefix Match Pattern with Ineligible Matched Prefixes

The 10.0.0.0/8 prefix does not match because the prefix length is too short. The 10.0.0.0/24 qualifies because the bit pattern matches and because the prefix length is between 22 and 26. The 10.0.0.0/30 prefix does not match because the bit pattern is too long. Any prefix that starts with 10 in the first octet and has a prefix length between 22 and 26 will match.

Note

Matching to a specific prefix length that is higher than the high-order bit count requires that the *ge-value* and *le-value* match.

### Prefix Lists

Prefix lists can contain multiple prefix matching specification entries that contain a permit or deny action. Prefix lists process in sequential order in a top-down fashion, and the first prefix match processes with the appropriate permit or deny action.

IOS prefix lists are configured with the global configuration command **ip prefix-list** *prefix-list-name* [**seq** *sequence-number*] {**permit** | **deny**} *high-order-bit-pattern/high-order-bit-count* [**ge** *ge-value*] [**le** *le-value*].

IOS XR prefix lists are configured with the global configuration command **ipv4 prefix-list** *prefix-list-name* [**seq** *sequence-number*] {**permit** | **deny**} *high-order-bit-pattern/high-order-bit-count* [**ge** *ge-value*] [**le** *le-value*].

If a sequence is not provided, the sequence number will auto-increment by 5 based on the highest sequence number. The first entry will be 5. Sequencing allows the deletion of a specific entry. Because prefix lists cannot be resequenced, it is advisable to leave enough space for insertion of sequence numbers at a later time.

#### Note

IOS requires the *ge-value* must be greater than the high-order bit count, and that the *le-value* must be greater than or equal to the *ge-value* (*high-order bit count* < *ge-value* <= *le-value*). IOS XR requires the *ge-value* must be greater than or equal to the high-order bit count, and that the *le-value* must be greater than or equal to the *ge-value* (*high-order bit count* <= *ge-value* <= *le-value*).

**Example 11-1** provides a sample prefix list named RFC1918 for all of the networks in the RFC 1918 address range (10.0.0.0 – 10.255.255.255, 172.16.0.0 – 172.31.255.255, 192.168.0.0 – 192.168.255.255). The prefix list allows only /32 prefixes to exist in the 192.168.0.0 network range and not exist in any other network range in the prefix list.

Notice that sequence 5 permits all /32 prefixes in the 192.168.0.0/13 bit pattern, then sequence 10 denies all /32 prefixes in any bit pattern, and then sequence 15, 20, 25 permit routes in the appropriate network ranges. The sequence order is important for the first two entries to ensure that only /32 prefixes exist in the 192.168.0.0 in the prefix list.

### Example 11-1 IOS and IOS XR Sample Prefix List

[Click here to view code image](#)

#### IOS

```
ip prefix-list RFC1918 seq 5 permit 192.168.0.0/13 ge 32
ip prefix-list RFC1918 seq 10 deny 0.0.0.0/0 ge 32
ip prefix-list RFC1918 seq 15 permit 10.0.0.0/7 ge 8
ip prefix-list RFC1918 seq 20 permit 172.16.0.0/11 ge 12
ip prefix-list RFC1918 seq 25 permit 192.168.0.0/15 ge 16
```

#### IOS XR

```
ipv4 prefix-list RFC1918 seq 5 permit 192.168.0.0/13 ge 32
ipv4 prefix-list RFC1918 seq 10 deny 0.0.0.0/0 ge 32
ipv4 prefix-list RFC1918 seq 15 permit 10.0.0.0/7 ge 8
ipv4 prefix-list RFC1918 seq 20 permit 172.16.0.0/12 ge 12
ipv4 prefix-list RFC1918 seq 25 permit 192.168.0.0/16 ge 16
```

#### Note

IOS nodes can associate a prefix list directly to a BGP neighbor for filtering of routes with the command **neighbor ip-address prefix-list prefix-list-name {in | out}**.

## Prefix Sets

IOS XR prefix sets are a comma delimited set of prefix matching specifications. Prefix sets are a component of IOS XR's RPL and do not allow the option for denying a prefix matching specification, and once a prefix matches an entry, searching in the prefix set stops. Prefix set configuration follows this process:

### Step 1. Define the prefix set.

Define the prefix-set with the command **prefix-set prefix-set-name**.

### Step 2. Enter the prefix set entries.

Configure each specific prefix entry with the command *high-order-bit-pattern* [/high-order-bit-count] [**ge** *ge-value*] [**le** *le-value*]. In lieu of setting the *ge-value* and *le-value* to the same setting, you can use the option of **eq** *eq-value*.

Step 2 is repeated for multiple entries, with every entry except the last one delimited with a comma.

A specific host entry can be specified by only entering a complete high-order bit pattern.

### Step 3. Exit the prefix set.

Exit the prefix set with the command **end-set**.

[Example 11-2](#) provides a sample prefix set that matches any networks within the RFC 1918 address space.

#### Example 11-2 IOS XR Sample Prefix Set

[Click here to view code image](#)

```
IOS XR
prefix-set PREFIX-SET-RFC1918
  10.0.0.0/8 ge 8,
  172.16.0.0/12 ge 12,
  192.168.0.0/16 ge 16
end-set
```

[Example 11-3](#) provides a sample prefix set that identifies two web servers with the IP address of 192.168.1.1 and 192.168.2.2. Notice that a high-order bit count length was not necessary in the configuration.

#### Example 11-3 IOS XR Sample Prefix Set with Host Entries

[Click here to view code image](#)

```
IOS XR
prefix-set WEBSERVERS
  192.168.1.1,
  192.168.2.2
end-set
```

### Regular Expressions

There may be times when filtering multiple entries based on a data pattern is required. A common use case involves the parsing of the AS\_Path for BGP. To parse through the large number of available autonomous system numbers (ASNs) (4,294,967,295), regular expressions (regex) are used. Regular expressions are based on query modifiers to select the appropriate content.

[Table 11-4](#) provides a brief list and description of the common regex query modifiers.

Modifier	Description
_ (Underscore)	Matches a space
^ (Caret)	Indicates the start of the string
\$ (Dollar sign)	Indicates the end of the string
[] (Brackets)	Matches a single character or nesting within a range
- (Hyphen)	Indicates a range of numbers in brackets
[^] (Caret in brackets)	Excludes the characters listed in brackets
() (Parentheses)	Used for nesting of search patterns
(Pipe)	Provides <i>or</i> functionality to the query
. (Period)	Matches a single character, including a space
* (Asterisk)	Matches 0 or more characters or patterns
+ (Plus sign)	1 or more instances of the character or pattern
? (Question mark)	Matches 1 or no instances of the character or pattern

Table 11-4 *Regex Query Modifiers*

Note

The characters `^$*+()[]?` are special control characters that cannot be used without using the backslash (`\`) escape character. For example, to match on the `*` in the output, you would use the `\*` syntax.

The BGP table can be parsed with regex using the command **show bgp address-family address-family-modifier **regexp** regex-pattern** on IOS and IOS XR nodes. Figure 11-5 provides a reference topology, with Example 11-4 providing a reference BGP table so that the various regex query modifiers can be demonstrated. The following section provides various scenarios to explain the query modifier and highlight the portion of the AS\_Path that matches the query.

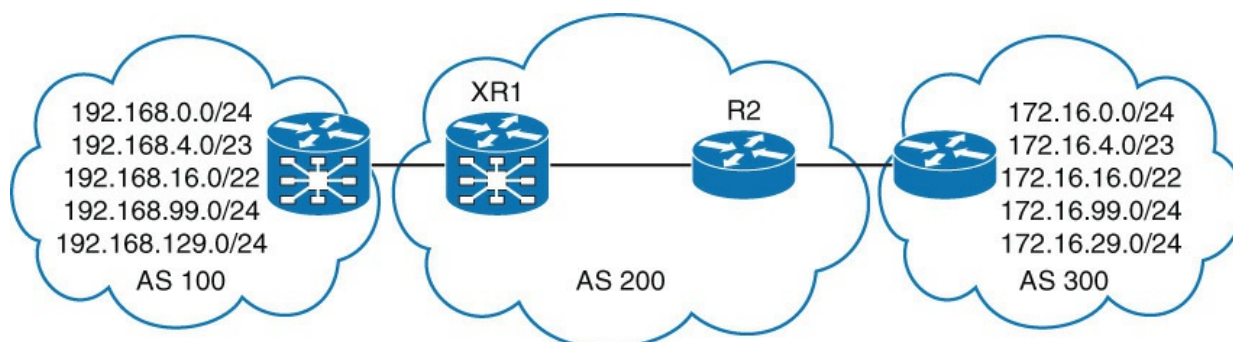


Figure 11-5 *BGP Regex Reference Topology*

**Example 11-4** *BGP Table for Regex Queries*

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/24	192.168.200.3	0		0 300 80 90	21003 2100 i
*> 172.16.4.0/23	192.168.200.3	0		0 300 1080 1090 1100	1110 i
*> 172.16.16.0/22	192.168.200.3	0		0 300 11234 21234 31234	i
*> 172.16.99.0/24	192.168.200.3	0		0 300 40	i
*> 172.16.129.0/24	192.168.200.3	0		0 300 10010 300 30010 30050	i
*>i192.168.0.0	10.12.1.1	0	100	0 100 80 90 21003 2100	i
*>i192.168.4.0/23	10.12.1.1	0	100	0 100 1080 1090 1100 1110	i
*>i192.168.16.0/22	10.12.1.1	0	100	0 100 11234 21234 31234	i
*>i192.168.99.0	10.12.1.1	0	100	0 100 40	i
*>i192.168.129.0	10.12.1.1	0	100	0 100 10010 300 30010 30050	i

#### Note

The AS Path for the prefix 172.16.129.0/24 has AS300 twice nonconsecutively for a specific purpose. This normally would not be seen in a real network, as it would indicate a routing loop.

## \_(Underscore)

**Query modifier function:** Matches a space

**Scenario:** Only display autonomous systems that passed through AS100. The first assumption is that the syntax **show bgp ipv4 unicast regex 100**, as shown in [Example 11-5](#), would be ideal. The regex query includes the following unwanted ASNs: 1100, 2100, 21003, and 10010.

### Example 11-5 BGP Regex Query for AS100

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast regex 100
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/24	192.168.200.3	0		0 300 80 90 21003 455	i
*> 172.16.4.0/23	192.168.200.3	0		0 300 878 1190 1100 1010	i
*> 172.16.129.0/24	192.168.200.3	0		0 300 10010 300 1010 40 50	i
*>i192.168.0.0	10.12.1.1	0	100	0 100 80 90 21003 455	i
*>i192.168.4.0/23	10.12.1.1	0	100	0 100 878 1190 1100 1010	i
*>i192.168.16.0/22	10.12.1.1	0	100	0 100 779 21234 45	i
*>i192.168.99.0	10.12.1.1	0	100	0 100 145 40	i
*>i192.168.129.0	10.12.1.1	0	100	0 100 10010 300 1010 40 50	i

[Example 11-6](#) uses the underscore (\_) before the ASN to imply a space left of AS100 to remove the unwanted ASNs. Unfortunately, this regex query includes the following unwanted ASN: 10010.

### Example 11-6 BGP Regex Query for AS100

[Click here to view code image](#)



```
R2#show bgp ipv4 unicast regexp _100
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.129.0/24	192.168.200.3	0		0 300	10010 300 1010 40 50 i
*>i192.168.0.0	10.12.1.1	0	100	0 100	80 90 21003 455 i
*>i192.168.4.0/23	10.12.1.1	0	100	0 100	878 1190 1100 1010 i
*>i192.168.16.0/22	10.12.1.1	0	100	0 100	779 21234 45 i
*>i192.168.99.0	10.12.1.1	0	100	0 100	145 40 i
*>i192.168.129.0	10.12.1.1	0	100	0 100	10010 300 1010 40 50 i

Example 11-7 provides the final query by using underscore ( ) before and after the ASN (100) to finalize the query for route that pass through AS100.

### Example 11-7 BGP Regex Query for AS\_100\_

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast regexp _100_
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i192.168.0.0	10.12.1.1	0	100	0 100	80 90 21003 455 i
*>i192.168.4.0/23	10.12.1.1	0	100	0 100	878 1190 1100 1010 i
*>i192.168.16.0/22	10.12.1.1	0	100	0 100	779 21234 45 i
*>i192.168.99.0	10.12.1.1	0	100	0 100	145 40 i
*>i192.168.129.0	10.12.1.1	0	100	0 100	10010 300 1010 40 50 i

#### ^ (Caret)

**Query modifier function:** Indicates the start of the string

**Scenario:** Only display routes that were advertised from AS300. At first glance, the command `show bgp ipv4 unicast regex _300_` might be acceptable for use, but in Example 11-8 the route 192.168.129.0/24 is also included.

### Example 11-8 BGP Regex Query for AS300

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast regexp _300_
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/24	192.168.200.3	0		0 300	80 90 21003 455 i
*> 172.16.4.0/23	192.168.200.3	0		0 300	878 1190 1100 1010 i
*> 172.16.16.0/22	192.168.200.3	0		0 300	779 21234 45 i
*> 172.16.99.0/24	192.168.200.3	0		0 300	145 40 i
*> 172.16.129.0/24	192.168.200.3	0		0 300	10010 300 1010 40 50 i
*>i192.168.129.0	10.12.1.1	.0	100	0 100	10010 300 1010 40 50 i

Because AS300 is directly connected, it would be more efficient to ensure that AS300 was the first AS listed. Example 11-9 shows the ^ in the regex pattern.

### Example 11-9 BGP Regex Query with Caret

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast regexp ^300_  
! Output omitted for brevity  
Network Next Hop Metric LocPrf Weight Path  
*> 172.16.0.0/24 192.168.200.3 0 0 300 80 90 21003 455 i  
*> 172.16.4.0/23 192.168.200.3 0 0 300 878 1190 1100 1010 i  
*> 172.16.16.0/22 192.168.200.3 0 0 300 779 21234 45 i  
*> 172.16.99.0/24 192.168.200.3 0 0 300 145 40 i  
*> 172.16.129.0/24 192.168.200.3 0 0 300 10010 300 1010 40 50 i
```

## \$ (Dollar Sign)

**Query modifier function:** Indicates the end of the string

**Scenario:** Only display routes that originated in AS 40. In [Example 11-10](#), the regex pattern `_40_` was used. Unfortunately, this also includes routes that originated in AS 50.

### Example 11-10 BGP Regex Query with AS 40

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast regexp _40_  
! Output omitted for brevity  
Network Next Hop Metric LocPrf Weight Path  
*> 172.16.99.0/24 192.168.200.3 0 0 300 145 40 i  
*> 172.16.129.0/24 192.168.200.3 0 0 300 10010 300 1010 40 50 i  
*>i192.168.99.0 10.12.1.1 0 100 0 100 145 40 i  
*>i192.168.129.0 10.12.1.1 0 100 0 100 10010 300 1010 40 50 i
```

[Example 11-11](#) provides the solution using the \$ for the regex the pattern `_40$`.

### Example 11-11 BGP Regex Query with Dollar Sign

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast regexp _40$  
! Output omitted for brevity  
Network Next Hop Metric LocPrf Weight Path  
*> 172.16.99.0/24 192.168.200.3 0 0 300 145 40 i  
*>i192.168.99.0 10.12.1.1 0 100 0 100 145 40 i
```

## [ ] (Brackets)

**Query modifier function:** Matches a single character or nesting within a range.

**Scenario:** Only display routes with an AS that contains 11 or 14 in it. The regex filter `1[14]` can be used as shown in [Example 11-12](#).

### Example 11-12 BGP Regex Query with Brackets

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast regexp 1[14]
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.4.0/23	192.168.200.3	0		0 300 878	1100 1010 i
*> 172.16.99.0/24	192.168.200.3	0		0 300 145	40 i
*>i192.168.4.0/23	10.12.1.1	0	100	0 100 878	1190 1100 1010 i
*>i192.168.99.0	10.12.1.1	0	100	0 100 145	40 i

## - (Hyphen)

**Query modifier function:** Indicates a range of numbers in brackets.

**Scenario:** Display routes only if the last two digits of the autonomous system are 40, 50, 60, 70, or 80.

Example 11-13 uses the regex query `[4-8]0_`.

## Example 11-13 BGP Regex Query with Hyphen

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast regexp [4-8]0_
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/24	192.168.200.3	0		0 300 80 90	21003 455 i
*> 172.16.99.0/24	192.168.200.3	0		0 300 145	40 i
*> 172.16.129.0/24	192.168.200.3	0		0 300 10010 300 1010	40 50 i
*>i192.168.0.0	10.12.1.1	0	100	0 100 80 90	21003 455 i
*>i192.168.99.0	10.12.1.1	0	100	0 100 145	40 i
*>i192.168.129.0	10.12.1.1	0	100	0 100 10010 300 1010	40 50 i

## [^] (Caret in Brackets)

**Query modifier function:** Excludes the character listed in brackets

**Scenario:** Only display routes where the second autonomous system from AS100 or AS300 does not start with 3-8. The first component of the regex query is to restrict the autonomous system to the AS100 or 300 with the regex query `^[13]00_`, and the second component is to filter out autonomous system starting with 3-8 with the regex filter `_[^3-8]`. The complete regex query is `^[13]00_[^3-8]`, as shown in Example 11-14.

## Example 11-14 BGP Regex Query with Caret in Brackets

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast regexp ^[13]00_[^3-8]
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.99.0/24	192.168.200.3	0		0 300 145	40 i
*> 172.16.129.0/24	192.168.200.3	0		0 300 10010 300 1010	40 50 i
*>i192.168.99.0	10.12.1.1	0	100	0 100 145	40 i
*>i192.168.129.0	10.12.1.1	0	100	0 100 10010 300 1010	40 50 i

## ( ) (Parentheses and | Pipe)

**Query modifier function:** Nesting of search patterns and provides *or* functionality

**Scenario:** Only display routes where the AS\_Path ends with AS 40 or 45 in it. [Example 11-15](#) demonstrates the regex filter `_4(5|0)$`.

### Example 11-15 BGP Regex Query with Parentheses

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast regexp _4(5|0)$
! Output omitted for brevity
      Network          Next Hop      Metric LocPrf Weight Path
*> 172.16.16.0/22     192.168.200.3    0           0 300 779 21234 45 i
*> 172.16.99.0/24     192.168.200.3    0           0 300 145 40 i
*>i192.168.16.0/22    10.12.1.1        0    100       0 100 779 21234 45 i
*>i192.168.99.0      10.12.1.1        0    100       0 100 145 40 i
```

## . (Period)

**Query modifier function:** Matches a single character, including a space

**Scenario:** Only display routes with an originating autonomous system of 1–99. In [Example 11-16](#), the regex query `__.$` requires a space, and then any character after that (including other spaces).

### Example 11-16 BGP Regex Query with Period

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast regexp __.$
! Output omitted for brevity
      Network          Next Hop      Metric LocPrf Weight Path
*> 172.16.16.0/22     192.168.200.3    0           0 300 779 21234 45 i
*> 172.16.99.0/24     192.168.200.3    0           0 300 145 40 i
*> 172.16.129.0/24    192.168.200.3    0           0 300 10010 300 1010 40 50 i
*>i192.168.16.0/22    10.12.1.1        0    100       0 100 779 21234 45 i
*>i192.168.99.0      10.12.1.1        0    100       0 100 145 40 i
*>i192.168.129.0     10.12.1.1        0    100       0 100 10010 300 1010 40 50 i
```

## + (Plus Sign)

**Query modifier function:** One or more instances of the character or pattern

**Scenario:** Only display routes where they contain at least one 10 in the AS\_Path, but the pattern 100 should not be used in matching. When building this regex expression, the first portion is building the matching pattern of `(10)+`, and then add the restriction portion of the query of `[^(100)]`. The combined regex pattern is `(10)+[^(100)]`, as shown in [Example 11-17](#).

### Example 11-17 BGP Regex Query with Plus Sign

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast regexp (10)+[^(100)]
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.4.0/23	192.168.200.3	0		0 300	1080 1090 1100 1110 i
*> 172.16.129.0/24	192.168.200.3	0		0 300	10010 300 30010 30050 i
*>i192.168.4.0/23	10.12.1.1	0	100	0 100	1080 1090 1100 1110 i
*>i192.168.129.0	10.12.1.1	0	100	0 100	10010 300 30010 30050 i

### ? (Question Mark)

**Query modifier function:** Matches one or no instances of the character or pattern

**Scenario:** Only display routes from the neighboring autonomous system or its directly connected autonomous system (that is, restrict to two autonomous systems away). This query is more complicated and requires us to define an initial query for identifying the autonomous system, which will be `[0-9]+`. The second component will include the space, and an optional second autonomous system. The `?` limits the autonomous system match to one or two autonomous systems, as shown in [Example 11-18](#).

Note

The `Ctrl+V` escape sequence must be used before entering the `?`.

### Example 11-18 BGP Regex Query with Dollar Sign

[Click here to view code image](#)

```
R1#show bgp ipv4 unicast regexp ^[0-9]+ ([0-9]+)?$
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.99.0/24	192.168.200.3	0		0 300	40 i
*>i192.168.99.0	10.12.1.1	0	100	0 100	40 i

### \* (Asterisk)

**Query modifier function:** Matches zero or more characters or patterns

**Scenario:** Display all routes from any autonomous system. This might seem like a useless task, but may be a valid requirement when using `AS_Path` access lists, which are explained later in this chapter. [Example 11-19](#) shows the regex query.

### Example 11-19 BGP Regex Query with Asterisk

[Click here to view code image](#)

```
R1#show bgp ipv4 unicast regexp .*
```

```
! Output omitted for brevity
```

```
Network          Next Hop      Metric LocPrf Weight Path
*> 172.16.0.0/24  192.168.200.3  0           0   300 80 90 21003 2100 i
*> 172.16.4.0/23  192.168.200.3  0           0   300 1080 1090 1100 1110 i
*> 172.16.16.0/22 192.168.200.3  0           0   300 11234 21234 31234 i
*> 172.16.99.0/24 192.168.200.3  0           0   300 40 i
*> 172.16.129.0/24 192.168.200.3  0           0   300 10010 300 30010 30050 i
*>i192.168.0.0    10.12.1.1      0   100    0   100 80 90 21003 2100 i
*>i192.168.4.0/23 10.12.1.1      0   100    0   100 1080 1090 1100 1110 i
*>i192.168.16.0/22 10.12.1.1      0   100    0   100 11234 21234 31234 i
*>i192.168.99.0   10.12.1.1      0   100    0   100 40 i
*>i192.168.129.0  10.12.1.1      0   100    0   100 10010 300 30010 30050 i
```

Note

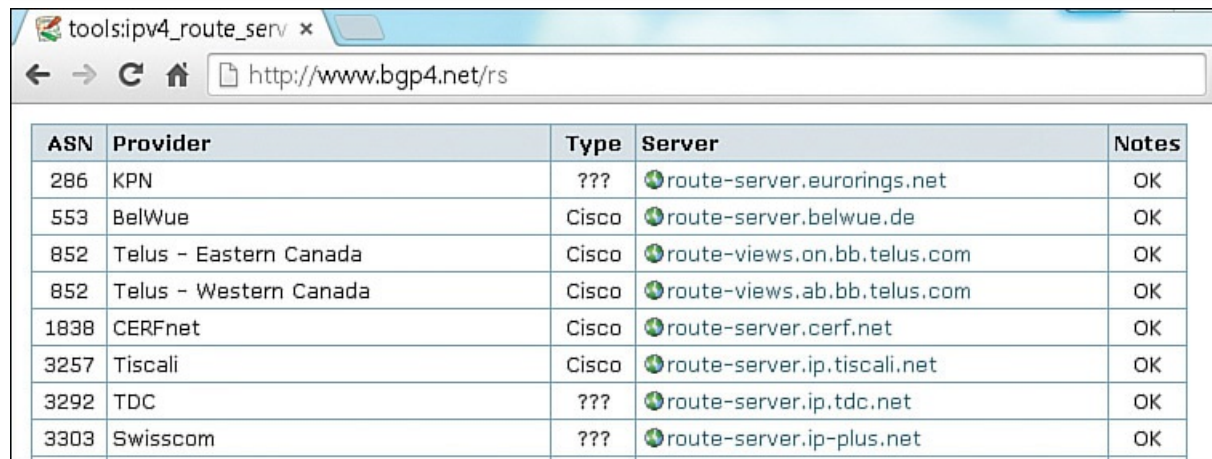
Regex is not limited to only BGP and can be used for parsing CLI output, too.

## Looking Glass and Route Servers

Hands-on experience is helpful when learning technologies such as regex. There are public devices called *looking glass* or *route servers* that allow users to log in and view BGP tables. Most of these devices are Cisco routers, but there are other vendors as well. These servers enable network engineers to see whether they are advertising their routes to the Internet, as they had intended, and provide a great method to try out regular expressions on the Internet BGP table. A quick search on the Internet will provide website listings of looking glass and route servers.

Note

The authors suggest <http://www.bgp4.net> and <http://www.traceroute.org>, as shown in Figure 11-6.



ASN	Provider	Type	Server	Notes
286	KPN	???	route-server.eurorings.net	OK
553	BelWue	Cisco	route-server.belwue.de	OK
852	Telus - Eastern Canada	Cisco	route-views.on.bb.telus.com	OK
852	Telus - Western Canada	Cisco	route-views.ab.bb.telus.com	OK
1838	CERFnet	Cisco	route-server.cerf.net	OK
3257	Tiscali	Cisco	route-server.ip.tiscali.net	OK
3292	TDC	???	route-server.ip.tdc.net	OK
3303	Swisscom	???	route-server.ip-plus.net	OK

Figure 11-6 BGP4.NET Listing of Route Servers

Figure 11-7 shows sample output for a public route server.

```

route-views.routeviews.org is now using AAA for logins.  Login with
username "rviews".  See http://routeviews.org/aaa.html

*****

User Access Verification

Username: rviews
route-views>show bgp ipv4 unicast regexp _109$
BGP table version is 1179859071, local router ID is 128.223.51.103
Status codes: s suppressed, d damped, h history, * valid, > best, i - inter
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop          Metric LocPrf Weight Path
*  12.5.186.0/23    4.69.184.193      0           0 3356 109 i
*                   66.185.128.48     2           0 1668 3356 109
*                   207.172.6.20      0           0 6079 3356 109
*                   207.172.6.1       0           0 6079 3356 109

```

Figure 11-7 Public Looking Glass Server

## AS\_PATH ACCESS LIST

Selecting routes by using the AS\_Path in a route map requires the definition of an AS\_Path ACL. Processing is performed in a sequential top-down order, and the first qualifying match will process against the appropriate permit or deny action. An implicit deny exists at the end of the AS\_Path ACL. IOS supports up to 500 AS\_Path ACLs and uses the command **ip as-path access-list** *acl-number* {**deny** | **permit**} *regex-query* for creating the AS\_Path ACL.

**Example 11-20** provides a sample AS\_Path access list that matches against any local internal BGP (iBGP) prefix or any prefix that passes through AS300.

### Example 11-20 AS\_Path Access List Configuration

[Click here to view code image](#)

```

ip as-path access-list 1 permit _300_
ip as-path access-list 1 permit ^$

```

**Example 11-21** provides samples of more complicated versions of the AS\_Path access list that matches any prefix that passed through the 16-bit private ASN range (64,512 – 65,534). The first iteration uses multiple ACE entries, and the second interace takes advantage of brackets and braces.

### Example 11-21 AS\_Path Access List for Private ASN

[Click here to view code image](#)

```

ip as-path access-list 2 permit _(6451[2-9])_
ip as-path access-list 2 permit _(645[2-9][0-9])_
ip as-path access-list 2 permit _(64[6-9][0-9][0-9])_
ip as-path access-list 2 permit _(65[0-4][0-9][0-9])_
ip as-path access-list 2 permit _(655[0-2][0-9])_
ip as-path access-list 2 permit _(6553[0-6])_

```

```
! The following two lines are entered into the CLI as one entry
ip as-path access-list 3 permit _(64(5(1[2-9]|[2-9][0-9])|([6-9][0-9]
[0-9]))|65([0-4][0-9][0-9]|5([0-2][0-9]|3[0-4])))_
```

#### Note

IOS nodes can associate a AS\_Path ACL directly to a BGP neighbor for filtering of routes with the command **neighbor ip-address filter-list acl-number {in | out}**.

## IOS XR AS\_PATH SELECTION OPTIONS

IOS XR route policies and AS path sets allow the selection of prefixes using regular expressions with the command structure of **ios-regex 'regex-query'**, providing identical output on IOS routers. IOS XR provides additional methods for simplifying AS\_Path queries, as explained in this section using [Example 11-22](#) as a reference BGP table before the query is performed.

### Example 11-22 Reference BGP Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.17.100.0/24	172.17.0.2	0		0 200 300 400	i
*> 172.17.200.0/24	172.17.0.2	0		0 200 300 300 300 300	i
*> 172.18.100.0/24	172.18.0.2	0		0 500 550 200	i
*> 172.19.100.0/24	172.19.0.2	0		0 600 200 600	i
*>i192.168.2.2/32	172.16.0.2	0	100	0	i

```
Processed 5 prefixes, 5 paths
```

#### is-local

The **is-local** option provides a method of selecting routes that come from the local AS from an iBGP peer. iBGP prefixes have a blank AS\_Path entry and this function is equivalent to the IOS regex query of **^\$**. [Example 11-23](#) provides the reference BGP table when using the IOS XR **is-local** AS\_Path query.

### Example 11-23 BGP Prefix Selection with is-local

[Click here to view code image](#)

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i192.168.2.2/32	172.16.0.2	0	100	0	i

```
Processed 1 prefixes, 1 paths
```

#### Note

The **is-local** matching parameter and regex query **^\$** is used by organizations that use multiple service providers and want to ensure that only their prefixes are advertised and not others.



## length

The **length** option allows for the selection of routes based on the length (count) of the AS\_Path entries. Selection can be based on a specific value, greater than or equal to, or less than or equal to a specific length. The syntax is **length {eq | ge | le | is} {count | parameter}**. IOS XR RPLs supports parameterization, which is explained later in this chapter in the section “[Parameterization](#).”

**Example 11-24** shows the reference BGP table when using the IOS XR **length** query for prefixes with a length greater than or equal to 3.

### Example 11-24 BGP Prefix Selection with length ge 3

[Click here to view code image](#)

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.17.100.0/24	172.17.0.2	0		0	200 300 400 i
*> 172.17.200.0/24	172.17.0.2	0		0	200 300 300 300 300 i
*> 172.18.100.0/24	172.18.0.2	0		0	500 550 200 i
*> 172.19.100.0/24	172.19.0.2	0		0	600 200 600 i

Processed 4 prefixes, 4 paths

## unique-length

A common BGP technique involves prepending (adding) your ASN multiple times to a prefix to influence the BGP best path selection algorithm. In the 172.17.200.0/24 prefix, AS300 was prepended three times. The **unique-length** option selects routes based on the length of unique ASNs. Prepended ASNs are not included. The syntax is **unique-length {eq | ge | le | is} {count | parameter}**.

**Example 11-25** shows the reference BGP table when using the IOS XR **unique-length** query selecting prefixes with a unique length greater than or equal to 3. Notice that the prefix 172.17.200.0/24 is not included, because it only has two unique ASNs, and that the other prefixes have three unique ASNs.

### Example 11-25 BGP Prefix Selection with unique-length ge 3

[Click here to view code image](#)

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.17.100.0/24	172.17.0.2	0		0	200 300 400 i
*> 172.18.100.0/24	172.18.0.2	0		0	500 550 200 i
*> 172.19.100.0/24	172.19.0.2	0		0	600 200 600 i

Processed 3 prefixes, 3 paths

## passes-through

The **passes-through** option identifies a prefix that passes through the ASNs listed in the provided ASN pattern. The syntax is **passes-through {as-number-pattern | parameter} [exact]**. The *as-number-pattern* can include multiple entries like 10 20. Without the **exact** keyword multiple instances of the autonomous system like 10 10 10 20 would still be a valid match for the pattern 10 20.

Note

In lieu of using an *as-number-pattern* or *parameter*, a range can be specified by placing two numbers in brackets between two periods. The range 1 to 500 could be represented as [1..500].

Example 11-26 shows the reference BGP table when using the **passes-through** query for prefixes that use an *as-number-pattern* of 200. Notice that the ASN could be from the originating autonomous system, neighboring autonomous system, or transit autonomous system.

**Example 11-26 BGP Prefix Selection with passes-Through '200'**

[Click here to view code image](#)

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.17.100.0/24	172.17.0.2	0		0	200 300 400 i
*> 172.17.200.0/24	172.17.0.2	0		0	200 300 300 300 300 i
*> 172.18.100.0/24	172.18.0.2	0		0	500 550 200 i
*> 172.19.100.0/24	172.19.0.2	0		0	600 200 600 i

Processed 4 prefixes, 4 paths

**neighbor-is**

The **neighbor-is** option identifies a prefix that passes is received from a neighbor in the specified ASN. The syntax is **neighbor-is** '{*as-number-pattern* | *parameter*}' [**exact**].

Example 11-27 shows the reference BGP table when using the **neighbor-is** query for prefixes that use an *as-number-pattern* of '200'. Notice that the only prefixes that come directly from the neighbor AS200 are in the BGP table.

**Example 11-27 BGP Prefix Selection with neighbor-is '200'**

[Click here to view code image](#)

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.17.100.0/24	172.17.0.2	0		0	200 300 400 i
*> 172.17.200.0/24	172.17.0.2	0		0	200 300 300 300 300 i

**originates-from**

The **originates-from** option identifies a prefix that passes is received from a neighbor in the specified ASN. The syntax is **originates-from** '{*as-number-pattern* | *parameter*}' [**exact**].

Example 11-28 shows the reference BGP table when using the **originates-from** query for prefixes that use an *as-number-pattern* of '200'. Notice that the only prefix in the BGP table originates from AS200.

**Example 11-28 BGP Prefix Selection with originates-from '200'**

[Click here to view code image](#)

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.18.100.0/24	172.18.0.2	0	0	500	550 200 i

Processed 1 prefixes, 1 paths

### AS Path Set

IOS XR AS path sets are a comma-delimited set of ASN matching options. AS path sets are a component of IOS XR's RPL and do not allow the usage of blocking a matching pattern. Once an ASN is matched, searching in the AS path set stops. AS path set configuration uses the following steps:

#### Step 1. Define the AS Path set.

Define the AS path set with the command **as-path-set** *as-path-set-name*.

#### Step 2. Enter the AS Path set entries.

Configure each specific prefix entry with one of the following commands:

- **ios-regex** '*regex-query*'
- **length** {*eq* | *ge* | *le* | *is*} {*count*|*parameter*}
- **unique-length** {*eq* | *ge* | *le* | *is*} {*count*|*parameter*}
- **passes-through** '{*as-number-pattern* | *parameter*}' [**exact**]
- **neighbor-is** '{*as-number-pattern* | *parameter*}' [**exact**]
- **originates-from** '{*as-number-pattern* | *parameter*}' [**exact**]

Step 2 is repeated for multiple entries with every entry except the last one delimited with a comma.

#### Step 3. Exit the AS Path set.

Exit the AS Path set with the command **end-set**.

Example 11-29 provides an AS path set configuration based on previous queries.

#### **Example 11-29** *AS\_Path Set Configuration*

[Click here to view code image](#)

```
as-path-set SAMPLE
  ios-regex '*',
  length ge 3,
  unique-length ge 3,
  passes-through '200',
  neighbor-is '200',
  originates-from '200'
end-set
```

## ROUTE MAPS

Route maps provide many different features to a variety of routing protocols. At the simplest level, route maps can filter networks similar to an ACL, but also provide additional capability to IOS nodes by adding or modifying a network attribute. Route maps must be referenced within a routing protocol to influence it. Route maps are a critical to BGP because it is the main component of modifying a unique routing policy on a neighbor-by-neighbor basis.

Route maps consist of four components:

- **Sequence Number:** Dictates the processing order of the route map.
- **Conditional Matching Criteria:** Identifies prefix characteristics (network, BGP path attribute, next hop, and so on) for a specific sequence.
- **Processing Action:** Permits or denies the prefix.
- **Optional Action:** Allows for manipulations depending on how the route map is referenced on the router. Actions can include modification, addition, or removal of route characteristics.

Route maps use the following command syntax: **route-map** *route-map-name* [**permit** | **deny**] [*sequence-number*]. The following rules apply to **route-map** statements:

- If a processing action is not provided, the default value of **permit** is used.
- If a sequence number is not provided, IOS will use the sequence number 10 automatically. It will not auto-matically increment on repeat command entries.
- If a matching statement is not included, an implied '*all prefixes*' is associated to the statement.
- Processing within a route map stops after all optional actions have processed (if configured) after matching a matching criteria.

**Example 11-30** provides a sample route map to demonstrate the four components of a route map shown earlier. The conditional matching criteria is based on network ranges specified in an ACL. Comments have been added to explain the behavior of the route map in each sequence.

### **Example 11-30** *Sample Route Map*

[Click here to view code image](#)

```
route-map EXAMPLE permit 10
  match ip address ACL-ONE
! Prefixes that match ACL-ONE are permitted. Route map completes processing upon
! a match

route-map EXAMPLE deny 20
  match ip address ACL-TWO
! Prefixes that match ACL-TWO are denied. Route map completes processing upon a
! match

route-map EXAMPLE permit 30
  match ip address ACL-THREE
  set metric 20
! Prefixes that match ACL-THREE are permitted and modify the metric. Route map
! completes processing upon a match

route-map EXAMPLE permit 40
! Because a matching criteria was not specified, all other prefixes are permitted
! If this sequence was not configured, all other prefixes would drop because of
! the implicit deny for all route maps
```

#### Note

When deleting a specific route map statement, include the sequence number to prevent deleting the entire route map.

### Conditional Matching

Now that the components and processing order of a route map have been explained, this section will expand on the aspect of how a route can be matched. [Example 11-28](#) shows the various options available within IOS.

As you can see, a number of conditional matching options are available. [Table 11-5](#) describes the command syntax for the most common methods for matching prefixes.

Match Command	Description
<code>match as-path <i>acl-number</i></code>	Selects prefixes based on regex query to isolate the ASN in the BGP path attribute (PA) AS_Path * Allows for multiple match variables
<code>match ip address {<i>acl-number</i>   <i>acl-name</i>}</code>	Selects prefixes based on network selection criteria defined in the ACL * Allows for multiple match variables
<code>match ip address prefix-list <i>prefix-list-name</i></code>	Selects prefixes based on prefix selection criteria * Allows for multiple match variables
<code>match local-preference</code>	Selects prefixes based on the BGP attribute local preference * Allows for multiple match variables
<code>match metric {1-4294967295   external 1-4294967295}[+ <i>deviation</i>]</code>	Selects prefixes based on the metric that can be exact, a range, or within acceptable deviation
<code>match tag <i>tag-value</i></code>	Selects prefixes based on a numeric tag (0–4294967295) that was set by another router * Allows for multiple match variables

Table 11-5 Conditional Match Options

### Multiple Conditional Match Conditions

If there are multiple variables of the same type (ACLs, prefix lists, tags, and so on) configured for a specific route map sequence, only one variable must match for the prefix to qualify. The Boolean logic uses an “**or**” operator for this configuration.

In [Example 11-31](#), sequence 10 requires that a prefix pass ACL-ONE or ACL-TWO. Notice that sequence 20 does not have a **match** statement, so all prefixes that are not passed in sequence 10 will qualify and are denied.

### Example 11-31 Multiple Match Variables Route Map

[Click here to view code image](#)

```
route-map EXAMPLE permit 10
  match ip address ACL-ONE ACL-TWO
!
route-map EXAMPLE deny 20
```

#### Note

Sequence 20 is redundant because of the implicit deny for any prefixes that are not matched in sequence 10.

If multiple match options are configured for a specific route map sequence, both match options must be met for the prefix to qualify for that sequence. The Boolean logic uses an “**and**” operator for this configuration.

In [Example 11-32](#), sequence 10 requires that the prefix match ACL ACL-ONE and that the metric be a value between 500 and 600. If the prefix does not qualify for both match options, the prefix will not qualify for sequence 10 and is denied because another sequence does not exist with a permit action.

### Example 11-32 Multiple Match Options Route Map

[Click here to view code image](#)

```
route-map EXAMPLE permit 10
  match ip address ACL-ONE
  match metric 550 +- 50
```

#### Complex Matching

Some network engineers find route maps too complex if the conditional matching criteria use an ACL, AS\_Path ACL, or prefix list that contains a **deny** statement in it. [Example 11-33](#) demonstrates a configuration where the ACL uses a **deny** statement for the 172.16.1.0/24 network range.

Reading configurations like this should follow the sequence order first, conditional matching criteria second, and only after a match occurs should the processing action and optional action be used. Matching a **deny** statement in the conditional match criteria excludes the route from that sequence in the route map.

The prefix 172.16.1.0/24 is denied by ACL-ONE, so that infers that there is not a match in sequence 10 and 20; therefore, the processing action (permit or deny) is not needed. Sequence 30 does not contain a match clause, so any remaining routes are permitted. The prefix 172.16.1.0/24 would pass on sequence 30 with the metric set to 20. The prefix 172.16.2.0/24 would match ACL-ONE and would pass in sequence 10.

### Example 11-33 Complex Matching Route Maps

[Click here to view code image](#)

```
ip access-list standard ACL-ONE
  deny 172.16.1.0 0.0.0.255
  permit 172.16.0.0 0.0.255.255

route-map EXAMPLE permit 10
  match ip address ACL-ONE
  !
route-map EXAMPLE deny 20
  match ip address ACL-ONE
  !
route-map EXAMPLE permit 30
  set metric 20
```

#### Note

Route maps process in the order of evaluation of the sequence, conditional match criteria, processing action, and optional action in that order. Any **deny** statements in the match component are isolated from the route map sequence action.

## Optional Actions

In addition to permitting the prefix to pass, route maps can modify route attributes. Table 11-6 briefly describes the most popular attribute modifications.

Set Action	Description
set as-path prepend {as-number-pattern   last-as 1-10}	Prepends the AS_Path for the network prefix with the pattern specified, or from multiple iterations from neighboring autonomous system.
set ip next-hop {ip-address   peer-address   self }	Sets the next-hop IP address for any matching prefix. BGP dynamic manipulation uses the peer-address or self keywords.
set local-preference 0-4294967295	Sets the BGP PA local preference.
set metric {+value   -value   value} *value parameters Are 0-4294967295	Modifies the existing metric or sets the metric for a route.
set origin {igp   incomplete}	Sets the BGP PA origin.
set tag tag-value	Sets a numeric tag (0–4294967295) for identification of networks by other routers.
set weight 0-65535	Sets the BGP PA weight.

Table 11-6 Route Map Set Actions

## Continue

Default route map behavior processes the route map sequences in order, and upon the first match, executes the processing action, performs any optional action (if feasible), and stops processing. This prevents multiple route map sequences from processing.

Adding the keyword **continue** to a route map allows the route map to continue processing other route map sequences. Example 11-34 provides a basic configuration. The network prefix 192.168.1.1 matches in sequence 10, 20, and 30. Because the keyword **continue** was added to sequence 10, sequence 20 will process, but sequence 30 will not because a **continue** command was not present in sequence 20. The 192.168.1.1 prefix will be permitted and modified so that the metric is 20 with a next-hop address of 10.12.1.1.

## Example 11-34 Route-Map with continue Keyword

[Click here to view code image](#)



```

ip access-list standard ACL-ONE
 permit 192.168.1.1 0.0.0.0
 permit 172.16.0.0 0.0.255.255
 !
ip access-list standard ACL-TWO
 permit 192.168.1.1 0.0.0.0
 permit 172.31.0.0 0.0.255.255
 !
route-map EXAMPLE permit 10
 match ip address ACL-ONE
 set metric 20
continue
!
route-map EXAMPLE permit 20
 match ip address ACL-TWO
 set ip next-hop 10.12.1.1
!
route-map EXAMPLE permit 30
 set ip next-hop 10.13.1.3

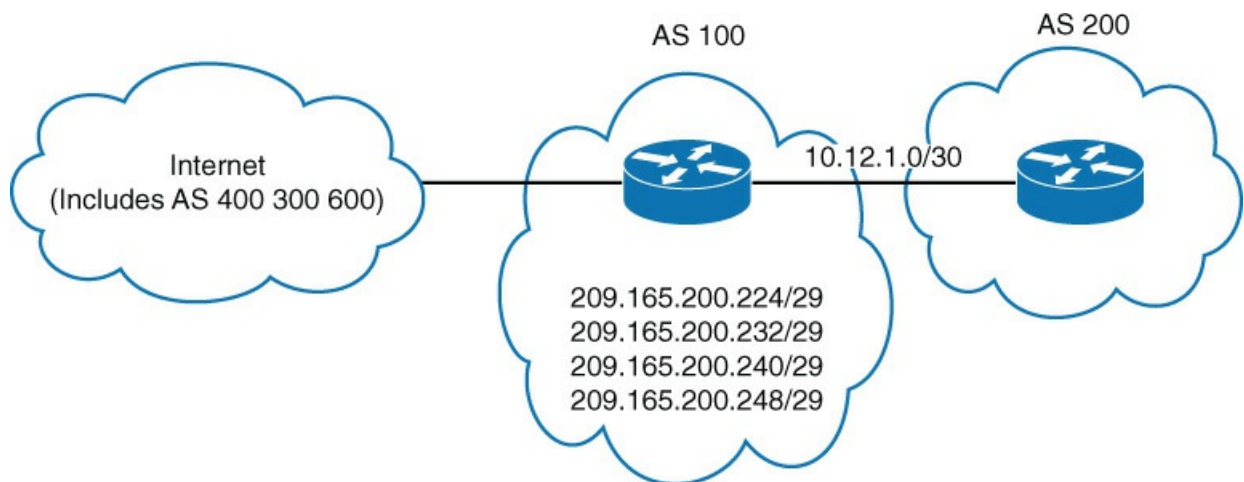
```

#### Note

The usage of the **continue** command is not common and adds complexity when troubleshooting route maps.

### Route Map Examples

To ensure that prefix lists, regex queries, and route maps are fully understood, this section provides a brief scenario to demonstrate the components of a route map when used with BGP as an attach point. In [Figure 11-8](#), AS200 connects to AS100, which then connects to the Internet that contains AS300, AS400, and AS600.



**Figure 11-8** *Topology Example*

[Example 11-35](#) shows AS200's BGP table with a standard eBGP peering to AS200 without a route map.

### **Example 11-35** *AS200 BGP Table*

[Click here to view code image](#)

```
AS200#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 209.165.200.224/29	10.12.1.1	100		0 100	400 300 600 i
*> 209.165.200.232/29	10.12.1.1	200		0 100	400 300 i
*> 209.165.200.240/29	10.12.1.1	300		0 100	400 i
*> 209.165.200.248/29	10.12.1.1	400		0 100	i

**Example 11-36** provides the configuration for the AS200 router to match the routing policy. AS200's routing policy states that the route 209.165.200.248/29 should be blocked, and any routes that contain AS300 in the AS\_Path should have the local preference set to 500.

### Example 11-36 AS200's Router Configuration

[Click here to view code image](#)

```
! The route-map is attached to the BGP neighbor in the appropriate address-family
```

```
router bgp 200
  bgp log-neighbor-changes
  no bgp default ipv4-unicast
  neighbor 10.12.1.1 remote-as 100
!
  address-family ipv4
    neighbor 10.12.1.1 activate
    neighbor 10.12.1.1 route-map SAMPLE in
  exit-address-family
```

```
! The prefix-list is created to match the prefix that is to be blocked
```

```
ip prefix-list PREFIX-BLOCK seq 10 permit 209.165.200.248/29
```

```
! The AS-Path ACL is created to match any prefixes that pass through AS300
```

```
ip as-path access-list 1 permit _300_
```

```
! The route-map provides the logic for AS200's route-policy. Notice that a
! variety of conditional match and action items are shown
```

```
route-map SAMPLE deny 5
  match ip address prefix-list PREFIX-BLOCK
!
route-map SAMPLE permit 10
  match as-path 1
  set local-preference 500
!
route-map SAMPLE permit 20
```

**Example 11-37** provides the BGP table after the SAMPLE route map was applied to the BGP neighbor 10.12.1.1 for inbound route policy processing. The 209.165.200.248/29 prefix was filtered out, and because 209.165.200.224/29 and 209.165.200.232/29 transit AS300, the local preference was set to 500.

### Example 11-37 AS200 BGP Table After Inbound SAMPLE Route Map

[Click here to view code image](#)

```
AS200#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 209.165.200.224/29	10.12.1.1	100	500	0 100 400	300 600 i
*> 209.165.200.232/29	10.12.1.1	200	500	0 100 400	300 i
*> 209.165.200.240/29	10.12.1.1		300		0 100 400 i

## ROUTING POLICY LANGUAGE

Routing policy language (RPL) expands on the fundamental concepts of route maps and provides a more powerful method of filtering and manipulating routes. RPL provides a common tool for IOS XR applications that commonly interact with BGP, IGP, and other component of the operating system. In addition to route policies, RPL includes *policy sets*. Policy sets consist of prefix sets, AS path sets, and community sets.

The RPL architecture addresses the following concerns:

- **Clarity:** Route policies use common conditional *if-then-else* programming structures, removing sequence numbers and processing in a top-down methodology similar to a computer program.

Route policies do not use ACLs, prefix lists, AS\_Path ACLs, or other components that allow the usage of **deny** statements. Deny statements in policy sets can add confusion when combined with the action to perform. RPL moves all permit or deny actions to the route policy itself, and not the matching criteria.

- **Scalability:** RPL allows for nesting of route policies. A route policy can reuse the logic defined in another route policy, thereby creating a hierarchical route policy and simplifying the parent route policy.

*Parameterization* provides a method of passing a value at an attachment point when the RPL is invoked, and allowing the reuse of RPL logic, but only changing the values modified.

Every route sent to a route policy for processing passes through the route policy structure. To ensure that every route has been examined, a *'ticket'* is required for the route to pass the default drop at the end of the route policy.

Route policies consist of four primary actions:

- **Pass:** The route is assigned a ticket to pass the default drop. Processing continues through additional RPL statements.

- **Done:** The route is assigned a ticket to pass the default drop. Processing stops for the specific route.

- **Set:** The route is assigned a ticket to pass the default drop. The specific route attribute is modified to the value set, and processing continues.

- **Drop:** The route is discarded and processing stops.

## Route Policy Structure

Route policy configuration uses the command **route-policy** *route-policy-name*. Inside the route policy, any actions, conditional **match** statements, and attribute modifications occur. Once all the route policy components are finalized, the route policy is defined with the command **end-policy**. Conditional **match** statements are not required, as shown in [Example 11-38](#), where an RPL can pass or drop all routes.

### Example 11-38 Simple Route Policy

[Click here to view code image](#)

```
route-policy PASS-ALL
  pass
end-policy
```

```
route-policy DROP-ALL
  drop
end-policy
```

#### Note

Route policy syntax verification occurs when the route policy is attached to the routing component to ensure that all statements are relevant to the protocol.

## Match Statements

Most route policies require manipulation based on specific route attributes. Match statements occur after **if** or **elseif** statements. [Table 11-7](#) briefly describes the most popular attributes that can be queried during route selection.

Match Command	Description
if as-path in {parameter   as-path-set-name   (ios-regex 'regex-pattern')}	Selects BGP prefixes based on AS Path sets or usage of an IOS regex query.
if as-path is-local	Selects BGP prefixes based on the prefix originating from an iBGP peer.
if as-path {neighbor-is   originates-from   passes-through } '{as-number-pattern   parameter}' [exact]	Selects BGP prefixes based on the prefix originating, received from, or transiting a specific ASN.
if as-path {length   unique-length} {eq   ge   is   le} {count   parameter}	Selects BGP prefixes based on the count of ASNs in the BGP prefix AS_Path path attribute.
if destination in {parameter   prefix-set-name   (high-order-bit-pattern [/high-order-bit-count] [ge ge-value] [le le-value])}	Selects prefixes based on the prefix set, or a prefix specification match parameters. The eq <i>eq-value</i> can be used in lieu of setting the <i>ge-value</i> and <i>le-value</i> to the same number.
if local-preference {eq   ge   is   le} {local-preference   parameter}	Selects prefixes based on the BGP PA local preference.
if med {eq   ge   is   le} {local-preference   parameter}	Selects prefixes based on the BGP PA MED (Multi-Exit Discriminator).
if next-hop in {parameter   prefix-set-name   (high-order-bit-pattern [/high-order-bit-count] [ge ge-value] [le le-value])}	Selects prefixes based on the next-hop attribute for the route. The eq <i>eq-value</i> can be used in lieu of setting the <i>ge-value</i> and <i>le-value</i> to the same number.
if origin is {egp   igp   incomplete   parameter}	Selects prefixes based on the BGP PA origin.
if tag {eq   ge   is   le} {tag-value   parameter}	Selects prefixes based a numeric tag (0–4294967295) that was set by another router

Table 11-7 Route Policy Match Options

Route maps required the use of establishing ACLs, prefix lists, AS\_Path ACLS, and so on for route selection to occur. Troubleshooting route maps may require looking at various components of the configuration to fully understand the logic. RPL provides improved troubleshooting capability by allowing the values to be set in the RPL directly (known as *inline*). Inline statements are normally contained within parentheses, and can contain multiple entries that are delimited by a comma.

#### Attribute Modification

In addition to the commands **done**, **drop**, and **pass**, route policies can modify route attributes. Table 11-8 briefly describes the most popular attribute modifications.

Set Action	Description
<code>prepend as-path { most-recent count   as-number-pattern count   parameter }</code>	Prepends the AS_Path for the network prefix with the pattern specified, or from multiple iterations from the neighboring autonomous system.
<code>replace as-path '{parameter   as-number-pattern}'</code>	Replaces the AS_Path pattern with the local autonomous system.
<code>set local-preference {local-preference   * value   + value   - value   parameter }</code>	Sets the BGP PA local preference. Using the *, +, - functions requires a value for modification which can be parameterized as well.
<code>set med {med   + value   - value   igp-cost   max-reachable   parameter }</code>	Sets the BGP PA MED. Using the + and – functions requires a value for modification, which can be parameterized as well.
<code>set next-hop {ip-address   discard   peer-address   self   parameter }</code>	Sets the next-hop IP address for matching prefix. BGP dynamic manipulation uses the <code>peer-address</code> or <code>self</code> keywords. The <code>discard</code> keyword automatically discards the route.
<code>set origin {egp   igp   incomplete   parameter }</code>	Sets the BGP PA origin.
<code>if tag {tag-value   parameter }</code>	Set a numeric tag (0–4294967295) for identification by another router.
<code>set weight {weight   parameter }</code>	Sets the BGP PA weight.

Table 11-8 Route Policy Attribute Modification Actions

### Common Route Policy Structure

Example 11-39 demonstrates a conditional match and action statement. Match-Condition-One must be true in order for Action-One and Action-Two to execute.

#### Example 11-39 Conditional Match and Action Statement

[Click here to view code image](#)

```
if Match-Condition-One then
    Action-One
    Action-Two
end-if
```

Example 11-40 provides a simple conditional RPL that passes all network prefixes within RFC 1918 address space. All three of the network ranges are inline between the parentheses with commas as delimiters. Notice that the drop action is not specified or required because only the matched prefixes are granted tickets to bypass the default drop.

#### Example 11-40 Inline Prefix Filtering Route Policy

[Click here to view code image](#)

```
route-policy RFC1918-INLINE
! The following two lines are entered into the CLI as one entry
  if destination in (10.0.0.0/8 ge 8, 172.16.0.0/12 ge 12, 192.168.0.0/16 ge 16)
then
  pass
endif
end-policy
```

**Example 11-41** provides the same logic, but the prefixes are in a prefix set versus inline configuration. Prefix sets are commonly used when multiple entries are needed, and inline configuration is used when the number of matching prefixes is smaller.

### **Example 11-41** *Prefix Set Filtering Route Policy*

[Click here to view code image](#)

```
prefix-set PREFIX-SET-RFC1918
  10.0.0.0/8 ge 8,
  172.16.0.0/12 ge 12,
  192.168.0.0/16 ge 16
end-set
!
route-policy RFC1918-PREFIX-SET
  if destination in PREFIX-SET-RFC1918 then
    pass
  endif
end-policy
```

When troubleshooting route policy configuration issues, policy sets can be expanded to show the entries in inline methodology with the command **show rpl route-policy route-policy-name inline**. **Example 11-42** demonstrates this feature.

### **Example 11-42** *Inline Policy Set Expansion*

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show rpl route-policy RFC1918-PREFIX-SET inline

route-policy RFC1918
  if destination in (10.0.0.0/8 ge 8, 172.16.0.0/12 ge 12, 192.168.0.0/16 ge 16)
then
  pass
endif
end-policy
```

**Example 11-43** illustrates a conditional match using ‘if-else’ logic. Objects that Match-Condition-One will execute Action-One, otherwise Action-Two will execute.

### **Example 11-43** *Conditional Match Using If-Else Logic*

[Click here to view code image](#)

```
if Match-Condition-One then
    Action-One
else
    Action-Two
end-if
```

**Example 11-44** expands on the previous RFC 1918 route policy logic using *'if-else'* conditional programming structure that passes all network prefixes within RFC 1918 address space. Even though the **drop** action is not required, it removes some of the vagueness.

### **Example 11-44** RFC 1918 Route Map with If-Else Logic

[Click here to view code image](#)

```
route-policy IF-ELSE-RFC1918
! The following two lines are one entry in the CLI
  if destination in (10.0.0.0/8 ge 8, 172.16.0.0/12 ge 12, 192.168.0.0/16 ge 16)
  then
    pass
  else
    drop
  endif
end-policy
```

**Example 11-45** illustrates the concept of the *elseif* logic that allows a second conditional match to execute in the event that the first conditional match failed.

### **Example 11-45** Conditional Match Using If-ElseIf Logic

[Click here to view code image](#)

```
if Match-Condition-One then
    Action-One
elseif Matching-Condition-Two then
    Action-Two
end-if
```

**Example 11-46** illustrates the concept of *if-elseif-else*, where there can be multiple (**elseif**) statements, but only one (**else**) statement.

### **Example 11-46** Conditional Match Using If-ElseIf-Else Logic

[Click here to view code image](#)



```
if Match-Condition-One then
    Action-One
elseif Match-Condition-Two then
    Action-Two
elseif Matching-Condition-Three then
    Action-Three
else
    Action-Four
end-if
```

**Example 11-47** demonstrates the RFC 1918 route policy using *if-elseif-else* logic. If the route is in the 10.0.0.0/8 network range, the prefix is granted a ticket. Routes that do not match the first criteria then identify whether the route is in the 172.16.0.0/12 network range, before granting a ticket. The third conditional match verifies that the route is in the 192.168.0.0/16 network range before granting a ticket. Any routes that have not passed the first three conditional match statements perform the drop action.

### **Example 11-47** *If-ElseIf-Else Route-Policy*

[Click here to view code image](#)

```
route-policy IF-ELSEIF-ELSE-RFC1918
  if destination in (10.0.0.0/8 ge 8) then
    pass
  elseif destination in (172.16.0.0/12 ge 12) then
    pass
  elseif destination in (192.168.0.0/16 ge 16) then
    pass
  else
    drop
  endif
end-policy
```

**Example 11-48** illustrates the concept of nesting conditional matches before taking action. The logic shows a simple two-condition match.

### **Example 11-48** *Conditional Statement Nesting*

[Click here to view code image](#)

```
if MATCHING-CONDITION-ONE then
    if AMATCHING-CONDITION-TWO then
        ACTION-ONE
    end-if
end-if
```

**Example 11-49** illustrates the concept of nesting conditional matches before taking action. The logic shows that Action-One or Action-Two depend on Match-Condition-One and Match-Condition-Two. Action-Three is depends solely on Match-Condition-One.

## Example 11-49 *Advanced Conditional Statement Nesting*

[Click here to view code image](#)

```
if Match-Condition-One then
  if Match-Condition-Two then
    Action-One
  else
    Action-Two
  end-if
else
  Action-Three
end-if
```

### Note

Conditional statement nesting provides a method of reducing processing time, and the first match is normally the most restrictive to minimize unnecessary processing on the secondary conditional statement.

[Example 11-50](#) demonstrates a route policy with nested conditional statements that pass only routes in the RFC 1918 space that pass through AS100. The AS requirement is more restrictive than the RFC1918 space, and is placed first.

## Example 11-50 *Nested Conditional Match*

[Click here to view code image](#)

```
route-policy MULTIPLE-CHECKS
  if as-path passes-through '100' then
    if destination in PREFIX-SET-RFC1918 then
      pass
    endif
  endif
end-policy
```

Unlike route maps, route policies continue to process unless they receive the **drop** or **done** keyword. [Example 11-51](#) illustrates a badly designed RPL that was intended to set the MED to 100 for prefixes in the 192.168.0.0/16 network range while setting the MED to 200 for all other routes.

## Example 11-51 *Bad RPL Design*

[Click here to view code image](#)

```
route-policy METRIC-MODIFICATION
  if destination in (192.168.0.0/16 ge 16) then
    set med 100
  endif
  set med 200
end-policy
```

Upon testing, all routes receive the MED of 200, including those within the 192.168.0.0/16 network range, because the **set med 200** statement is not conditional and executes after the MED was modified on the 192.168.0.0/16 prefixes.

The use of the **done** command, which stops processing of the route policy, or usage of *if-else* logic remediates the design flaw. [Example 11-52](#) shows both solutions.

### Example 11-52 Good RPL Design

[Click here to view code image](#)

```
route-policy METRIC-MODIFICATION
  if destination in (10.0.0.0/8 ge 8) then
    set med 100
    done
  endif
  set med 200
end-policy
```

```
route-policy METRIC-MODIFICATION
  if destination in (10.0.0.0/8 ge 8) then
    set med 100
  else
    set med 200
  endif
end-policy
```

### Boolean Operators

Boolean operators provide a method to create compound conditions without the usage of nested conditional statements.

#### Negation

The **not** keyword is negation and essentially inverts the outcome. If a conditional match returns a value of true, a negated version of the conditional match returns a value of false. If a conditional match returns a value of false, a negated version of the conditional match returns a value of true. Negation has the highest precedence of the Boolean operators.

[Example 11-53](#) demonstrates a route policy that passes *only* those routes that are *not* in the RFC 1918 address range.

### Example 11-53 Boolean Negation Example

[Click here to view code image](#)

```
route-policy BOOLEAN-NOT
  if not destination in PREFIX-SET-RFC1918 then
    pass
  endif
end-policy
```

#### Conjunction

The **and** keyword is conjunctive and requires that both conditional matches are true for the conditional action to execute. Conjunction has the second highest precedence of the Boolean operators.

[Example 11-54](#) demonstrates a route policy that requires a route to match the PREFIX-SET-RFC1918 and pass through AS100 to pass. If both conditions are not met, the prefix does not pass the default drop at the end of the route policy.

### **Example 11-54** Boolean Conjunctive Example

[Click here to view code image](#)

```
route-policy BOOLEAN-AND
  if destination in PREFIX-SET-RFC1918 and as-path passes-through '100' then
    pass
  endif
end-policy
```

### **Disjunction**

The **or** keyword is disjunctive and allows one or both conditional matches to be true for the conditional action to execute. Disjunction has the lowest precedence of the Boolean operators.

[Example 11-55](#) demonstrates a route policy that requires a route to match the PREFIX-SET-RFC1918 or pass through AS100 to pass. If neither condition is met, the prefix does not pass the default drop at the end of the route policy.

### **Example 11-55** Boolean Disjunctive Example

[Click here to view code image](#)

```
route-policy BOOLEAN-OR
  if destination in PREFIX-SET-RFC1918 or as-path passes-through '100' then
    pass
  endif
end-policy
```

### **Order of Processing**

Boolean operators process in the following order:

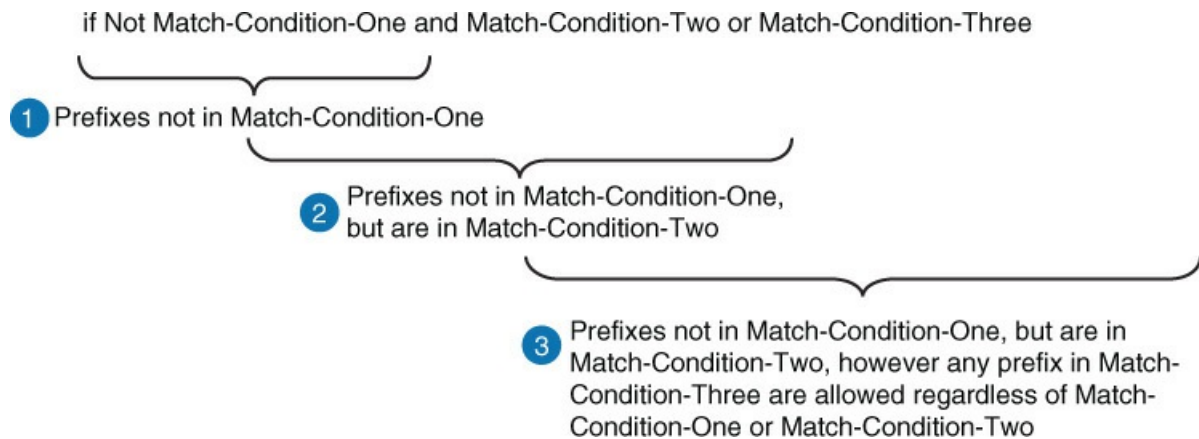
1. Parentheses
2. Not
3. And
4. Or

[Figure 11-9](#) demonstrates the processing order of conditional match statement with multiple Boolean operators:

**Step 1.** Combines all the negation operators

**Step 2.** Combines the negated results with the conjunction **match** statements

**Step 3.** Selects either the results from Step 2 or the newly added match condition



**Figure 11-9** Boolean Operator Order of Processing

The order of processing is not intuitive to some engineers, and so it is beneficial to use parentheses to clarify the logic. The following shows the same Boolean operator order of processing logic from [Figure 11-9](#) but with parentheses:

if (((Not **Match-Condition-One**) and **Match-Condition-Two**) or

**Match-Condition-Three**)

Parentheses preempt all other Boolean operators and can change the output of a route policy. [Example 11-56](#) demonstrates a route policy that requires a route to not pass through AS100 or AS200 and must be within the 192.168.0.0/16 network range.

### **Example 11-56** Route Policy Parentheses

[Click here to view code image](#)

```
! The following two lines are one entry in the CLI
if not (as-path passes-through '100' or as-path passes-through '200') and
destination in (192.168.0.0/16 ge 16)
```

If parentheses were not used, the route policy would state that the route does not pass through AS100, but does pass through AS200, and is within the 192.168.0.0/16 network range.

### **Comparing Prefix Sets to Prefix Lists**

IOS XR route policies do not use ACLs, prefix lists, AS\_Path ACLs, or other components that allow the usage of **deny** statements. Policy sets do not include the capability to deny entries, but equivalent functionality comes through the use of Boolean negation and conjunction.

[Example 11-57](#) represents a prefix list that blocks the 192.168.1.1/32 prefix, and permits all other prefixes within the 192.168.0.0/16 network range.

### **Example 11-57** Prefix List

[Click here to view code image](#)

```
ipv4 prefix-list PREFIX-LIST deny 192.168.1.1/32
ipv4 prefix-list PREFIX-LIST permit 192.168.0.0/16 ge 16
```

**Example 11-58** demonstrates a route policy that blocks the 192.168.1.1/32 prefix while matching all other prefixes within the 192.168.0.0/16 network range. This example uses inline prefixes, but could also use prefix sets or any other matching capability. Notice the use of parentheses to clarify the order of processing.

### **Example 11-58** *Route-Policy with Prefix List Matching Capability*

[Click here to view code image](#)

```
route-policy PREFIX-LIST
! The following two lines are one entry in the CLI
  if (destination in (192.168.0.0/16 ge 16) and (not destination in (192.168.1.1)))
  then
    pass
  endif
end-policy
```

### **Parameterization**

Parameterization allows for the reuse of route policies that are identical except for the values that are used for a matching condition or for route manipulation. **Example 11-59** demonstrates two simple route policies that verify that routes only originate from the appropriate number.

### **Example 11-59** *Two Route Policies with Identical Logic*

[Click here to view code image](#)

```
route-policy NEIGHBOR-ONE
  if as-path neighbor-is '1' then
    pass
  endif
end-policy

route-policy NEIGHBOR-TWO
  if as-path neighbor-is '2' then
    pass
  endif
end-policy
```

Parameterization allows one route policy to be created that accepts variables for processing in the logic. The number of parameters are defined when the route policy is created, and are contained in parentheses with comma delimitation. The parameters are user-friendly words that must start with a \$ (dollar sign).

**Example 11-60** shows the previous RPL with parameterization. Notice where the parameter is defined in the RPL.

### **Example 11-60** *Route Policy with Parameterization*

[Click here to view code image](#)

```
route-policy PARAMETERIZATION($PARAM1)
  if as-path neighbor-is '$PARAM1' then
    pass
  endif
end-policy
```

The actual value passed to the route policy is configured at the attachment point of the RPL. This RPL is based on BGP path attributes, and the parameter is set within the BGP configuration, as shown in [Example 11-61](#).

### Example 11-61 BGP Route Policy with Parameterization

[Click here to view code image](#)

```
! Some configuration omitted for brevity
router bgp 200
  neighbor 10.1.1.1
    address-family ipv4 unicast
      route-policy PARAMETERIZATION(1) in
    !
  neighbor 10.2.2.2
    address-family ipv4 unicast
      route-policy PARAMETERIZATION(2) in
```

The passing of parameters allows for prefix set names, AS Path set names, or actual values. It does not allow for spaces or special characters and some matching criteria works better when entered as a set, and only the set name is passed as a parameter. [Example 11-62](#) demonstrates a valid route policy attachment command and an invalid route policy attachment command caused due to special characters in the parameter passing.

### Example 11-62 Route Policy with Multiple Parameterization

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1 (config-bgp-nbr-af) #route-policy PARM-PREFIX-LEN(192.168.0.0) in
RP/0/0/CPU0:XR1 (config-bgp-nbr-af) #route-policy PARM-PREFIX-LEN(192.168.0.0/24) in
% Invalid input detected at '^' marker.
```

Parameterization can use multiple values for matching or setting values. [Example 11-63](#) demonstrates a route policy that accepts multiple parameters and sets the MED if the prefix comes from the neighbor autonomous system specified.

### Example 11-63 Route Policy with Multiple Parameterization

[Click here to view code image](#)

```
route-policy PARAMETERIZATION($ASPATH-PARAM, $MED-PARAM)
  if as-path neighbor-is '$ASPATH-PARAM' then
    set med $MED-PARAM
  else
    pass
  endif
end-policy
```

**Example 11-64** demonstrates multiple parameters passed to the route policy at the point of attachment.

### **Example 11-64** BGP Route Policy with Parameterization

[Click here to view code image](#)

```
! Some configuration omitted for brevity
router bgp 200
  neighbor 10.1.1.1
  address-family ipv4 unicast
    route-policy PARAMETERIZATION(1, 100) in
```

Imagine a router that has 100 neighbors that use the same logic, with minor changes to the variables. Finding the exact route policy can be a tedious and time-consuming process when troubleshooting an issue. Using parameterization in a route policy reduces the size of the configuration, simplifies locating a route policy, and simplifies troubleshooting. **Example 11-65** demonstrates this concept with side-by-side configurations from just three BGP peers.

### **Example 11-65** With and Without RPL Parameterization Configuration

[Click here to view code image](#)



## Regular

```
! Some configuration omitted for brevity
```

```
router bgp 200
  neighbor 10.1.1.1
    address-family ipv4 unicast
    route-policy PEERING-ONE in
  !
  neighbor 10.2.2.2
    address-family ipv4 unicast
    route-policy PEERING-TWO in
  !
  neighbor 10.3.3.3
    address-family ipv4 unicast
    route-policy PEERING-THREE in
  !
  route-policy PEERING-ONE
    if as-path neighbor-is in ('1') then
      if destination in ('10.10.1.0/24') then
        pass
      endif
    endif
  end-policy
  !
  route-policy PEERING-TWO
    if as-path neighbor-is in ('2') then
      if destination in ('10.20.2.0/24') then
        pass
      endif
    endif
  end-policy
  !
  route-policy PEERING-THREE
    if as-path neighbor-is in ('3') then
      if destination in ('10.30.3.0/24') then
        pass
      endif
    endif
  end-policy
```

### With Parameterization

```
! Some configuration omitted for brevity
router bgp 200
  neighbor 10.1.1.1
    address-family ipv4 unicast
      route-policy PEERING(1, 10.10.1.0/24) in
    !
  neighbor 10.2.2.2
    address-family ipv4 unicast
      route-policy PEERING(2, 10.20.2.0/24) in
    !
  neighbor 10.3.3.3
    address-family ipv4 unicast
      route-policy PEERING(3, 10.30.3.0/24) in
    !
  route-policy PEERING($PATH, $NETWORK)
    if as-path neighbor-is '$PATH' then
      if destination in $NETWORK then
        pass
      endif
    endif
  end-policy
```

### Route Policy Nesting

RPL allows for the reuse of route policies in a hierarchically format by allowing one route policy to invoke a second route policy. This concept allows for common block of logic to be reused by other route policy and providing scalability to the RPL architecture.

**Example 11-66** demonstrates the concept of RPL nesting. The route policy PARENT is attached to the routing protocol, and invokes the route policy CHILD. The CHILD policy invokes the route policy GRANDCHILD that executes a pass, returns execution to the CHILD route policy, which then returns execution to the PARENT route policy.

### Example 11-66 Route Policy Nesting

**Click here to view code image**

```
route-policy GRANDCHILD
  pass
end-policy

route-policy CHILD
  apply GRANDCHILD
end-policy

route-policy PARENT
  apply Child
end-if
```

#### Note

The implied default drop executes only on the route policy that is attached to, and tickets are carried from child to parent policies so that the default drop does not occur at the end of policy execution.

**Example 11-67** demonstrates route policy nesting, where the route policy PARENT-AS100 invokes the route policy CHILD-RFC1918. The logic is maintained while shrinking the size of the parent route policy.

### **Example 11-67** *Route Policy Nesting Example*

[Click here to view code image](#)

```
route-policy PARENT-AS100
  apply CHILD-RFC1918
  if as-path originates-from '100' then
    prepend as-path most-recent
  endif
end-policy
!
route-policy CHILD-RFC1918
  if destination in (10.0.0.0/8 ge 8, 172.16.0.0/12 ge 12, 192.168.0.0/16 ge 16)
  then
    drop
  endif
  pass
end-policy
```

RPL supports nesting with parameterization, as shown in [Example 11-68](#).

### **Example 11-68** *Route Policy Nesting Example*

[Click here to view code image](#)

```
route-policy CHILD ($MED-PARAM,$ORG-PARAM)
  set med $MED-PARAM
  set origin $ORG-PARAM
end-policy

route-policy PARENT
  apply CHILD (10, incomplete)
end-policy
```

#### **Original Value**

**Example 11-69** demonstrates a route policy that modifies the MED, and then drops packets that match a specific MED value. In the route policy, a route with a MED of 100 changes to a MED of 200, and later any route with a MED of 200 is dropped. Human intuition implies that routes with a MED of 100 or 200 would drop.

### **Example 11-69** *Matching on Changed Attributes*

[Click here to view code image](#)

```
route-policy ORIGINAL-VALUES
  if med eq 100 then
    set med 200
  endif
  if med eq 200 then
    drop
  endif
end-policy
```

To prevent unintended actions from executing in a route-policy, IOS XR performs conditional matches only on the original values. A conditional match does not occur on intermediary values during the route policy processing. In [Example 11-69](#), only the original routes with a MED of 200 are dropped; the routes with values set to 200 are not dropped.

### Editors

Once a route policy, prefix set, AS path set, or community set is committed to the running configuration, future modifications replace the existing configuration as shown in [Example 11-70](#). The command **abort** will exit from the configuration mode.

### Example 11-70 Warning to Prevent Overwriting an Existing RPL

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1(config)#route-policy PASSALL
% WARNING: Policy object route-policy PASSALL' exists! Reconfiguring it via CLI will
replace current definition. Use 'abort to cancel.
```

IOS XR provides three text editors (Emacs, Nano, and Vim) for modification of policy sets and route policies without completely replacing the existing configuration. There is no context-sensitive help to assist with the changes inside the editors, but simple typos are easy to correct. Upon saving the changes in the editor, IOS XR will prompt to commit the changes, so that the change still appears in the configuration commit list.

The EXEC command **edit {as-path-set | community-set | prefix-set | route-policy} name {emacs | nano | vim}** will launch the appropriate editor and load the appropriate configuration. [Example 11-71](#) demonstrates this concept.

### Example 11-71 Modification of Existing RPL Component

[Click here to view code image](#)

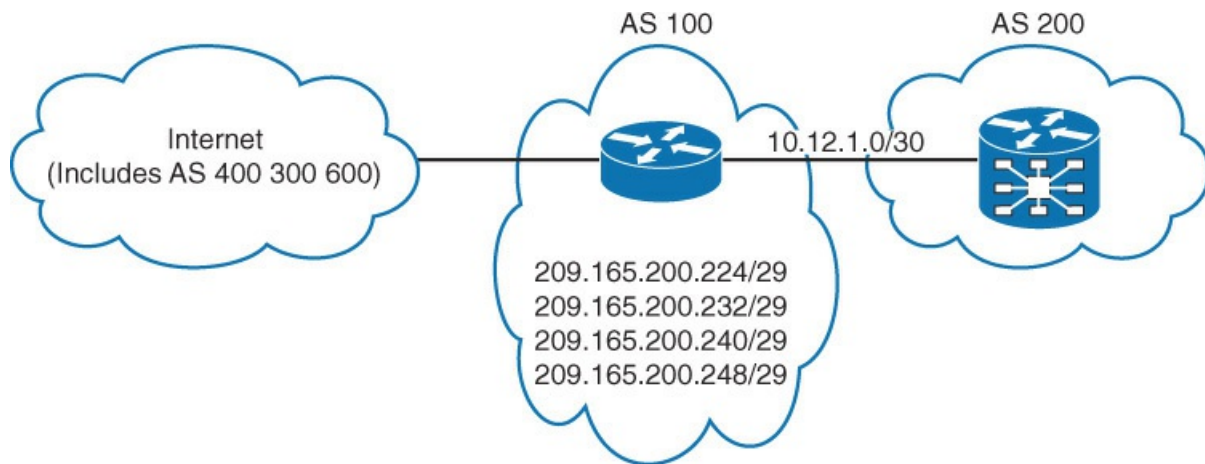
```
RP/0/0/CPU0:XR1#edit route-policy PASSALL nano
[OK]
Proceed with commit (yes/no/cancel)? [cancel]:
```

#### Note

Any change to a route policy requires that the change is acceptable for all active attach points and is checked during the commit.

## RPL Examples

To ensure that prefix sets, AS path sets, and route policies are fully understood, this section provides a brief scenario to combine all the common components into a sample BGP topology. [Figure 11-10](#) shows AS200 connected to AS100, which then connects to the Internet.



**Figure 11-10** Topology Example

[Example 11-72](#) shows AS200's BGP table with a standard external BGP (eBGP) peering to AS200.

### Example 11-72 Conditional Match and Action Statement

[Click here to view code image](#)

```
RP/0/0/CPU0:AS200#show bgp ipv4 unicast
! Output omitted for brevity
  Network          Next Hop    Metric LocPrf Weight Path
*> 209.165.200.224/29 10.12.1.1    100           0 100 400 300 600 i
*> 209.165.200.232/29 10.12.1.1    200           0 100 400 300 i
*> 209.165.200.240/29 10.12.1.1    300           0 100 400 i
*> 209.165.200.248/29 10.12.1.1    400           0 100 i

Processed 4 prefixes, 4 paths
```

[Example 11-73](#) provides the configuration for the AS200 router to match the routing policy. AS200's routing policy states that the route 209.165.200.248/29 should be blocked and that any routes that contain AS300 in the AS\_Path should have the local preference set to 500.

### Example 11-73 AS200's Router Configuration with AS\_Path and Prefix Sets

[Click here to view code image](#)

```

! The route-map is attached to the BGP neighbor in the appropriate address-family
router bgp 200
  neighbor 10.12.1.1
    address-family ipv4 unicast
      route-policy SAMPLE in
! The prefix-set is created to match the prefix that is to be blocked
prefix-set PREFIX-SET-BLOCK
  209.165.200.248/29
end-set
! The AS-Path is created to match the any prefix that passes through AS300
as-path-set AS-PATH-SET
  passes-through '300'
end-set
! The route policy provides the logic for AS200
route-policy SAMPLE
  if destination in PREFIX-SET-BLOCK then
    drop
  elseif as-path in AS-PATH-SET then
    set local-preference 500
  else
    pass
  endif
end-policy

```

**Example 11-74** provides the same configuration except for the route policy uses inline values.

### **Example 11-74** *AS200's Router Configuration with Inline Route Policy*

[Click here to view code image](#)

```

router bgp 200
  neighbor 10.12.1.1
    address-family ipv4 unicast
      route-policy SAMPLE in
!
route-policy SAMPLE
  if destination in (209.165.200.248/29) then
    drop
  elseif as-path passes-through '300' then
    set local-preference 500
  else
    pass
  endif
end-policy

```

**Example 11-75** provides the BGP table after the SAMPLE route policy was applied to the BGP neighbor 10.12.1.1 for inbound route policy processing.

### **Example 11-75** *AS200 BGP Table After Inbound SAMPLE Route Map*

[Click here to view code image](#)

```
RP/0/0/CPU0:AS200#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 209.165.200.224/29	10.12.1.1	100	500	0	100 400 300 600 i
*> 209.165.200.232/29	10.12.1.1	200	500	0	100 400 300 i
*> 209.165.200.240/29	10.12.1.1	300		0	100 400 i

```
Processed 3 prefixes, 3 paths
```

## RPL Verification

IOS XR includes a variety of commands to further analyze RPL components. The command **show rpl** displays all the route policies, prefix sets, community sets, and AS path sets that are configured on a router.

The status of the route policy is shown with the command **show rpl route-policy states**.

Three states exist for a route policy:

- **Active:** Any route policy that is attached directly to a routing component or that is referenced by an active parent route policy
- **Unused:** Any route policy that is not attached directly to a routing component
- **Inactive:** Any route policy that is not attached directly to a routing component, or that has a parent route policy in an unused or inactive state

Example 11-76 demonstrates the output of the **show rpl route-policy states** command for XR1. Notice that the route policy PARENT is in an unused state and that the route policy CHILD is in an inactive state because it is referenced by the PARENT route policy that is not attached to a routing component.

### Example 11-76 Display of RPL States

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show rpl route-policy states
```

```
! Output omitted for brevity
```

```
The following policies are (ACTIVE)
```

```
-----  
route-policy PASSALL  
  pass  
end-policy  
!  
route-policy STATIC-2-OSPF  
  if destination in (192.168.2.2) then  
    pass  
  endif  
end-policy  
!
```

```
The following policies are (INACTIVE)
```

```
-----  
route-policy CHILD  
  pass  
end-policy  
!
```

```
The following policies are (UNUSED)
```

```
-----  
route-policy PARENT  
  apply CHILD  
end-policy  
!
```

Attachment points for active route policies are shown with the command **show rpl route-policy route-policy-name attachpoints**.

**Example 11-77** displays the attachment point for the PASSALL route policy. The output shows that the route policy is attached to the BGP neighbor 172.19.0.2 for the IPv4 address family for inbound and outbound route processing.

### **Example 11-77** *PASSALL Route Policy Attachment Points*

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show rpl route-policy PASSALL attachpoints
```

```
BGP Attachpoint: Neighbor
```

```
Neighbor/Group          type  afi/safi  in/out  vrf name  bound by  
-----  
172.19.0.2              --    IPv4/uni  out     default  PASSALL  
172.19.0.2              --    IPv4/uni  in      default  PASSALL
```



## Redistribution RPL Verification

Example 11-78 displays the attachment point for the STATIC-2-OSPF route policy. The output shows that the route policy is attached to the OSPFv2 process 1 for redistribution of static routes. Redistribution is covered in detail in [Chapter 13](#), “Route Redistribution.”

### Example 11-78 STATIC-2-OSPF Route Policy Attachment Points

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show rpl route-policy STATIC-2-OSPF attachpoints

OSPFv2 Attachpoint: Redistribution to OSPFv2 process

source      instance  to process vrf name  bound by
-----
static      ----      1          default  STATIC-2-0
```

## BGP RPL Verification

A quick method to identify routes that are filtered from the BGP table uses the command **show bgp address-family address-family-modifier route-policy route-policy-name**. Example 11-79 shows the unmodified BGP table using the PASSALL route policy that will be used for comparison in the next example.

### Example 11-79 BGP Table with PASSALL Route Policy

[Click here to view code image](#)

```
RP/0/0/CPU0:AS200#show bgp ipv4 unicast
! Output omitted for brevity

Network          Next Hop          Metric LocPrf Weight Path
* > 209.165.200.224/29 10.12.1.1         100           0 100 400 300 600 i
* > 209.165.200.232/29 10.12.1.1         200           0 100 400 300 i
* > 209.165.200.240/29 10.12.1.1         300           0 100 400 i
* > 209.165.200.248/29 10.12.1.1         400           0 100 i

Processed 4 prefixes, 4 paths
```

Example 11-80 provides a preview of the route policy against the BGP table. Although the 209.165.200.248/19 prefix was filtered, the local preference does not reflect the value of 500 that should have been set on the 209.165.200.224/29 and 209.165.200.232/29 network prefixes.

### Example 11-80 BGP Table Preview with SAMPLE Route Policy

[Click here to view code image](#)

```
RP/0/0/CPU0:AS200#show bgp ipv4 unicast route-policy SAMPLE
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 209.165.200.224/29	10.12.1.1	100		0 100 400 300 600	i
*> 209.165.200.232/29	10.12.1.1	200		0 100 400 300	i
*> 209.165.200.240/29	10.12.1.1	300		0 100 400	i

```
Processed 3 prefixes, 3 paths
```

The command **show bgp policy route-policy route-policy-name** displays advertisements under a proposed route policy. [Example 11-81](#) demonstrates the command that displays output in a different format so that all the BGP path attributes can be seen before and after the proposed route policy.

### Example 11-81 Viewing BGP Advertisements with Proposed Route Policy

[Click here to view code image](#)

```
RP/0/0/CPU0:AS200#show bgp policy route-policy SAMPLE
```

```
! Output omitted for brevity
```

```
209.165.200.224/29 is advertised to 10.12.1.1
```

```
Received Path ID 0, Local Path ID 1, version 12
```

```
Attributes after inbound policy was applied:
```

```
next hop: 10.12.1.1
```

```
MET ORG AS
```

```
origin: IGP neighbor as: 100 metric: 100
```

```
aspath: 100 400 300 600
```

```
Attributes after outbound policy was applied:
```

```
next hop: 10.12.1.2
```

```
MET ORG AS LOCAL
```

```
origin: IGP neighbor as: 100 metric: 100 local pref: 500
```

```
aspath: 200 100 400 300 600
```

```
209.165.200.232/29 is advertised to 10.12.1.1
```

```
! Output omitted for brevity
```

```
Attributes after outbound policy was applied:
```

```
next hop: 10.12.1.2
```

```
MET ORG AS LOCAL
```

```
origin: IGP neighbor as: 100 metric: 200 local pref: 500
```

```
aspath: 200 100 400 300
```

```
209.165.200.240/29 is advertised to 10.12.1.1
```

```
! Output omitted for brevity
```

```
Attributes after outbound policy was applied:
```

```
next hop: 10.12.1.2
```

```
MET ORG AS
```

```
! This prefix did not match, and the local preference was not added
```

```
origin: IGP neighbor as: 100 metric: 300
```

```
aspath: 200 100 400
```

## SUMMARY

This chapter covered several important building block features that are necessary for manipulating routes, within a routing protocol.

- ACLs provide a method of identifying networks. Extended ACLs provide the ability to select the network and advertising router for IGP protocols, and provide the ability to use wildcards for the network and subnet mask for BGP routes.
- Prefix lists and prefix sets identify networks based on the high-order bit pattern, high-order bit count, and required prefix-length requirements.
- Regular expressions (regex) provide a method of parsing output in a systematic way. Regex is commonly used for BGP filtering, but can be used in IOS and IOS XR CLI for parsing output, too.
- Route maps can filter routes similar to an ACL and provide the capability to modify route attributes. Route maps consist of sequence numbers, matching criteria, processing action, and optional modifying actions. If matching criteria is not specified, all routes qualify for that route map sequence. Multiple conditional matching requirements of the same type are a Boolean **or**, and multiple conditional matching requirements of different types are a Boolean **and**.
- IOS XR uses route policy language (RPL) for filtering or manipulating routes. RPL consists of policy sets (AS\_Path, prefix, community) and route policies. Route policies use a programmatic structure and provide scalability through nesting and parameterization.

## REFERENCES IN THIS CHAPTER

Tahir, Mobeen, et al. *Cisco IOS XR Fundamentals*. Indianapolis: Cisco Press, 2009.

Cisco. Cisco IOS Software Configuration Guides. <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides. <http://www.cisco.com>

## Chapter 12. Advanced Route Manipulation

This chapter covers the following topics:

- Conditional routing of packets
- Administrative distance manipulation
- Route filtering

This chapter explores route manipulation using conditional routing, administrative distance (AD), and route filtering. Traffic manipulation is commonly used for security purposes, traffic manipulation, conservation of memory, or during redistribution.

### CONDITIONAL ROUTING OF PACKETS

A router makes forwarding decisions based on the destination address of the IP packet. Some scenarios accommodate other factors, such as packet length or source address, when deciding where the router should forward a packet.

IOS policy-based routing (PBR) and IOS XR access-list-based forwarding (ABF) allow for conditional routing of packets based on packet characteristics besides the destination IP address.

PBR and ABF provide the following capabilities:

- Routing by protocol type (ICMP, TCP, UDP, and so on)
- Routing by source IP address, destination IP address, or both
- Manually assigning different network paths to the same destination based on tolerance for latency, link speed or utilization for specific transient traffic

Drawbacks of conditional routing include the following:

- Administrative burden in scalability
- Lack of network intelligence
- Troubleshooting complexity

Packets are examined for PBR and ABF processing as they are received on the router interface. IOS local PBR policies can also identify traffic originating from the router, but there is not an equivalent functionality on IOS XR.

PBR and ABF verify the existence of the next-hop IP address and then forward packets using the specified next-hop address. Additional next-hop addresses can be configured so that if the first next-hop address is not in the Routing Information Base (RIB), the secondary next-hop addresses can be used. If none of the specified next-hop addresses exist in the routing table, the packets are not conditionally forward.

---

Note

PBR and ABF policies do not modify the RIB because the policies are not universal for all packets. This can often complicate troubleshooting because the routing table displays the next-hop address learned from the routing protocol but does not account for a different next-hop address for the conditional traffic.

### Policy-Based Routing Configuration

IOS PBR configuration uses a route map with **match** and **set** statements that are then attached to the inbound interface. The following steps are used:

#### Step 1. Define a route map.

The route map is configured with the command **route-map** *route-map-name* [**permit** | **deny**] [*sequence-number*]

#### Step 2. Identify the conditional match criteria.

The conditional match criteria can be based on packet length with the command **match length** *minimum-length maximum-length*, or by using the packet IP address fields with an ACL using the command **match ip address** {*access-list-number* | *acl-name*}

#### Step 3. Specify the next hop.

The command **set ip** [**default**] **next-hop** *ip-address* [... *ip-address*] is used to specify one or more next-hops for packets that match the criteria.

The optional **default** keyword changes the behavior so that the next-hop address specified by the route map is only used if the destination address does not exist in the RIB. If a viable route exists in the RIB, that route is the next-hop address used for forwarding the packet.

#### Step 4. Apply the route map to the inbound interface.

The route map is applied with the interface parameter command **ip policy route-map** *route-map-name*

### Access-List-Based Forwarding Configuration

IOS XR ABF configuration uses an access control list (ACL) that specifies the next-hop IP addresses. Unlike IOS PBR, ABF uses hardware programming for the conditional routing. The following steps are used:

#### Step 1. Define an access list.

The ACL is configured with the command **ipv4 access-list** *acl-name*.

#### Step 2. Specify the next-hop addresses in the ACL access control entry (ACE).

Define the appropriate information for matching of packets with the command [*sequence-number*] **permit** *protocol source-ip source-wildcard destination destination-wildcard* [**default**] **nexthop1** **ipv4** *ip-address* [**nexthop2** **ipv4** *ip-address*] [**nexthop3** **ipv4** *ip-address*]. The **nexthop** commands define the conditional forwarding used by the ACL.

The optional **default** keyword changes the behavior so that the next-hop address specified is used only if the destination address does not have a matching route in the RIB. If the route exists in the RIB, that route is the next-hop address used.

ABF supports only up to three next-hop IP addresses; unlike PBR, however, ABF does not support the criteria of matching on packet length.

### Step 3. Apply the ACL to an interface in the ingress direction.

The ACL is applied by configuring the interface parameter command **ipv4 access-group acl-name ingress**.

#### Note

ABF is still an ACL that will drop any nonmatching traffic with an implicit deny. A **permit ipv4 any any** is highly recommended if the ACL is used to only conditional route packets.

Figure 12-1 provides a topology example for illustrating PBR and ABF concepts. R1, R2, XR5, and R6 are using the Open Shortest Path First (OSPF) Protocol to route traffic between R1 and R6. R3 and R4 use static routes to ensure full connectivity between all routers. Traffic between R2 and XR5 flows across the 10.25.1.0/24 network because R3 and R4 are not participating in the OSPF routing domain.

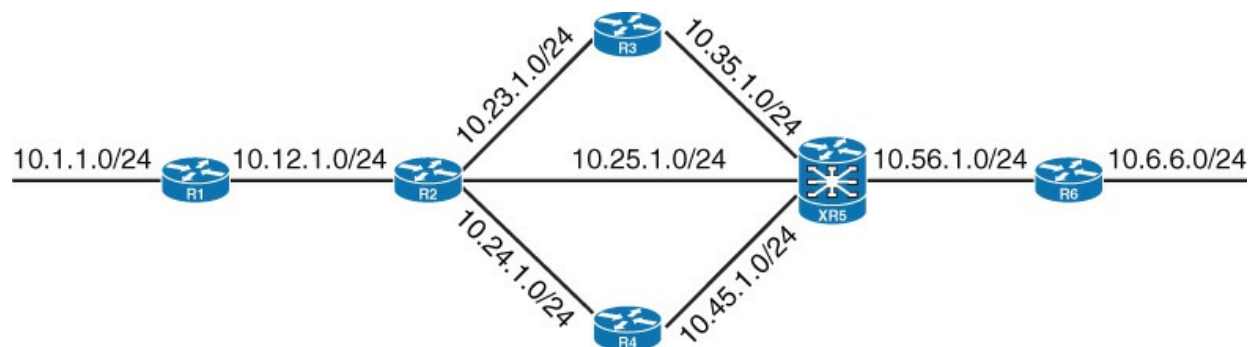


Figure 12-1 PBR and ABF Next-Hop Topology

Example 12-1 shows the normal traffic path using traceroute between the 10.1.1.0/24 and 10.6.6.0/24 networks without PBR or ABF configured.

### Example 12-1 Traceroute for Normal Traffic Flow

[Click here to view code image](#)

```
R1#traceroute 10.6.6.6 source 10.1.1.1
 1 10.12.1.2 0 msec 0 msec 0 msec
 2 10.25.1.5 1 msec 1 msec 1 msec
 3 10.56.1.6 6 msec * 2 msec
```

```
R6#traceroute 10.1.1.1 source 10.6.6.6
 1 10.56.1.5 2 msec 1 msec 1 msec
 2 10.25.1.2 2 msec 1 msec 2 msec
 3 10.12.1.1 2 msec * 3 msec
```

**Example 12-2** provides sample IOS PBR configuration for 10.1.1.0/24 network traffic destined for the 10.6.6.0/24 network, which will use the 10.23.1.0/24 link with a next hop to 10.23.1.3. IOS XR ABF configuration accommodates for return traffic (10.6.6.0/24 to 10.1.1.0/24) to use the same link (10.23.1.0/24).

### **Example 12-2** IOS PBR Next Hop and IOS XR ABF Next Hop

[Click here to view code image](#)

```
R2
ip access-list extended ACL-PBR
 permit ip 10.1.1.0 0.0.0.255 10.6.6.0 0.0.0.255
!
route-map NEXTHOP permit 10
 match ip address ACL-PBR
 set ip next-hop 10.23.1.3
!
interface Ethernet1/0
 ip address 10.12.1.2 255.255.255.0
 ip policy route-map NEXTHOP
```

```
XR5
ipv4 access-list NEXTHOP
 10 permit ipv4 10.6.6.0/24 10.1.1.0/24 nexthop1 ipv4 10.45.1.4
 20 permit ipv4 any any

interface GigabitEthernet0/0/0/4
 ipv4 address 10.56.1.5 255.255.255.0
 ipv4 access-group NEXTHOP ingress
```

**Example 12-3** shows the traffic path between the 10.1.1.0/24 and 10.6.6.0/24 networks after applying the conditional route forwarding policies. Notice that the path does not use the 10.25.1.0/24 network as shown earlier.

### **Example 12-3** R1 to R6 Paths Demonstrating PBR and ABF

[Click here to view code image](#)

```
R1#traceroute 10.6.6.6 source 10.1.1.1
 1 10.12.1.2 0 msec 0 msec 0 msec
 2 10.23.1.3 1 msec 0 msec 3 msec
 3 10.35.1.5 3 msec 2 msec 0 msec
 4 10.56.1.6 2 msec * 3 msec
```

```
R6#traceroute 10.1.1.1 source 10.6.6.6
 1 10.56.1.5 2 msec 1 msec 1 msec
 2 10.35.1.3 0 msec 1 msec 2 msec
 3 10.23.1.2 2 msec 1 msec 2 msec
 4 10.12.1.1 2 msec * 3 msec
```

[Example 12-4](#) demonstrates that applying a PBR and ABF configuration does not modify the routing table. Conditional packet forwarding is outside the view of the RIB

### Example 12-4 R2 and XR5 Routing Table

[Click here to view code image](#)

```
R2#show ip route
! Output omitted for brevity
O      10.6.6.0/24 [110/21] via 10.25.1.5, 00:43:05, GigabitEthernet0/0
```

```
RP/0/0/CPU0:XR5#show route
! Output omitted for brevity
O      10.1.1.0/24 [110/21] via 10.25.1.2, 01:47:30, GigabitEthernet0/0/0/3
```

#### Note

PBR may process on hardware dependent upon the platform and conditional matching conditions. If PBR does not process on hardware, it will process on the RP consuming additional CPU and memory whereas ABF always processes on hardware application-specific integrated circuits (ASICs).

### Local PBR

Packets originated by the router are not policy routed by default. IOS routers include a feature for policy routing the locally generated traffic through *local PBR*. Local PBR policies are applied to the router with the global configuration command **ip local policy route-map-name**.

Revisiting the topology in [Figure 12-1](#), R2 maintains a requirement that traffic originating from its Loopback 0 interface (192.168.2.2) destined to 10.6.6.0/24 should use the 10.23.1.0/24 link instead of the 10.25.1.0/24 link. [Example 12-5](#) demonstrates the local PBR configuration needed to accomplish this.

### Example 12-5 Local PBR Configuration

[Click here to view code image](#)



```

R2
ip access-list extended ACL-LOCAL-PBR
 permit permit ip 192.168.2.2 0.0.0.0 10.6.6.0 0.0.0.255
!
route-map LOCAL-PBR permit 10
 match ip address ACL-LOCAL-PBR
 set ip next-hop 10.23.1.3

ip local policy route-map LOCAL-PBR

```

Example 12-6 provides verification that the traffic between 192.168.2.2 and 10.6.6.0/24 network uses the 10.23.1.0/24 network, and all other traffic will use the 10.25.1.0/24 network.

### Example 12-6 Local PBR Verification

[Click here to view code image](#)

```

R2#traceroute 10.6.6.6
 1 10.25.1.5 2 msec 1 msec 1 msec
 2 10.56.1.6 2 msec * 1 msec

R2#traceroute 10.6.6.6 source loopback 0
 1 10.23.1.3 2 msec 0 msec 1 msec
 2 10.35.1.5 1 msec 0 msec 0 msec
 3 10.56.1.6 2 msec * 2 msec

```

## ADMINISTRATIVE DISTANCE

A router uses administrative distance (AD) to determine which route to install into the RIB when there are multiple paths to the same destination from multiple routing protocols. The AD indicates the reliability of a routing protocol using a value between 0–255, with the lower value deemed more reliable over a higher value. A value of 255 is undesirable and will not install into the RIB. Table 12-1 provides the default AD for the common routing protocol.

Connection/Routing Protocol	Default AD
Connected	0
Static	1
eBGP	20
EIGRP (internal )	90
OSPF	110
IS-IS	115
RIP	120
EIGRP (external)	170
iBGP	200

Table 12-1 Default AD Values

In some network designs, the path provided by the lower AD is not the optimal path. Raising or

lowering the AD within a routing protocol is a technique for modifying, which route is installed into the RIB. Modification of the AD is a router-specific function and could lead to routing loops if the change is not consistently deployed throughout the network.

Figure 12-2 demonstrates a network topology using Enhanced Interior Gateway Routing Protocol (EIGRP) on R1, R3, R4, and R5; and OSPF on R1, R2, and R5. The path between R5 to R1 will use the slower serial links over the faster Ethernet links because EIGRP's AD (90) is lower than OSPF's AD (110). Modifying the AD on R5 so that OSPF's AD is lower than EIGRP will ensure that R5 uses the faster Ethernet links when forwarding traffic to R1. To ensure that the return traffic uses the same path, the AD will need to be modified on R1 as well.

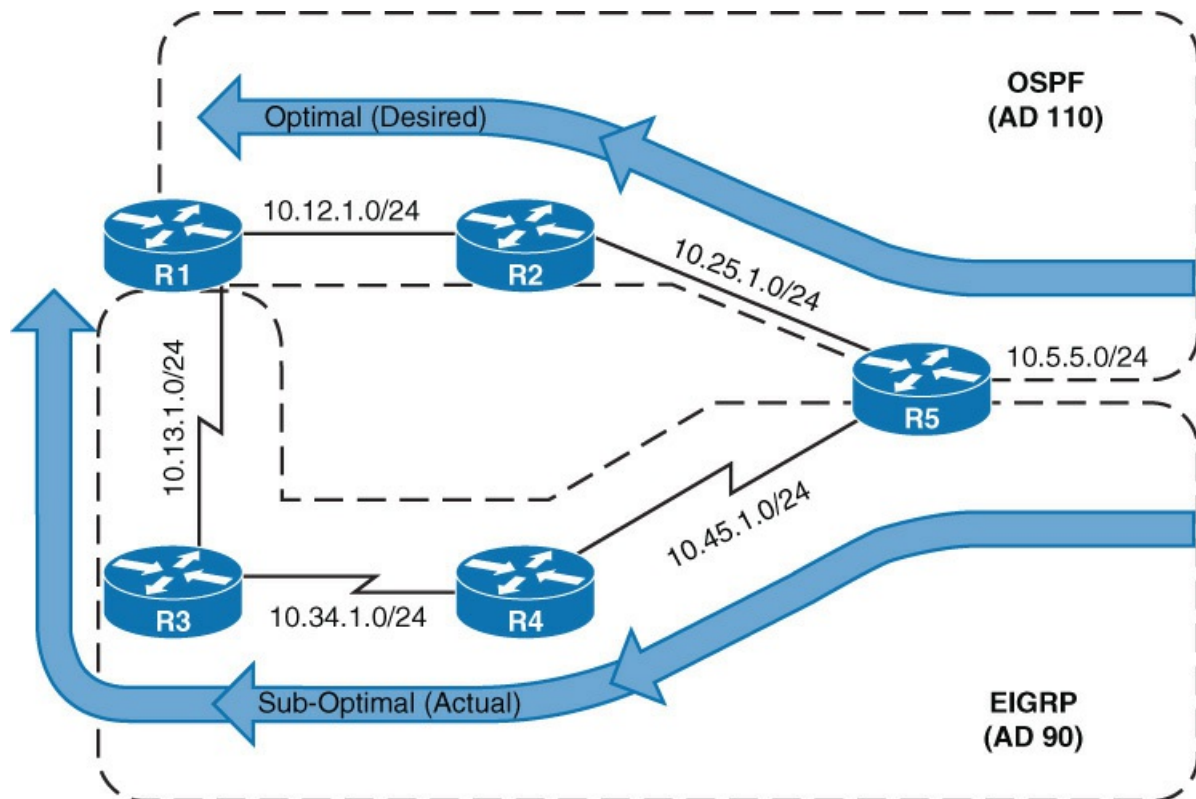


Figure 12-2 Suboptimal Routing Because of Administrative Distance

Note

Remember that each protocol uses its own logic for identifying the best path it sends to the Routing Information Base (RIB). It is possible for the router to prefer a path with a higher AD due to the manner that the protocol's best path algorithm executes. This could cause unintended results when a second routing protocol has the same route.

For example, suppose that a prefix is available via OSPF (110 AD), external Border Gateway Protocol (eBGP) (20 AD), and internal BGP (iBGP) (200 AD). If the BGP best path algorithm selects the iBGP path over the eBGP path, the router will install the route learned via OSPF into the RIB because the AD of 110 is lower than 200. Although this conflicts with the logic from a pure AD perspective, remember that AD determines preference during installation into the RIB between competing routing protocols and not within a protocol.

The following sections explain how to modify the AD for each protocol.

### Modifying EIGRP AD

EIGRP already differentiates between routes learned from within the autonomous system and routes learned from outside of the autonomous system by assigning different AD:

**Internal EIGRP:** 90

## External EIGRP: 170

To modify the default AD IOS routers use the EIGRP configuration command **distance eigrp ad-internal ad-external**, and IOS XR routers use the EIGRP configuration command **distance ad-internal ad-external**. Valid values for the AD are between 1 and 255; a value of 255 stops the installation of the route into the RIB.

IOS routers allow selective AD modification for specific internal networks with the command **distance ad source-ip source-ip-wildcard [acl-number | acl-name]**. The *source-ip* option restricts the modification to routes in the EIGRP table that were learned from a specific router, and the optional ACL restricts to a specific network prefix.

### Note

EIGRP does not allow the selective AD modification based on prefixes for external EIGRP routes.

**Example 12-7** demonstrates how to configure the AD for internal EIGRP routes to 205 and external EIGRP route to 210 for XR2 and R3.

### Example 12-7 EIGRP AD Manipulation Configuration

[Click here to view code image](#)

#### XR2

```
router eigrp 100
 address-family ipv4
   distance 205 210
 interface GigabitEthernet0/0/0/0
 !
 interface GigabitEthernet0/0/0/2
```

#### R3

```
router eigrp 100
 network 10.23.0.0 0.0.255.255
 network 10.34.0.0 0.0.255.255
 distance eigrp 205 210
```

**Example 12-8** provides a snippet of the routing table for XR2 and R3. Notice that the internal EIGRP AD is now 205 instead of the default 90 and that the external EIGRP distance is 210 instead of the default value of 170.

### Example 12-8 EIGRP AD Manipulation Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route eigrp
```

```
! Output omitted for brevity
```

```
D 172.16.3.0/24 [205/130816] via 10.12.1.1, 00:02:08, GigabitEthernet0/0/0/0
```

```
D EX 172.16.4.0/24 [210/3072] via 10.12.1.1, 00:02:08, GigabitEthernet0/0/0/0
```

```
R3#show ip route eigrp
```

```
! Output omitted for brevity
```

```
D 172.20.3.0 [205/130816] via 10.34.1.4, 00:00:56, GigabitEthernet0/1
```

```
D EX 172.20.4.0 [210/3072] via 10.34.1.4, 00:00:56, GigabitEthernet0/1
```

## Modifying OSPF AD

OSPF uses the same default AD 110 value for routes learned from within the OSPF routing domain and for routes learned from outside of the OSPF routing domain.

IOS and IOS XR routers can modify the default AD with the OSPF configuration command **distance ospf {external | interarea | intra-area} ad**. The command allows for setting different AD for each OSPF network type.

Note

The perspective of **interarea** and **intra-area** can change from router to router, and caution should be used if the two values do not match.

IOS and IOS XR routers allow selective AD modification for specific networks with the command **distance ad source-ip source-ip-wildcard [acl-number | acl-name]**. The *source-ip* option restricts the modification to routes in the OSPF LSDB learned from the advertising router of the LSA. The *source-ip* address fields match the RID for the advertising route. The optional ACL restricts to a specific network prefix.

Example 12-9 demonstrates how to modify XR2 and R3 default OSPF AD so that

■ **Intra-area routes:** 111

■ **Interarea routes:** 112

■ **External routes:** 180

**Example 12-9** *OSPF Customized AD Configuration*

[Click here to view code image](#)

#### XR2

```
router ospf 1
  distance ospf intra-area 111 inter-area 112 external 180
  area 0
    interface GigabitEthernet0/0/0/2
    !
  area 1
    interface GigabitEthernet0/0/0/0
```

#### R3

```
router ospf 1
  distance ospf intra-area 111 inter-area 112 external 180 distance eigrp 205 210
  network 10.23.0.0 0.0.255.255
  network 10.34.0.0 0.0.255.255
```

**Example 12-10** provides a snippet of the routing table for XR2 and R3. Notice that the AD for intra-area routes is 111, the AD for interarea routes is 112, and the AD for external routes is 180 instead of the default AD of 110 for all three types.

### Example 12-10 OSPF AD Manipulation Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route ospf
```

```
! Output omitted for brevity
```

```
O    172.16.3.0/24 [111/2] via 10.12.1.1, 00:08:04, GigabitEthernet0/0/0/0
O IA 172.20.1.0/24 [112/3] via 10.23.1.3, 00:08:04, GigabitEthernet0/0/0/2
O E2 172.20.2.0/24 [180/20] via 10.23.1.3, 00:08:04, GigabitEthernet0/0/0/2
```

```
R3#show ip route eigrp
```

```
! Output omitted for brevity
```

```
O        10.45.1.0/24 [111/20] via 10.34.1.4, 00:08:42, GigabitEthernet0/0
O IA     172.16.1.0 [112/3] via 10.23.1.2, 00:08:42, GigabitEthernet0/0
O E2    172.20.2.0 [180/20] via 10.34.1.4, 00:08:42, GigabitEthernet0/1
```

### Modifying IS-IS AD

It is not possible to modify the AD between Level 1 (L1), Level 2 (L2), internal or external Intermediate System-to-Intermediate System (IS-IS) routes. The AD for all route types are modified with the IOS command **distance ad {ip | source-ip source-ip-wildcard [acl-number | acl-name]**. The optional ACLs allow for conditional matching of prefixes for selective changes of AD. IOS XR routers allow selective AD modification for specific route prefixes with the command **distance ad [prefix/prefix-length [prefix-list-name]]**. The *prefix/prefix-length* identifies the source of the route, and the *prefix-list-name* identifies the routes from the source.

**Example 12-11** displays the source IP address used by a router. The link-state packet (LSP) IP value of XR1 (192.168.1.1) is used for specifying a source address for the AD modification. The value did not come from any of the IP reachability TLVs (Type Length-Value) because they support multiple IP addresses.

## Example 12-11 IS-IS Source ID Lookup

[Click here to view code image](#)

```
R4#show isis database detail XR1.00-00
! Output omitted for brevity

IS-IS Level-1 Link State Database:
XR1.00-00          0x00000005  0xFFE5      999          0/0/0
  Hostname: XR1
  IP Address: 192.168.1.1
```

Example 12-12 demonstrates the configuration for XR1 that sets the AD to 230 for IS-IS routes learned via routers in the 192.168.0.0/16 range with /32 prefix-length. R3's configuration changes the AD to 230 for routes learned via 192.168.1.1 or 192.168.2.2 with a /32 prefix length.

## Example 12-12 IS-IS AD Manipulation Configuration

[Click here to view code image](#)

```
XR1
ipv4 prefix-list LOOPBACKS
 10 permit 192.168.0.0/16 eq 32
!
router isis ISIS
 address-family ipv4 unicast
  distance 230 192.168.0.0/16 LOOPBACKS
```

```
R3
router isis
 net 49.0001.0000.0000.0003.00
 log-adjacency-changes
 distance 230 192.168.1.1 0.0.0.0 LOOPBACKS
 distance 230 192.168.2.2 0.0.0.0 LOOPBACKS
!
ip access-list standard LOOPBACKS
 permit 192.168.1.1
 permit 192.168.2.2
```

Example 12-13 verifies that only the AD for the loopback addresses are modified to 230.

## Example 12-13 IS-IS AD Manipulation Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route isis

i L1 10.23.1.0/24 [115/20] via 10.12.1.2, 00:00:03, GigabitEthernet0/0/0/0
i L1 192.168.2.2/32 [230/20] via 10.12.1.2, 00:00:03, GigabitEthernet0/0/0/0
i L1 192.168.3.3/32 [230/20] via 10.12.1.2, 00:00:03, GigabitEthernet0/0/0/0
```

```
R3#show ip route isis
! Output omitted for brevity

i L1    10.12.1.0/30 [115/20] via 10.23.1.2, 05:33:21, GigabitEthernet0/0
i L1    192.168.1.1 [230/20] via 10.23.1.2, 05:33:21, GigabitEthernet0/0
i L1    192.168.2.2 [230/20] via 10.23.1.2, 05:33:21, GigabitEthernet0/0
```

## Modifying BGP AD

BGP differentiates between routes learned from iBGP peers, routes learned from eBGP peers, and routes learned locally. IOS routers allow the modification of AD for routes received from a specific neighbor with the address family command **distance ad source-ip source-wildcard [acl-number | acl-name]**. IOS and IOS XR routers use the BGP configuration command **distance bgp external-ad internal-ad local-routes** to set the AD for each BGP network type.

### Note

Locally learned routes are from aggregate (summary) or backdoor networks. Routes advertised via the network statement use the AD setting for iBGP routes.

Figure 12-3 provides a sample BGP topology for demonstrating AD manipulation. XR2 and R3 advertise a summary prefix (10.0.0.0/8) so that local routes are present in the BGP table.

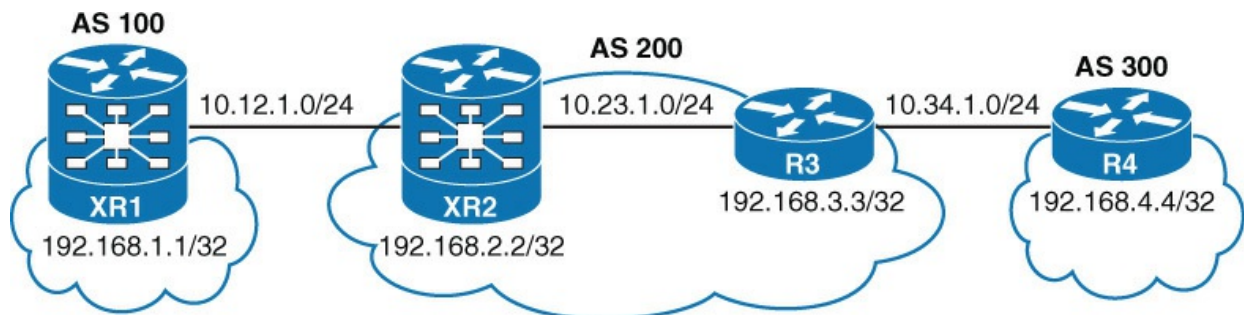


Figure 12-3 BGP AD Manipulation Topology

Example 12-14 provides the BGP configuration for XR2 and R3 to demonstrate the AD modification for BGP so that eBGP routes are set to 44, iBGP routes are set to 55, and locally learned routes are set to 66.

## Example 12-14 BGP AD Manipulation Configuration

[Click here to view code image](#)

```
XR2
router bgp 200
address-family ipv4 unicast
distance bgp 44 55 66
network 10.12.1.0/24
network 10.23.1.0/24
network 192.168.2.2/32
aggregate-address 10.0.0.0/8
```

### R3

```
router bgp 200
  address-family ipv4
    network 10.23.1.0 mask 255.255.255.0
    network 10.34.1.0 mask 255.255.255.0
    network 192.168.3.3 mask 255.255.255.255
    aggregate-address 10.0.0.0 255.0.0.0
    distance bgp 44 55 66
  exit-address-family
```

**Example 12-15** confirms that the eBGP routes are set to 44, iBGP routes are set to 55, and locally generated routes are set to 66.

### Example 12-15 BGP AD Manipulation Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route bgp
! Output omitted for brevity

B    10.0.0.0/8 [66/0] via 0.0.0.0, 00:01:07, Null0
B    10.34.1.0/24 [55/0] via 10.23.1.3, 00:00:55
B    192.168.1.1/32 [44/0] via 10.12.1.1, 00:00:10
B    192.168.3.3/32 [55/0] via 10.23.1.3, 00:00:55
B    192.168.4.4/32 [55/0] via 10.34.1.4, 00:00:53
```

```
R3#show ip route bgp
! Output omitted for brevity

B    10.0.0.0/8 [66/0] via 0.0.0.0, 00:00:21, Null0
B    10.12.1.0/24 [55/0] via 10.23.1.2, 00:37:49
B    192.168.1.1 [55/0] via 10.12.1.1, 00:37:44
B    192.168.2.2 [55/0] via 10.23.1.2, 00:37:49
B    192.168.4.4 [44/0] via 10.34.1.4, 00:38:34
```

#### Note

The AD does not change until the BGP session is reset with the neighbor that the route is learned from. BGP resets are explained at the end of this chapter.

## ROUTE FILTERING AND MANIPULATION

Route filtering is a method for selectively identifying routes that are advertised or received from neighbor routers. Route filtering may be used to manipulate traffic flows, reduce memory utilization, or to improve security. For example, it is common for Internet service providers (ISPs) to deploy route filters on BGP peerings to customers. Ensuring that only the customer routes are allowed over the peering prevents the customer from accidentally becoming a transit autonomous system on the Internet.

Filtering of routes within a routing protocol is accomplished with *distribute lists* for IOS nodes, and *route policies* for IOS XR nodes. Distributed lists typically match off an ACL (standard or extended), prefix list, route map, gateway, or the interface receiving or advertising the update.



ACLs and prefix lists match against the prefix, the gateway matches against the next-hop IP address, and specifying an interface restricts the filtering to updates received or sent through the specified interface. Route maps allow conditional matching to occur on other route characteristics.

Note

When a distribute list references an extended ACL, the source fields of the ACL refer to the advertising router, and the destination fields indicate the network, as demonstrated in the following sections.

### EIGRP Filtering by Prefix

EIGRP supports filtering of routes as they are received or advertised from an interface.

Figure 12-4 illustrates the concept that inbound filtering drops routes prior to the Diffusing Update Algorithm (DUAL) processing, and therefore the routes are not installed into the RIB because they are not known. However, if the filtering occurs during outbound route advertisement, the routes are processed by DUAL and do install into the local RIB of the advertising router.

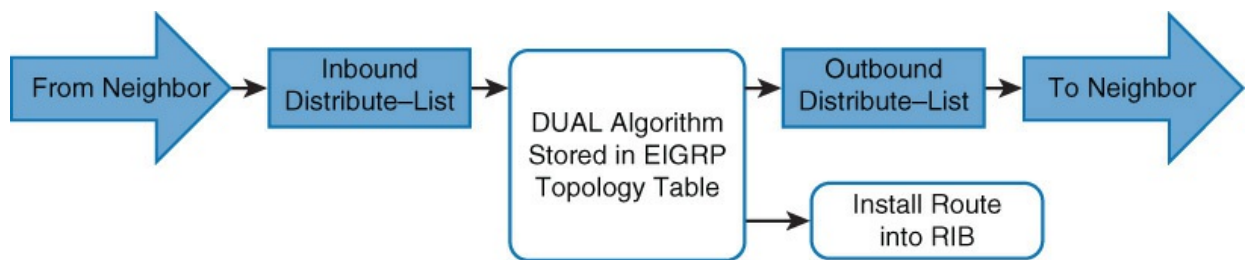


Figure 12-4 EIGRP Distribute List Filtering Logic

IOS routers use the EIGRP configuration command **distribute-list** {*acl-number* | *acl-name* | **prefix** *prefix-list-name* | **route-map** *route-map-name* | **gateway** *prefix-list-name*} [**gateway** *prefix-list-name*] {**in** | **out**} [*interface-type* *interface-number*] to filter routes. Prefixes that match against **deny** statements are filtered, and prefixes that match against a **permit** are passed. The **gateway** command can be used by itself or combined with a prefix list, ACL, or route map to restrict prefixes based on the next-hop forwarding address. Specifying an interface restricts the filtering to the interface that the route was received or advertised out of.

IOS XR routers use the EIGRP address-family configuration command **route-policy** *route-policy-name* [*parameters*] {**in** | **out**} to filter routes. Placing the configuration underneath a specific interface restricts the filtering to routes advertised and received on that specific interface.

Figure 12-5 illustrates an EIGRP network. XR2 and R3 will filter routes inbound and outbound to demonstrate how EIGRP can filter prefixes between neighbors.

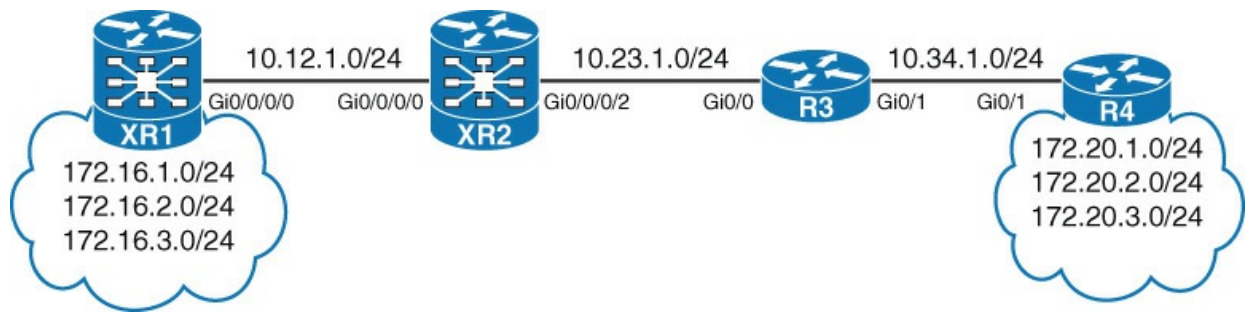


Figure 12-5 EIGRP Distribute List Filtering Topology

Example 12-16 provides the routing tables of XR2 and R3 before the route filtering is applied. Notice that all the routes in the 172.16.0.0/16 and 172.20.0.0/16 range are present on both XR2 and R3.

### Example 12-16 XR2 and R3 Route Tables

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route eigrp
! Output omitted for brevity

D    10.34.1.0/24 [90/3072] via 10.23.1.3, 00:24:26, GigabitEthernet0/0/0/2
D    172.16.1.0/24 [90/130816] via 10.12.1.1, 00:23:25, GigabitEthernet0/0/0/0
D    172.16.2.0/24 [90/130816] via 10.12.1.1, 00:23:25, GigabitEthernet0/0/0/0
D    172.16.3.0/24 [90/130816] via 10.12.1.1, 00:23:25, GigabitEthernet0/0/0/0
D    172.20.1.0/24 [90/131072] via 10.23.1.3, 00:13:41, GigabitEthernet0/0/0/2
D    172.20.2.0/24 [90/131072] via 10.23.1.3, 00:13:41, GigabitEthernet0/0/0/2
D    172.20.3.0/24 [90/131072] via 10.23.1.3, 00:13:41, GigabitEthernet0/0/0/2
```

```
R3#show ip route eigrp
! Output omitted for brevity

D    10.12.1.0/24 [90/3072] via 10.23.1.2, 00:26:41, GigabitEthernet0/0
D    172.16.1.0 [90/131072] via 10.23.1.2, 00:23:49, GigabitEthernet0/0
D    172.16.2.0 [90/131072] via 10.23.1.2, 00:23:49, GigabitEthernet0/0
D    172.16.3.0 [90/131072] via 10.23.1.2, 00:23:49, GigabitEthernet0/0
D    172.20.1.0 [90/130816] via 10.34.1.4, 00:14:06, GigabitEthernet0/1
D    172.20.2.0 [90/130816] via 10.34.1.4, 00:14:06, GigabitEthernet0/1
D    172.20.3.0 [90/130816] via 10.34.1.4, 00:14:06, GigabitEthernet0/1
```

Example 12-17 provides the configuration of XR2 to demonstrate inbound filtering of the 172.16.1.0/24 prefix on XR2, and the outbound filtering of the 172.16.2.0/24 prefix. R3 uses an ACL to filter the 172.20.1.0/24 prefix on receipt and a prefix list to block the 172.20.2.0/24 upon advertisement.

### Example 12-17 EIGRP Route Filtering Configuration

[Click here to view code image](#)

**XR2**

```
router eigrp 100
address-family ipv4
  route-policy IN-EIGRP-FILTERING in
  route-policy OUT-EIGRP-FILTERING out
interface GigabitEthernet0/0/0/0
!
interface GigabitEthernet0/0/0/2
!
!
!
route-policy IN-EIGRP-FILTERING
  if not destination in (172.16.1.0/24) then
    pass
  endif
end-policy
!
route-policy OUT-EIGRP-FILTERING
  if not destination in (172.16.2.0/24) then
    pass
  endif
end-policy
```

**R3**

```
router eigrp 100
  distribute-list ACL-EIGRP-ONE in
  distribute-list prefix PREFIX-EIGRP-TWO out
network 10.23.0.0 0.0.255.255
network 10.34.0.0 0.0.255.255
!
ip access-list standard ACL-EIGRP-ONE
deny 172.20.1.0
permit any
!
ip prefix-list PREFIX-EIGRP-TWO seq 10 deny 172.20.2.0/24
ip prefix-list PREFIX-EIGRP-TWO seq 20 permit 0.0.0.0/0 le 32
```

**Example 12-18** displays the routing table on XR2 and R3 after enabling EIGRP filtering on the routers. XR2 still has the 172.16.2.0/24 prefix installed in the RIB, but the route was not advertised to R3. R3 has the 172.20.0/24 prefix installed in the RIB but did not advertise the prefix to XR2. The 172.16.1.0/24 and 172.20.1.0/24 network did not install into either router because they were blocked by the inbound filtering policy.

**Example 12-18 EIGRP Route Filtering Verification**

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route eigrp
```

```
! Output omitted for brevity
```

```
D 10.34.1.0/24 [90/3072] via 10.23.1.3, 00:13:03, GigabitEthernet0/0/0/2
D 172.16.2.0/24 [90/130816] via 10.12.1.1, 00:46:35, GigabitEthernet0/0/0/0
D 172.16.3.0/24 [90/130816] via 10.12.1.1, 00:46:35, GigabitEthernet0/0/0/0
D 172.20.3.0/24 [90/131072] via 10.23.1.3, 00:05:11, GigabitEthernet0/0/0/2
```

```
R3#show ip route eigrp
```

```
! Output omitted for brevity
```

```
D 10.12.1.0/24 [90/3072] via 10.23.1.2, 00:07:38, Ethernet0/0
D 172.16.3.0 [90/131072] via 10.23.1.2, 00:07:38, Ethernet0/0
D 172.20.2.0 [90/130816] via 10.34.1.4, 00:02:24, Ethernet0/1
D 172.20.3.0 [90/130816] via 10.34.1.4, 00:02:24, Ethernet0/1
```

### EIGRP Filtering by Hop Count

EIGRP is a hybrid distance vector routing protocol and does keep track of hop counts. In addition to filtering by prefixes, EIGRP supports filtering by hop counts. By default, an EIGRP router will allow only routes up to 100 hops away to be installed into the EIGRP topology table. Routes with the EIGRP hop count path attribute higher than 100 will not install into the EIGRP topology table.

The hop count can be changed on IOS and IOS XR routers with the EIGRP configuration command **metric maximum-hops** *hop-count*.


### EIGRP Offset Lists

Modifying the EIGRP path metric provides traffic engineering in EIGRP. Modifying the delay setting for an interface modifies all routes that are received and advertised from that router's interface. *Offset lists* allow for the modification of route attributes based on direction of the update, specific prefix, or combination of direction and prefix.

IOS routers use the EIGRP configuration command **offset-list** *off-set-value* {*acl-number* | *acl-name*} {**in** | **out**} [*interface-type interface-number*] to modify the metric value of a route. Specifying an interface restricts the conditional match for the offset list to the interface that the route is received or advertised out of.

On the downstream neighbor, the path metric increases by the offset value specified in the offset list. The offset value is calculated from an additional delay value that was added to the existing delay in the EIGRP path attribute. [Figure 12-6](#) shows the modified path metric formula when an offset delay is included.

$$\text{Metric} + \text{offset} = 256 * \left( \left( \frac{10^7}{\text{Min. Bandwidth}} + \frac{\text{Total Delay}}{10} \right) + \text{Offset Delay} \right)$$



$$\text{Offset} = 256 * \text{Offset Delay}$$

**Figure 12-6** EIGRP Offset Value Calculation

IOS XR routers use the EIGRP address family configuration command **route-policy** *route-*

*policy-name* {**in** | **out**} with the action command **add eigrp-metric bandwidth delay reliability load mtu** inside the RPL. Delay is measured in tens-of-microseconds ( $\mu$ s), and your calculations should be divided by ten to ensure the correct value is applied. Only the delay value should use a nonzero value.

Figure 12-7 provides an EIGRP topology to demonstrate EIGRP offset lists. XR1 is advertising the 172.16.1.0/24, 172.16.2.0/24, and 172.16.3.0/24 networks into EIGRP.

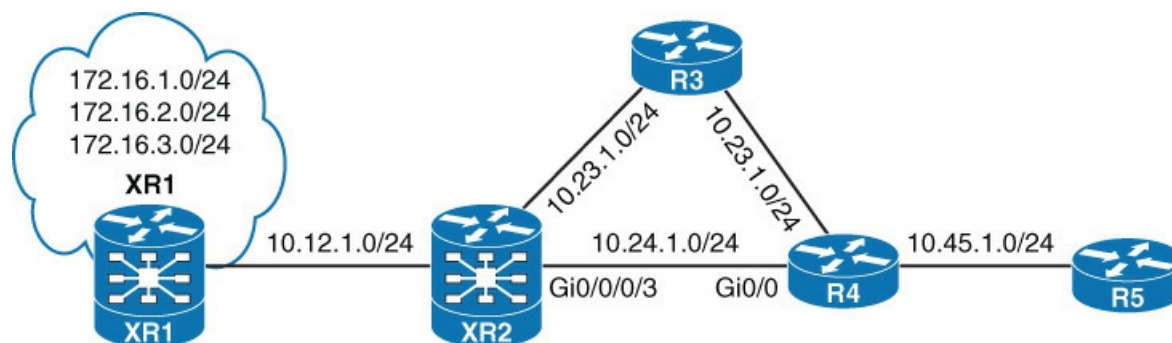


Figure 12-7 EIGRP Offset List Topology

Example 12-19 displays R4’s EIGRP routing table. Notice that the path metric for 172.16.1.0/24, 172.16.2.0/24, and 172.16.3.0/24 is the same value (131072).

### Example 12-19 R4 Routing Table

[Click here to view code image](#)

```
R4#show ip route eigrp
! Output omitted for brevity
D      10.12.1.0/24 [90/3072] via 10.24.1.2, 00:21:43, Ethernet0/0
D      10.23.1.0/24 [90/3072] via 10.34.1.3, 00:21:43, Ethernet0/1
D      172.16.1.0 [90/131072] via 10.24.1.2, 00:00:07, Ethernet0/0
D      172.16.2.0 [90/131072] via 10.24.1.2, 00:00:35, Ethernet0/0
D      172.16.3.0 [90/131072] via 10.24.1.2, 00:00:35, Ethernet0/0
```

Example 12-20 provides the configuration of XR2 and R4 so that traffic from R4 and R5 toward the 172.16.1.0/24 and 172.16.2.0/24 networks do not use the 10.24.1.0/24 network link. XR2 adds a delay of 100 microseconds ( $\mu$ s) to the 172.16.1.0/24 prefix upon advertisement of the Gi0/0/0/3 interface, and R4 adds an offset value of 2,560 to the 172.16.1.0/24 prefix upon receipt via the Gi0/0 interface. The 172.16.3.0/24 is not modified by XR2 or by R4.

### Example 12-20 EIGRP Offset List Configuration

[Click here to view code image](#)

**XR2**

```
router eigrp 100
  address-family ipv4
    interface GigabitEthernet0/0/0/0
    !
    interface GigabitEthernet0/0/0/2
    !
    interface GigabitEthernet0/0/0/3
      route-policy EIGRP-OUT out
    !
  !
route-policy EIGRP-OUT
  if destination in (172.16.1.0/24) then
    add eigrp-metric 0 10 0 0 0
  else
    pass
  endif
end-policy
```

**R4**

```
router eigrp 100
  network 10.24.0.0 0.0.255.255
  network 10.34.0.0 0.0.255.255
  network 10.45.0.0 0.0.255.255
  offset-list ACL-ONE in 2560 GigabitEthernet0/0
  !
ip access-list standard ACL-ONE
  permit 172.16.2.0
```

**Example 12-21** provides the routing tables of R4. Traffic from R4 and R5 toward the 172.16.1.0/24 and 172.16.2.0/24 will route through R4; all other network traffic is unaffected by this change and will use the 10.24.1.0/24 network link. Notice that the path metrics for 172.16.1.0/24 and 172.16.2.0/24 are exactly the same, even though IOS and IOS XR uses different configuration command syntax.

**Example 12-21 EIGRP Offset List Verification**

[Click here to view code image](#)

```
R4#show ip route eigrp
! Output omitted for brevity

D      10.12.1.0/24 [90/3072] via 10.24.1.2, 00:18:13, GigabitEthernet0/0
D      10.23.1.0/24 [90/3072] via 10.34.1.3, 00:18:13, GigabitEthernet0/1
                               [90/3072] via 10.24.1.2, 00:18:13, GigabitEthernet0/0
D      172.16.1.0 [90/131328] via 10.34.1.3, 00:18:02, GigabitEthernet0/1
D      172.16.2.0 [90/131328] via 10.34.1.3, 00:18:02, GigabitEthernet0/1
D      172.16.3.0 [90/131072] via 10.24.1.2, 00:18:02, GigabitEthernet0/0
```

**Example 12-22** displays the topology table for the 172.16.1.0/24 and 172.16.2.0/24 networks. Even though the path metric was modified by XR2 and R4 using different commands, the path

metric is the same because offset lists work by modifying only the delay path metric.

### Example 12-22 EIGRP Offset List Delay Inspection

[Click here to view code image](#)

```
R4#show ip eigrp topology 172.16.1.0/24
! Output omitted for brevity
EIGRP-IPv4 Topology Entry for AS(100)/ID(10.45.1.4) for 172.16.1.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 131328
  10.34.1.3 (Ethernet0/1), from 10.34.1.3, Send flag is 0x0
    Composite metric is (131328/131072), route is Internal
    Minimum bandwidth is 1000000 Kbit
    Total delay is 5030 microseconds
    Hop count is 3
  10.24.1.2 (Ethernet0/0), from 10.24.1.2, Send flag is 0x0
    Composite metric is (133632/133376), route is Internal
    Minimum bandwidth is 1000000 Kbit
    Total delay is 5120 microseconds
    Hop count is 2

R4#show ip eigrp topology 172.16.2.0/24
! Output omitted for brevity
EIGRP-IPv4 Topology Entry for AS(100)/ID(10.45.1.4) for 172.16.2.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 131328
  10.34.1.3 (Ethernet0/1), from 10.34.1.3, Send flag is 0x0
    Composite metric is (131328/131072), route is Internal
    Minimum bandwidth is 1000000 Kbit
    Total delay is 5030 microseconds
    Hop count is 3
  10.24.1.2 (Ethernet0/0), from 10.24.1.2, Send flag is 0x0
    Composite metric is (133632/133376), route is Internal
    Minimum bandwidth is 1000000 Kbit
    Total delay is 5120 microseconds
    Hop count is 2
```

### OSPF Filtering (Local)

OSPF is a link-state protocol that requires all routers in the same area to maintain an identical copy of the link-state database (LSDB). Unlike EIGRP, OSPF distribute lists only prevent the route from installing in the local RIB. It does not prevent the installation of the LSA into the LSDB or the propagation of the LSA throughout the area. [Figure 12-8](#) demonstrates the distribute list logic used by OSPF.

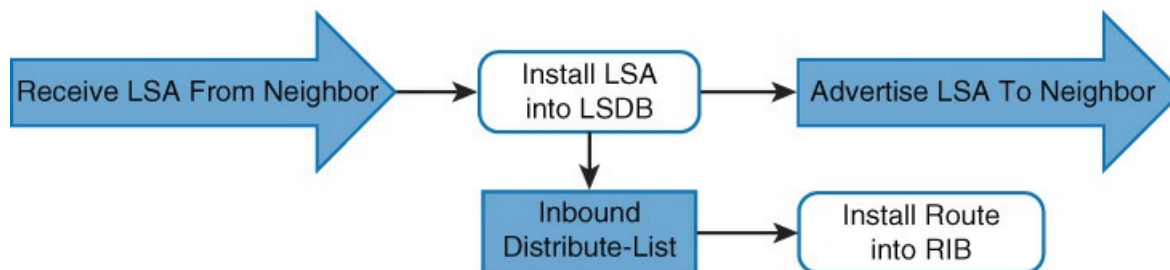


Figure 12-8 OSPF Distribute List Filtering Logic

IOS routers use the OSPF configuration command **distribute-list** {acl-number | acl-name | prefix prefix-list-name | route-map route-map-name} in for configuring a distribute list, and IOS XR routers use the OSPF configuration command **distribute-list** {acl-number | route-

**policy route-policy-name} in.**

Figure 12-9 illustrates an OSPF topology with routes in the 172.16.0.0/16 range advertised in Area 1, and routes in the 172.20.0.0/16 range advertised into Area 2. XR2 and R3 will filter routes from the RIB to demonstrate OSPF distribute lists.

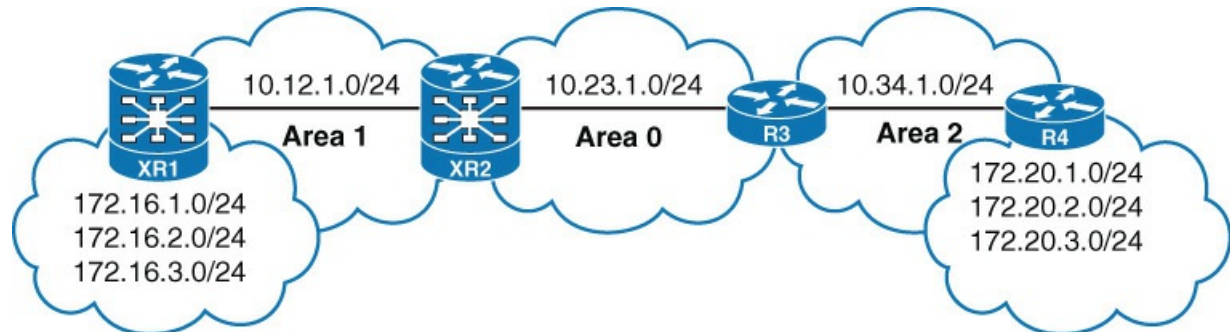


Figure 12-9 OSPF Local Distribute List Filtering

Example 12-23 provides the configuration of XR2 and R3. XR2 filters the 172.16.1.0/24 and 172.20.3.0/24 networks from the RIB, and R3 filters the 172.16.3.0/24 and 172.20.1.0/24 network from the RIB.

### Example 12-23 OSPF Distribute List Configuration

[Click here to view code image](#)

#### XR2

```
router ospf 1
  distribute-list route-policy OSPF-IN in
  area 0
    interface GigabitEthernet0/0/0/2
    !
  area 1
    interface GigabitEthernet0/0/0/0
    !

route-policy OSPF-IN
  if not (destination in (172.16.1.0/24, 172.20.3.0/24)) then
    pass
  endif
end-policy
```

#### R3

```
router ospf 1
  router-id 192.168.3.3
  distribute-list ACL-OSPF-ONE in
  !
ip access-list standard ACL-OSPF-ONE
  deny 172.20.1.0
  deny 172.16.3.0
  permit any
```



**Example 12-24** displays the routing table for XR2 and R3. The 172.16.1.0/24 network is removed from XR2's RIB but is present on R3. The 172.20.1.0/24 network is removed from R3's RIB but is present on XR2.

### Example 12-24 OSPF Distribute List Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route ospf
```

```
! Output omitted for brevity
```

```
O IA 10.34.1.0/24 [110/2] via 10.23.1.3, 00:05:15, GigabitEthernet0/0/0/2
O   172.16.2.0/24 [110/2] via 10.12.1.1, 00:05:15, GigabitEthernet0/0/0/0
O   172.16.3.0/24 [110/2] via 10.12.1.1, 00:05:15, GigabitEthernet0/0/0/0
O IA 172.20.1.0/24 [110/3] via 10.23.1.3, 00:05:15, GigabitEthernet0/0/0/2
O IA 172.20.2.0/24 [110/3] via 10.23.1.3, 00:05:15, GigabitEthernet0/0/0/2
```

```
R3#show ip route ospf
```

```
! Output omitted for brevity
```

```
10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
O IA 10.12.1.0/24 [110/2] via 10.23.1.2, 00:02:36, GigabitEthernet0/0
O IA 172.16.1.0 [110/3] via 10.23.1.2, 00:02:36, GigabitEthernet0/0
O IA 172.16.2.0 [110/3] via 10.23.1.2, 00:02:36, GigabitEthernet0/0
O   172.20.2.0 [110/2] via 10.34.1.4, 00:02:36, GigabitEthernet0/1
O   172.20.3.0 [110/2] via 10.34.1.4, 00:02:36, GigabitEthernet0/1
```

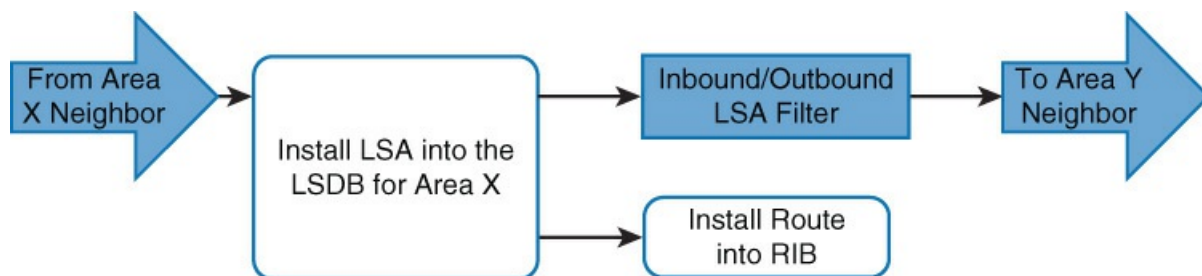
#### Note

A distribute list on an ABR does not prevent Type 1 LSAs from becoming Type 3 LSAs in a different area because the Type 3 LSA generation occurs before the distribute list is processed.

However, a distribute list on an ABR prevents Type 3 LSAs coming from the backbone from being regenerated into nonbackbone areas because this regeneration process happens after the distribute list is processed. A distribute list should not be used for filtering of prefixes between areas, as the following section identifies more preferred techniques.

### OSPF Filtering (Area)

Some network designs require that OSPF routes should not be advertised by the ABRs into other areas. Summarizing with the **no-advertise** option (explained in Chapter 7, "Advanced OSPF") is very efficient, but some designs require the selection based on other criteria. Filtering Type 3 LSA generation still allows the routes to install in the appropriate area of the OSPF database, and does not generate a Type 3 LSA for the area being filtered. [Figure 12-10](#) demonstrates the concept.



**Figure 12-10** OSPF Area Filtering Concepts

[Figure 12-11](#) demonstrates that the ABR can filter routes as they advertise out of an area into

other areas, or into an area. XR2 is able to filter routes (link-state advertisements [LSAs]) as they leave Area 1 or enter Area 0, and that R3 can filter routes as they leave Area 0 or enter Area 2. The same logic applies with routes advertised in the opposition direction.

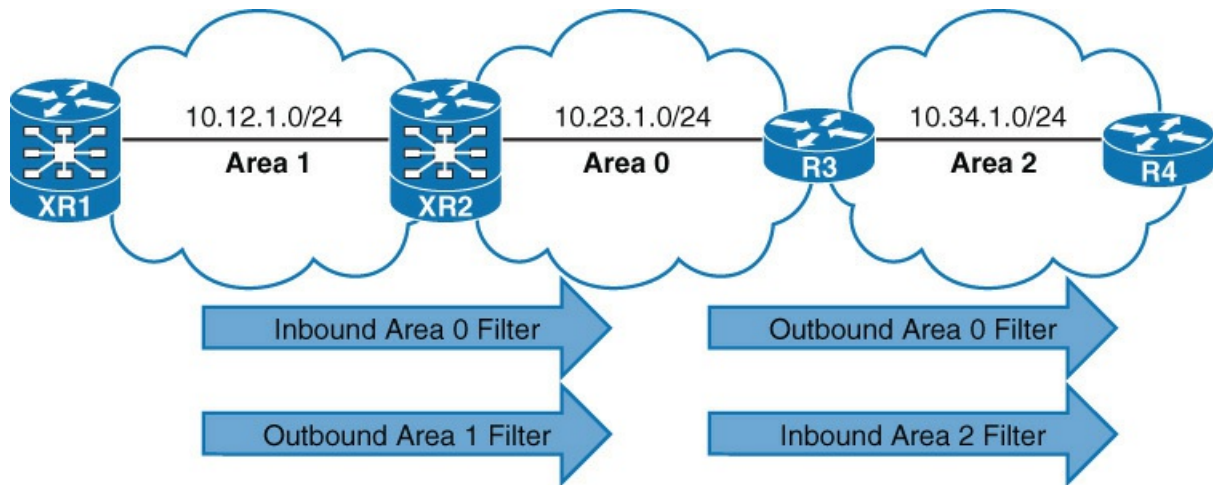


Figure 12-11 OSPF Area Filtering Logic

IOS routers filter prefixes crossing the ABR with the command **area area-id filter-list prefix prefix-list-name {in | out}**, and IOS XR routers use the OSPF area configuration command **route-policy route-policy-name {in | out}** under the appropriate area.

Figure 12-12 provides a reference topology where XR1 is advertising the 172.16.1.0/24 and 172.16.2.0/24 network prefixes, and R4 is advertising the 172.20.1.0/24 and 172.20.2.0/24 network prefixes into OSPF.

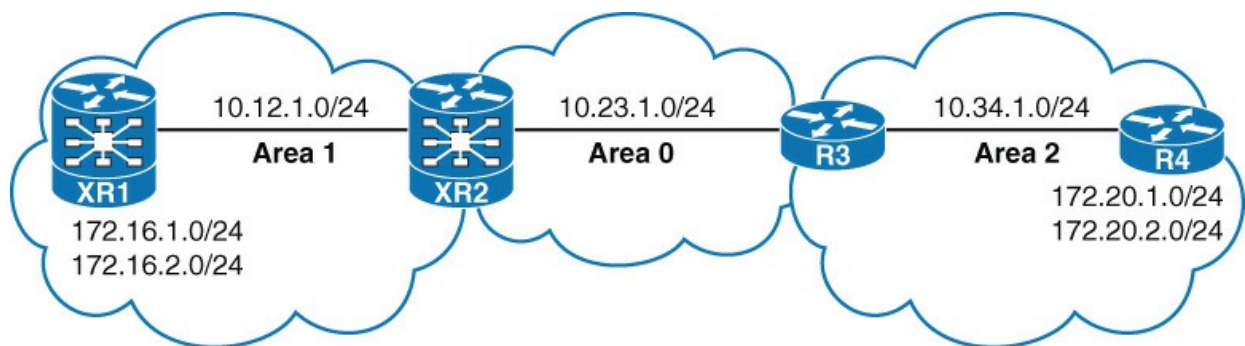


Figure 12-12 Topology for Demonstrating OSPF Area Filtering

Example 12-25 provides the OSPF configuration for XR2 and R3. XR2 is filtering the 172.20.1.0/24 network upon entry into Area 1, and R3 is filtering the 172.16.1.0/24 network as it leaves Area 0.

### Example 12-25 OSPF Area Filtering Configuration

[Click here to view code image](#)

**XR2**

```
router ospf 1
  area 0
    interface GigabitEthernet0/0/0/2
    !
  !
  area 1
    route-policy OSPF-AREA-FILTER in
    interface GigabitEthernet0/0/0/0
  !
route-policy OSPF-AREA-FILTER
  if not destination in (172.20.1.0/24) then
    pass
  endif
end-policy
```

**R3**

```
router ospf 1
  router-id 192.168.3.3
  network 10.23.1.0 0.0.0.255 area 0
  network 10.34.1.0 0.0.0.255 area 2
  area 0 filter-list prefix PREFIX-FILTER out
  !
ip prefix-list PREFIX-FILTER seq 5 deny 172.16.1.0/24
ip prefix-list PREFIX-FILTER seq 10 permit 0.0.0.0/0 le 32
```

**Example 12-26** displays the routing table on XR1 where the 172.20.1.0/24 network is no longer present. R4's routing table does not contain the 172.16.1.0/24 network either, which verifies that the area filtering was successful for both routes.

**Example 12-26 OSPF Area Filtering Verification**

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route ospf
```

```
! Output omitted for brevity
```

```
O IA 10.23.1.0/24 [110/2] via 10.12.1.2, 01:31:53, GigabitEthernet0/0/0/0
O IA 10.34.1.0/24 [110/3] via 10.12.1.2, 01:31:53, GigabitEthernet0/0/0/0
O IA 172.20.2.0/24 [110/4] via 10.12.1.2, 00:07:33, GigabitEthernet0/0/0/0
```

```
R4#show ip route ospf
```

```
! Output omitted for brevity
```

```
O IA 10.12.1.0/24 [110/3] via 10.34.1.3, 00:00:07, Ethernet0/1
O IA 10.23.1.0/24 [110/2] via 10.34.1.3, 00:00:07, Ethernet0/1
C 10.34.1.0/24 is directly connected, Ethernet0/1
O IA 172.16.2.0 [110/4] via 10.34.1.3, 00:00:07, Ethernet0/1
C 172.20.1.0/24 is directly connected, Loopback0
C 172.20.2.0/24 is directly connected, Loopback1
```

## IS-IS Filtering (Local)

Cisco IOS 15.3.(3)M introduced the concept of filtering specific IS-IS prefixes from the RIB, as demonstrated in [Figure 12-13](#). Because IS-IS is a link-state protocol, and all routers must maintain an identical level link-state protocol database (LSPDB), routes are filtered only from installation into the local RIB. The concept is identical to OSPF's local filtering, which was explained earlier.

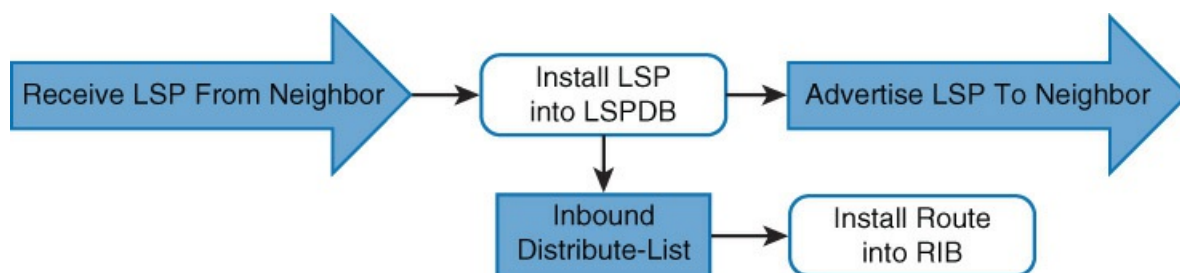


Figure 12-13 IS-IS Distribute List Filtering Logic

IS-IS inbound filtering is done with the command **distribute-list** {*acl-number* | **prefix prefix-list-name** | **route-map route-map-name**} **in**.

### Note

IOS XR does not support this feature at the time of this writing.

## BGP Filtering

Chapter 11, “Route Maps and Route Policy,” covers advanced topics in BGP filtering and manipulation such as route policies, route maps, prefix lists, and autonomous system path (AS\_Path) ACLs. The IOS implementation of BGP allows for filtering of BGP prefixes with a distribute-list to maintain consistency with the other interior gateway protocols (IGPs).

Distribute lists allow the filtering of network prefixes on a neighbor-by-neighbor basis using standard or extended ACLs. IOS XR does not support BGP distribute lists, and filtering must be done through a route policy. IOS routers configure a BGP distribute list with the BGP address family configuration command **neighbor ip-address distribute-list** {*acl-number* | *acl-name*} {**in**|**out**}. Remember that extended ACLs for BGP use the source fields to match the network portion, and the destination fields to match against the network mask.

### Note

A BGP neighbor cannot use a distribute list and prefix list at the same time for receiving or advertising routes.

[Figure 12-14](#) shows a BGP topology example to demonstrate BGP distribute lists on R3. In addition to all routers advertising their peering links, XR1 is advertising the 192.168.1.1/32 network, and R4 is advertising the 192.168.4.4/32 network.

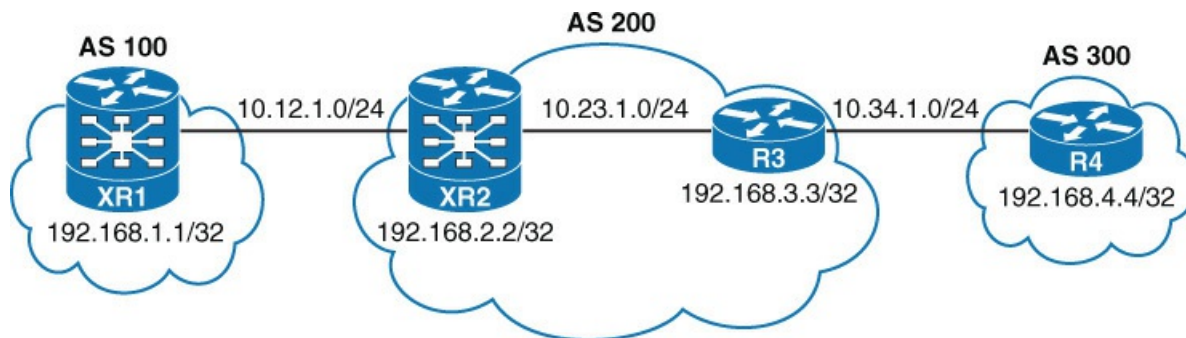


Figure 12-14 BGP Distribute List Topology

Example 12-27 provide R3's BGP table before any filtering is done.

### Example 12-27 Reference BGP Table

[Click here to view code image](#)

```
R3#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i10.12.1.0/24	10.23.1.2	0	100	0	i
* i10.23.1.0/24	10.23.1.2	0	100	0	i
*>	0.0.0.0	0		32768	i
*> 10.34.1.0/24	0.0.0.0	0		32768	i
*>i192.168.1.1/32	10.12.1.1	0	100	0	100 i
*>i192.168.2.2/32	10.23.1.2	0	100	0	i
*> 192.168.3.3/32	0.0.0.0	0		32768	i
*> 192.168.4.4/32	10.34.1.4	0		0	300 i

Example 12-28 displays R3's BGP configuration that demonstrates filtering with distribute lists. The configuration uses an extended ACL EXT-BGP-LOOPBACK that only allows receipt of networks from R4 and XR2 that are in the 192.168.0.0/16 range with a /32 prefix length.

### Example 12-28 BGP Distribute List Configuration

[Click here to view code image](#)

```
R3
router bgp 200
!
address-family ipv4
network 10.23.1.0 mask 255.255.255.0
network 10.34.1.0 mask 255.255.255.0
network 192.168.3.3 mask 255.255.255.255
neighbor 10.23.1.2 distribute-list EXT-BGP-LOOPBACK in
neighbor 10.34.1.4 distribute-list EXT-BGP-LOOPBACK in
exit-address-family
!
ip access-list extended EXT-BGP-LOOPBACK
permit ip 192.168.0.0 0.0.255.255 host 255.255.255.255
```

Example 12-29 verifies that only the networks in the 192.168.0.0/16 range are accepted by XR2

and R4.

### Example 12-29 OSPF Route Filtering Configuration

[Click here to view code image](#)

```
R3#show bgp ipv4 unicast
! Output omitted for brevity

      Network          Next Hop          Metric LocPrf Weight Path
*> 10.23.1.0/24        0.0.0.0           0         32768 i
*> 10.34.1.0/24        0.0.0.0           0         32768 i
*> 192.168.1.1/32     10.12.1.1         0         100      0 100 i
*> 192.168.2.2/32     10.23.1.2         0         100      0 i
*> 192.168.3.3/32     0.0.0.0           0         32768 i
*> 192.168.4.4/32     10.34.1.4         0         0        300 i
```

IOS routers support multiple techniques for filtering routes in BGP, and each technique is executed in order depending on whether the filtering technique is configured for inbound versus outbound processing. [Table 12-2](#) provides the processing order for inbound and outbound configurations.

Order of Processing	Inbound	Outbound
First	Route map	Prefix/distribute list
Second	Filter list	Filter list
Third	Prefix/ distribute list	Route map

Table 12-2 IOS BGP Order of Processing for Filter Techniques

IOS XR simplifies the process by only allowing route policies for filtering prefixes for BGP peers.

#### Clearing BGP Connections

Depending on the change to the BGP route manipulation technique, the BGP session may need to be refreshed to take effect. BGP supports two methods of clearing a BGP session, the first method is a *hard reset*, which tears down the BGP session, removes BGP routes from the peer, and is the most disruptive. The second method is a *soft reset*, which invalidates the BGP cache and requests a full advertisement from its BGP peer.

IOS nodes initiate a hard reset with the command **clear ip bgp ip-address [soft]**, and the command **clear bgp ip-address [graceful]** is used on IOS XR nodes. Soft reset on an IOS nodes use the optional **soft** keyword, and IOS XR nodes use the optional **graceful** keyword. Sessions can be cleared with all BGP neighbors by using an asterisk (\*) in lieu of the peer's IP address.

When a BGP policy changes, the BGP table must be processed again so that the neighbors can be notified accordingly. Routes received by a BGP peer must be processed again. If the BGP session supports *route refresh* capability, the peer re-advertises (refreshes) the prefixes to the requesting router, allowing for the inbound policy to process using the new policy changes. The route refresh capability is negotiated for each address family when the session is established.

Performing a soft reset on sessions that support route refresh capability actually initiates a route

refresh. Soft resets can be performed for a specific address family with the command **clear bgp address-family address-family-modifier ip-address soft [in | out]**. Soft resets reduce the amount of routes that must be exchanged if multiple address families are configured with a single BGP peer. Changes to the outbound routing policies use the optional **out** keyword, and changes to inbound routing policies use the optional **in** keyword.

Older IOS versions that do not support route refresh capability require the usage of inbound soft reconfiguration so that updates to inbound route policies can be applied without performing a hard reset. Inbound soft reconfiguration does not purge the Adj-RIB-In table after routes process into the Loc-RIB table. The Adj-RIB-In maintains only the raw unedited routes that were received from the neighbors and thereby allows the inbound route policies to be processed again. Enabling this feature can consume a significant amount of memory because the Adj-RIB-In table stays in memory. Inbound soft reconfiguration uses the address family command **neighbor ip-address soft-reconfiguration inbound** for IOS nodes, and IOS XR nodes use the neighbor-specific address family command **soft-reconfiguration inbound**. Figure 12-15 demonstrates the three BGP tables and their interaction with route processing.

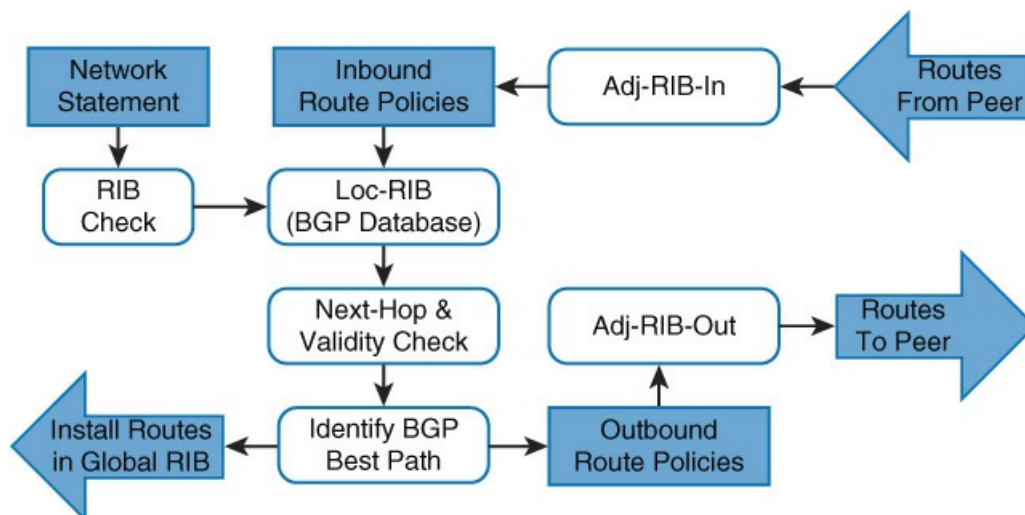


Figure 12-15 BGP Table and Route Policy Processing

## SUMMARY

This chapter demonstrated that the default routing protocol behavior for a router can be manipulated using AD and filtering. Packet forwarding decisions can bypass routing protocols altogether through the use of conditional routing policies.

- Policy-based routing (PBR), local PBR, and ACL-based forwarding (ABF) allow for conditional routing of packets based on other characteristics of the data packet besides the destination address.
- A router uses AD to determine which route to install into the RIB when there are multiple paths to the same destination from multiple routing protocols.
- Filtering routes from distance vector and path vector routing protocols can occur as routes are received or advertised. Filtering upon receipt prevents the installation into the RIB; filtering during advertisement allows the routes to install into the RIB but not advertise to downstream neighbors.

- Link-state routing protocols require that the LSDB remain consistent for all routers in the area so that the SPF algorithm consistently executes on all routers in the area.
- Filtering of routes is only possible with
- Summarization with the **no-advertise** option, which prevents the routes from crossing areas or from being injected into the routing domain on ASBRs
- Removal of routes from the RIB on a local router with distribute lists
- Area filtering on OSPF ABRs with the area filter

## REFERENCES IN THIS CHAPTER

Cisco. *Cisco IOS Software Configuration Guides*. <http://www.cisco.com>

Cisco. *Cisco IOS XR Software Configuration Guides*. <http://www.cisco.com>



## Chapter 13. Route Redistribution

This chapter covers the following topics:

- Redistribution
- Protocol-specific configuration
- Challenges with redistribution
- Avoiding routing loops

An organization might use multiple routing protocols, split up the routing domain between multiple instances (processes) of the same routing protocol, or need to merge networks with another organization that uses a different routing protocol. In all these scenarios, the routes from one routing protocol process need to be exchanged with a different routing protocol process to provide full connectivity. Redistribution is the method of injecting routes from one routing protocol into another routing protocol.

Figure 13-1 illustrates a network that has multiple routing protocols that are not working together. R1, R2, and R3 exchange routes using Enhanced Interior Gateway Routing Protocol (EIGRP), and R3, R4, and R5 exchange routes with Open Shortest Path First (OSPF) Protocol. R1 and R5 advertise their Loopback 0 interfaces (192.168.1.1/32 and 192.168.5.5/32) into their appropriate routing protocol, but they cannot establish connectivity to each other. Only R3 can connect to R1 and R5 because it is the only router that participates with both routing protocols and has a complete view of the network.

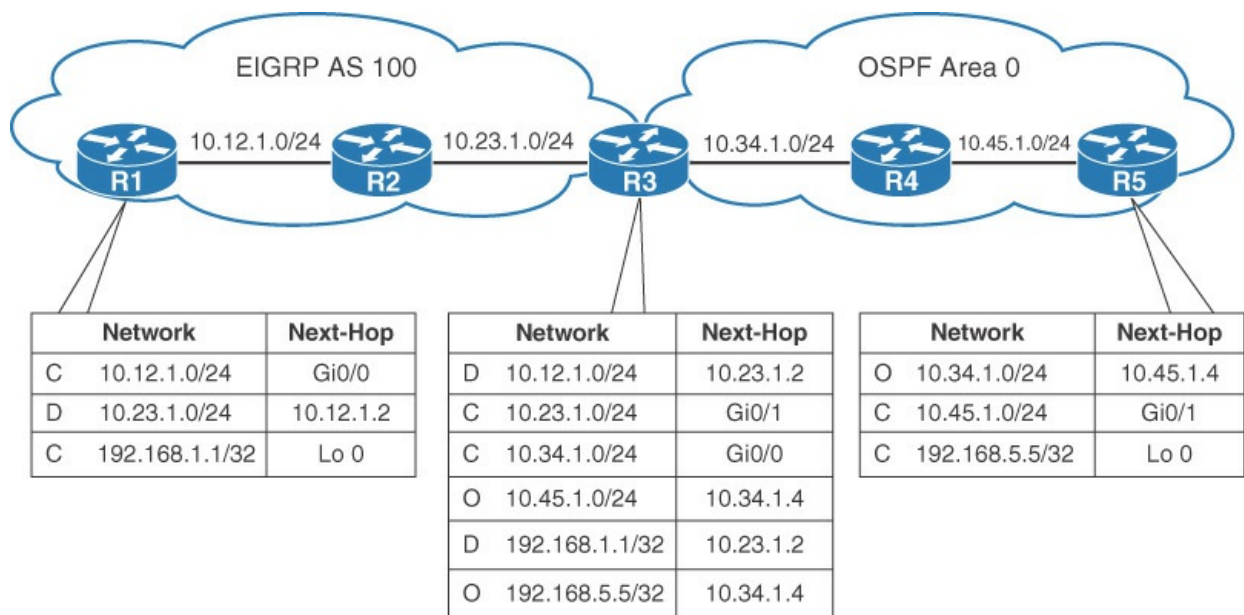
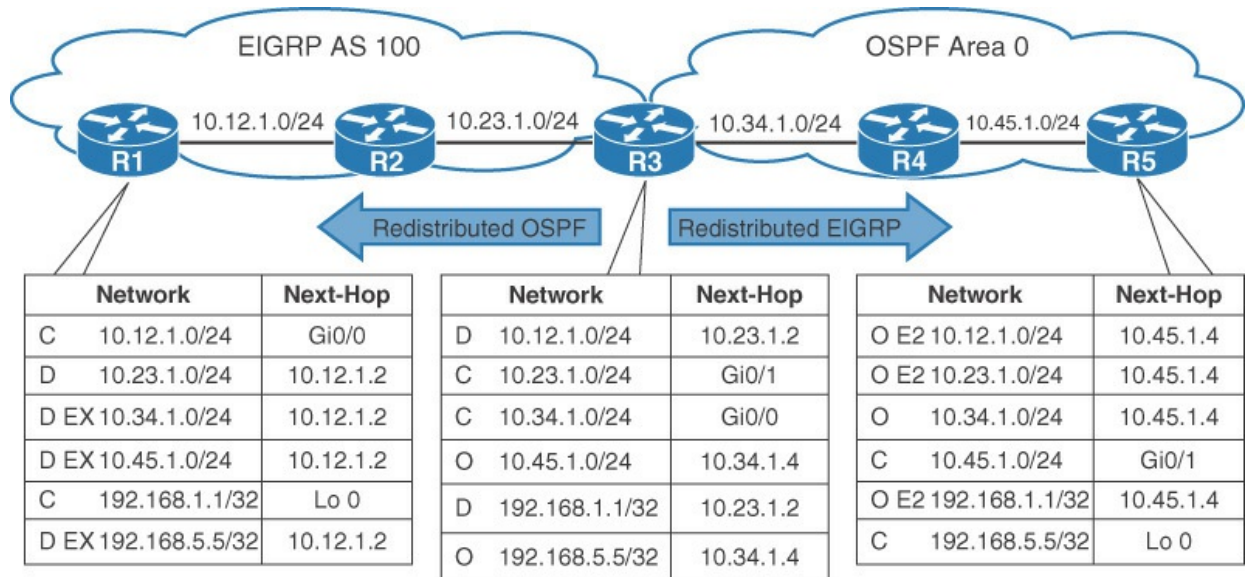


Figure 13-1 Topology with Multiple Routing Protocols

Even though R3 has all the routes to EIGRP and OSPF, the routes do not automatically redistribute between the routing protocols. Redistribution must be configured so that EIGRP

routes are injected into OSPF, and OSPF routes inject into EIGRP. Mutual redistribution is the process where both routing protocols redistribute into each other in both directions on the same router.

In [Figure 13-2](#), the R3 router is performing mutual redistribution. The OSPF routes are present in EIGRP as external routes, and the EIGRP routes are present in the OSPF routing domain as external routes (Type 5 link-state advertisements [LSAs]). R1 and R5 can establish connectivity between their loopbacks because the appropriate routes are present in the routing table.



**Figure 13-2** Mutual Redistribution Topology

## REDISTRIBUTION BASICS

Redistribution always encompasses two routing protocols: a source and destination. The source protocol provides the routes that are to be redistributed, and the destination protocol receives the injected routes. The redistribution configuration exists under the destination protocol and identifies the source protocol. Using a route map or route policy allows for the filtering or modification of route attributes during the injection into the destination protocol. [Table 13-1](#) provides a list of source protocols for redistribution.

Route Source	Description
Static	Any static route that is present in the Routing Information Base (RIB). * Static can only be a source.
Connected	Any interface in an up state that is not associated with the destination protocol. * Connected can only be a source.
EIGRP	Any routes in EIGRP, including EIGRP-enabled connected networks.
OSPF	Any routes in the OSPF link-state database (LSDB), including OSPF-enabled interfaces.
IS-IS	Any routes in the IS-IS LSPDB, except the local Intermediate System-to-Intermediate System (IS-IS) Protocol-enabled interfaces.
BGP	Any routes in the Border Gateway Protocol (BGP) Loc-RIB table learned externally. Internal BGP (iBGP) routes require the command <b>redistribute internal</b> for redistribution into interior gateway protocol (IGP) routing protocols.

**Table 13-1** Redistribution Source Protocol Chart

### Redistribution Is Not Transitive

When redistributing between two or more routing protocols on a single router, redistribution is not transitive. In other words, when a router redistributes protocol 1 into protocol 2, and protocol 2 redistributes into protocol 3, the routes from protocol 1 does not redistribute into protocol 3.

**Example 13-1** provides sample logic for EIGRP mutually redistributing into OSPF and OSPF mutually redistributing with BGP.

### **Example 13-1** Problematic Multiprotocol Redistribution Logic

[Click here to view code image](#)

```
router eigrp
  redistribute ospf
router ospf
  redistribute eigrp
  redistribute bgp
router bgp
  redistribute ospf
```

**Figure 13-3** illustrates redistribution on the router. The EIGRP route 172.16.1.0/24 redistributes into OSPF but does not redistribute into BGP. The BGP route 172.16.3.0/24 redistributes into OSPF but does not be redistribute into EIGRP. The prefix 172.16.2.0/24 will redistribute to both EIGRP and BGP.

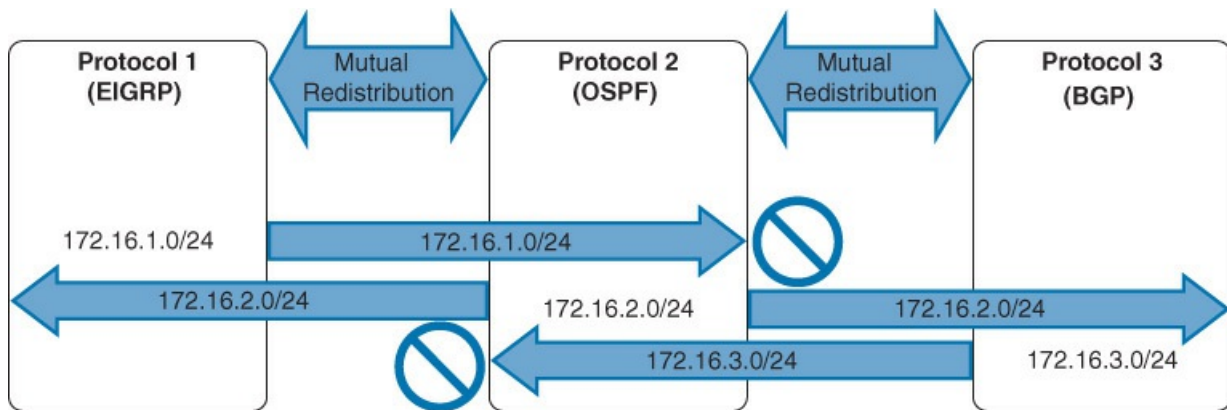


Figure 13-3 Nontransitive Redistribution Logic

For routes to exchange between all three routing protocols requires that mutual redistribution must be configured between all three protocols, as shown in [Example 13-2](#).

### Example 13-2 Multiprotocol Redistribution Logic

[Click here to view code image](#)

```
router eigrp
 redistribute ospf
 redistribute bgp
router ospf
 redistribute eigrp
 redistribute bgp
router bgp
 redistribute ospf
 redistribute eigrp
```

Now that all three routing protocols are mutually redistributed, EIGRP's 172.16.1.0/24 network exists in OSPF and BGP, OSPF's 172.16.2.0/24 network exists in EIGRP and BGP, and BGP's 172.16.3.0/24 network exists in OSPF and EIGRP. [Figure 13-4](#) illustrates the router processing for all three network prefixes.

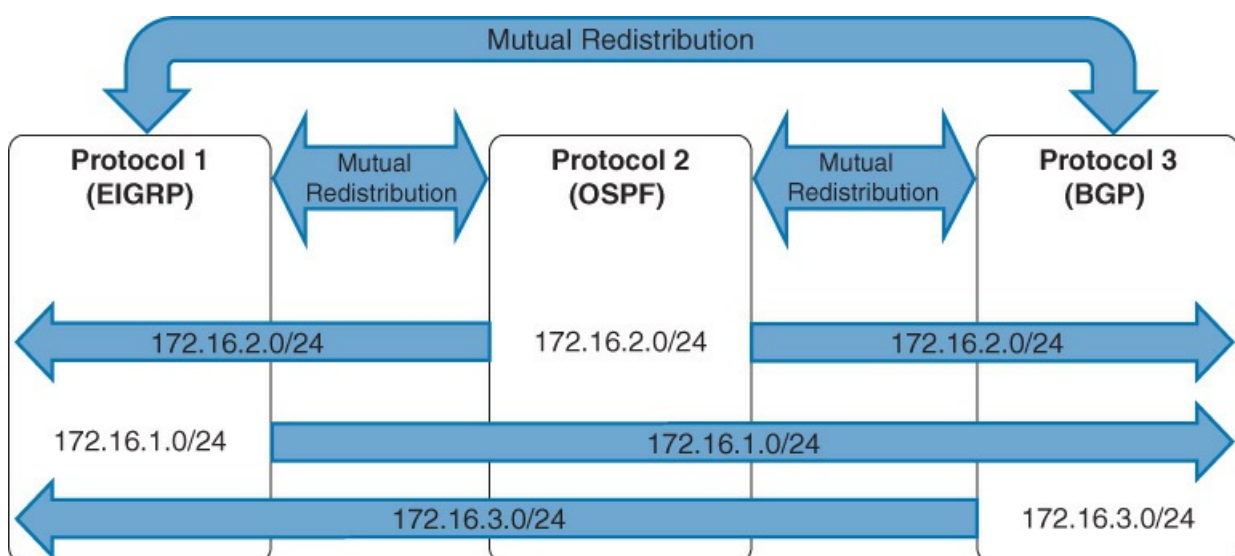


Figure 13-4 Multiprotocol Mutual Redistribution

## Sequential Protocol Redistribution

Sequential protocol redistribution is redistribution between multiple protocols over a series of routers, as shown in Figure 13-5. R2 redistributes the EIGRP 192.168.1.1/32 prefix into BGP, and R4 redistributes the BGP 192.168.1.1/32 prefix into OSPF. All three routing protocols contain the 192.168.1.1/32 prefix.

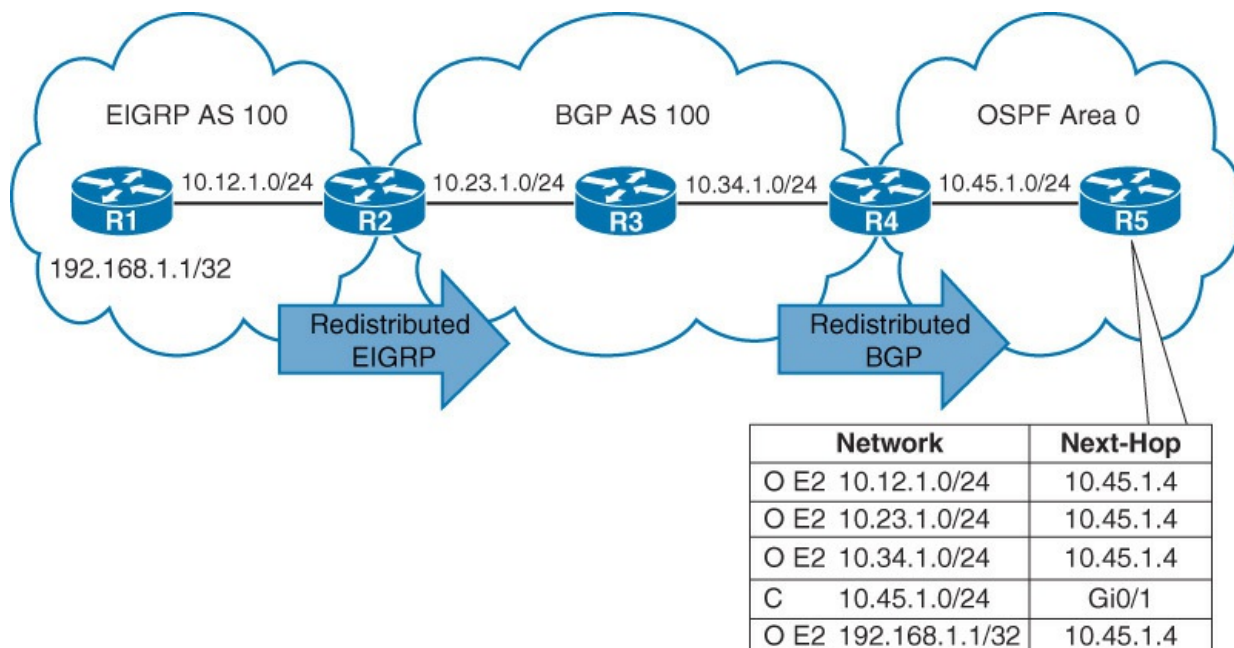


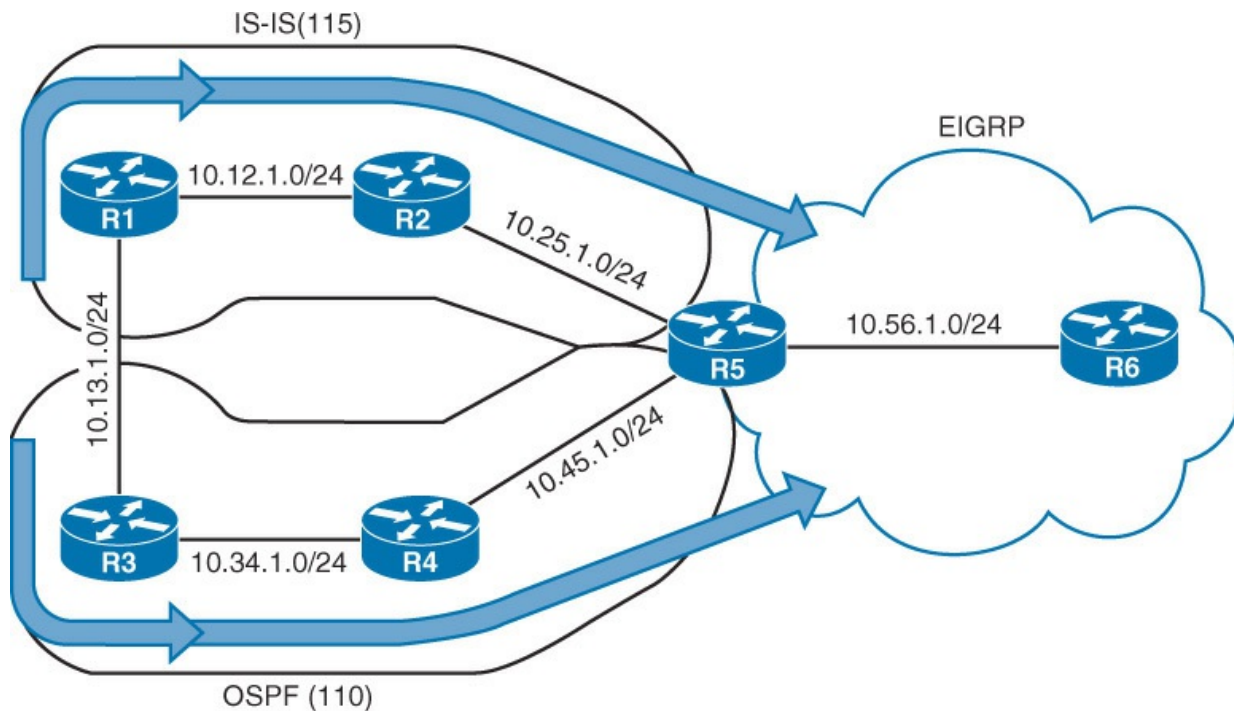
Figure 13-5 Sequential Protocol Redistribution on Different Routers

### Routes Must Exist in the RIB

A route must exist in the RIB for it to redistribute into the destination protocol. In essence, this provides a safety mechanism by ensuring that the route is deemed reachable by the redistributing router.

In addition to the route being in the RIB, the source protocol that redistributes into the destination protocol must be the source for the route in the RIB. This ensures that the router redistributes only the route it deemed as the best for the destination protocol. The only exception for that logic is for directly connected interfaces participating in the source protocol because they will have an administrative distance (AD) of 0.

In Figure 13-6, R1 and R3 both advertise the 10.13.1.0/24 network, and R5 is redistributing IS-IS and OSPF routes into EIGRP.



**Figure 13-6** Identification of Source Protocol Topology

R5 receives route information for 10.13.1.0/24 from both IS-IS and OSPF routing protocols. [Example 13-3](#) provides verification that R5 contains an entry for the 10.13.1.0/24 network in OSPF's LSDB and IS-IS's LSPDB.

### Example 13-3 Verification of Network in Link-State Databases

[Click here to view code image](#)

```
R5#show isis database detail R1.00-00 level-1
! Output omitted for brevity
IS-IS Level-1 LSP R1.00-00
  Metric: 10      IP 10.12.1.0 255.255.255.0
  Metric: 10      IP 10.13.1.0 255.255.255.0
```

```
R5#show ip ospf database router 192.168.3.3
! Output omitted for brevity
      OSPF Router with ID (192.168.5.5) (Process ID 1)
Link State ID: 192.168.3.3
Advertising Router: 192.168.3.3

Link connected to: a Stub Network
(Link ID) Network/subnet number: 10.13.1.0
(Link Data) Network Mask: 255.255.255.0
```

R5 determines that the most desirable path to reach 10.13.1.0/24 is OSPF because it has a lower AD than IS-IS. [Example 13-4](#) displays R5's routing table. The 10.13.1.0/24 OSPF route was inserted into the RIB.

### Example 13-4 R5's Routing Table



[Click here to view code image](#)

```
R5#show ip route
! Output omitted for brevity

i L1    10.12.1.0/24 [115/20] via 10.25.1.2, 22:46:01, GigabitEthernet0/1
O       10.13.1.0/24 [110/30] via 10.45.1.4, 00:04:27, GigabitEthernet0/0
C       10.25.1.0/24 is directly connected, GigabitEthernet0/1
O       10.34.1.0/24 [110/20] via 10.45.1.4, 22:48:24, GigabitEthernet0/0
C       10.45.1.0/24 is directly connected, GigabitEthernet0/0
C       10.56.1.0/24 is directly connected, GigabitEthernet0/2
```

**Example 13-5** displays the EIGRP topology table for the 10.13.1.0/24 network. During redistribution, R5 checks the Routing Information Base (RIB) for the 10.13.1.0/24 network and verifies its existence in the RIB and confirms that the source protocol is the protocol that installed the route. EIGRP identifies the source protocol as OSPF with a path metric of 30.

### **Example 13-5** EIGRP Topology Table for the 10.13.1.0/24 Network

[Click here to view code image](#)

```
R5#show ip eigrp topology 10.13.1.0/24
! Output omitted for brevity
EIGRP-IPv4 Topology Entry for AS(100)/ID(10.56.1.5) for 10.13.1.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 2560000256
  Descriptor Blocks:
  10.45.1.4, from Redistributed, Send flag is 0x0
    External data:
      AS number of route is 1
      External protocol is OSPF, external metric is 30
      Administrator tag is 0 (0x00000000)
```

#### Note

When redistributing from a source protocol of higher AD into a destination protocol with a lower AD, the route shown in the routing table will always be that of the source protocol.

## Metrics

Every routing protocol has a different methodology for calculating the best path for a route. For example, EIGRP can use bandwidth, delay, load, and reliability for calculating its best path, whereas OSPF primarily uses the path metric for calculating the shortest path first (SPF) tree (SPT). OSPF cannot calculate the SPF tree using EIGRP path attributes, and EIGRP cannot run Diffusing Update Algorithm (DUAL) using only the total path metric. The destination protocol must provide relevant metrics to the destination protocols so that the destination protocol can calculate the best path for the redistributed routes.

Every protocol provides a seed metric at the time of redistribution that allows the destination protocol to calculate a best path. The seed metric is a baseline and may reflect a loss of information during redistribution when redistribution occurs between two different protocol types (that is, EIGRP to OSPF). A route map or route policy can modify the seed metric for a route during redistribution. [Table 13-2](#) provides a list of seed metrics for the destination routing protocol.

Protocol	Default Seed Metric
EIGRP	Infinity. Routes set with infinity do not install into the EIGRP topology table.
OSPF	All routes are Type 2 external. Routes sourced from BGP use a seed metric of 1, and all other protocols uses a seed metric of 20.
IS-IS	0.
BGP	Origin is set to incomplete, the Multi-Exit Discriminator (MED) is set to the IGP metric, and the weight is set to 32,768.

Table 13-2 Redistribution Source Protocol Chart

## PROTOCOL-SPECIFIC CONFIGURATION

Every routing protocol has a unique redistribution behavior and is explained throughout this section. The following command syntaxes exist in the destination protocol to identify the source routing protocol:

■ IOS routers use the command **redistribute** {**connected** | **static** | **eigrp** *as-number* | **ospf** *process-id* [**match** {**internal** | **external** [**1**|**2**}] } | **is-is** [*instance-id*] [**level-1** | **level-1-2** | **level-2**] | **bgp** *as-number* } [*destination-protocol-options*].

■ IOS XR routers use the command **redistribute** {**connected** | **static** | **eigrp** *as-number* [**match** {**internal** | **external**}] } | **ospf** *process-id* [**match** {**internal** | **external** [**1**|**2**}] } | **is-is** [*instance-id*] [**level-1** | **level-1-2** | **level-2**] | **bgp** *as-number* } [*destination-protocol-options*].

Redistribution commonly uses route maps or route policies to manipulate or filter routes on the redistributing router. [Table 13-3](#) provides additional conditional matching commands for route selection during redistribution.



<b>Application</b>	<b>Match command</b>	<b>Description</b>
Route map	<b>match interface</b> <i>interface-type</i> <i>interface-number</i>	Selects prefixes based on the outbound interface for selection of routes
Route map	<b>match route-type</b> {external [ <i>type-1</i>   <i>type-2</i> ]   internal   level-1   level-2   local   nssa-external [ <i>type-1</i>   <i>type-2</i> ]}	Selects prefixes based on routing protocol characteristics: <b>external:</b> External BGP, EIGRP, or OSPF <b>internal:</b> Internal EIGRP or intra/interarea OSPF routes <b>level-1:</b> IS-IS L1 routes <b>level-2:</b> IS-IS L2 routes <b>local:</b> Locally generated BGP routes <b>nssa-external:</b> NSSA external (Type 7 LSAs)
Route policy	<b>if route-type is</b> {interarea   internal   level-1   level-2   local   ospf-external-type-1   ospf- external-type-2   ospf-inter-area   ospf-intra-area   ospf-nssa-type-1   ospf-nssa-type-2   type-1   type-2   <i>parameter</i> }	Selects routes by route-type per protocol IS-IS: interarea, level-1, level-2 OSPF: ospf-intra-area, ospf inter-area, ospf-external-type-1, ospf-external- type-2, ospf-nssa-type-1, ospf-nssa- type-2, type-1, type-2

Table 13-3 Route Map and Route Policy Match Options

Table 13-4 provides the route map or route policy set actions that modifies the route as it redistributes into the destination protocol.

<b>Application</b>	<b>Set Action</b>	<b>Description</b>
Route map	<b>set as-path prepend</b> { <i>as-number-pattern</i>   <b>last-as</b> 1-10}	Prepends that AS_Path for the network prefix with the pattern specified, or from multiple iterations from neighboring autonomous system.
Route map	<b>set ip next-hop</b> { <i>ip-address</i>   <b>peer-address</b>   <b>self</b> }	Sets the next-hop IP address for any matching prefix. BGP dynamic manipulation uses the <b>peer-address</b> or <b>self</b> keywords.
Route map Route policy	<b>set level</b> { <b>level-1</b>   <b>level-1-2</b>   <b>level-2</b> }	Sets the IS-IS level on which the routes should be installed during redistribution.
Route map Route policy	<b>set local-preference</b> 0-4294967295	Sets the BGP PA local preference.
Route map	<b>set metric</b> { <b>+value</b>   <b>-value</b>   <i>value</i> }  * <i>value</i> parameters are 0–4294967295	Modifies the existing metric or sets the metric for a route.
Route policy	<b>set med</b> { <i>med</i>   <b>+ value</b>   <b>- value</b>   <b>igp-cost</b>   <b>max-</b> <b>reachable</b>   <i>parameter</i> }	Sets the BGP PA MED.  Using the + and - functions requires a value for modification, which can be parameterized as well.
Route policy	<b>set ospf-metric</b> { <i>metric</i>   <i>parameter</i> }	Sets the OSPF metric during redistribution.
Route policy	<b>set isis-metric</b> { <i>metric</i>   <i>parameter</i> }	Sets the metric for IS-IS routes during redistribution.
Route map Route policy	<b>set metric-type</b> { <b>external</b>   <b>internal</b>   <b>type-1</b>   <b>type-2</b> }	Sets the IS-IS metric type as external or internal; or sets the OSPF metric type as <b>type-1</b> or <b>type-2</b> .
Route map	<b>set origin</b> { <b>igp</b>   <b>incomplete</b> }	Sets the BGP PA origin.
Route policy	<b>set origin</b> { <b>egp</b>   <b>igp</b>   <b>incomplete</b>   <i>parameter</i> }	
Route map Route policy	<b>set weight</b> 0-65535	Sets the BGP PA weight.
Route policy	<b>set next-hop</b> { <b>ip-address</b>   <b>discard</b>   <b>peer-address</b>   <b>self</b>   <i>parameter</i> }	Sets the next-hop IP address for matching prefix. BGP dynamic manipulation uses the <b>peer-address</b> or <b>self</b> keywords. The <b>discard</b> keyword automatically discards the route.

**Table 13-4** Route Map and Route Policy Set Actions

### Source-Specific Behaviors

The following sections explain specific behaviors for each protocol from a source perspective.

## Connected Networks

A common scenario in service provider networks involves the need for external Border Gateway Protocol (BGP) peering networks to exist in the routing table by other internal BGP (iBGP) routers within the autonomous system. Instead of enabling the routing protocol on the interface so that the network is installed into the routing topology, the networks could be redistributed into the interior gateway protocol (IGP). Choosing to not enable a routing protocol on that link removes security concerns within the IGP.

Connected networks are the networks associated with primary and secondary IP addresses for any up interfaces that are not participating with the destination protocol. At times during redistribution, only select interfaces need to be redistributed. This is accomplished using a route map that selects only the desired interfaces.

**Example 13-6** provides a reference route policy and route map for selecting the specific connected network 192.168.1.1/32 on the Loopback 0 interface. IOS XR route policies do not allow the selection of networks by interfaces but by destination networks.

### **Example 13-6** *Selective Connected Network Redistribution*

[Click here to view code image](#)

```
IOS XR
route-policy RPL-LOOPBACK0
  if destination in (192.168.1.1/32) then
    pass
  endif
end-policy
```

```
IOS
route-map RM-LOOPBACK0 permit 10
  match interface Loopback0
```

## IS-IS

IS-IS selects routes from the L2 LSPDB for redistribution into the destination routing protocol. Using the optional level commands for redistribution or through route policies allows for the selection of routes from the L1 LSPDB, too. IS-IS provides only the routes learned from other routers during redistribution and does not include any IS-IS enabled connected interfaces.

**Figure 13-7** demonstrates a topology where IS-IS is redistributed into OSPF. Only the 10.45.1.0/24 network is redistributed into OSPF. The locally connected 10.34.1.0/24 on R3 is not included by IS-IS in the redistribution process. For the connected network (10.34.1.0/24) to be redistributed into OSPF, a second redistribution command is required with **connected** as the source protocol.

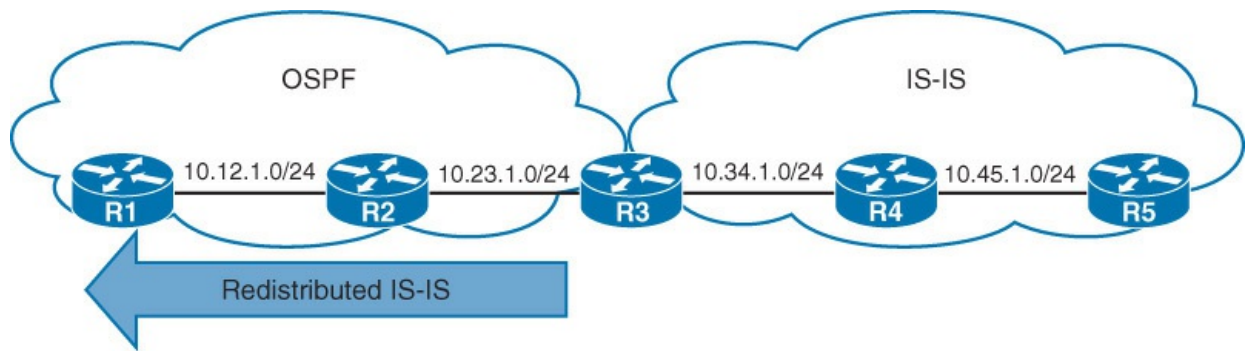


Figure 13-7 IS-IS Redistribution Topology

## BGP

By default, BGP redistributes only eBGP routes into IGP protocols. In Figure 13-8, R3 advertises the 192.168.3.3/32 network and R4 advertises the 192.168.4.4/32 network into BGP. R2 is redistributing BGP into OSPF, but only the 192.168.4.4/32 address is redistributed because it is an eBGP route. The iBGP route from R3 was not included because of BGP loop-prevention rules. It is assumed that the IGP routing topology already has a path to reach R3's network 192.168.3.3/32 because it is in the same autonomous system. BGP's default behavior requires that a route has an AS\_Path to redistribute into an IGP routing protocol.

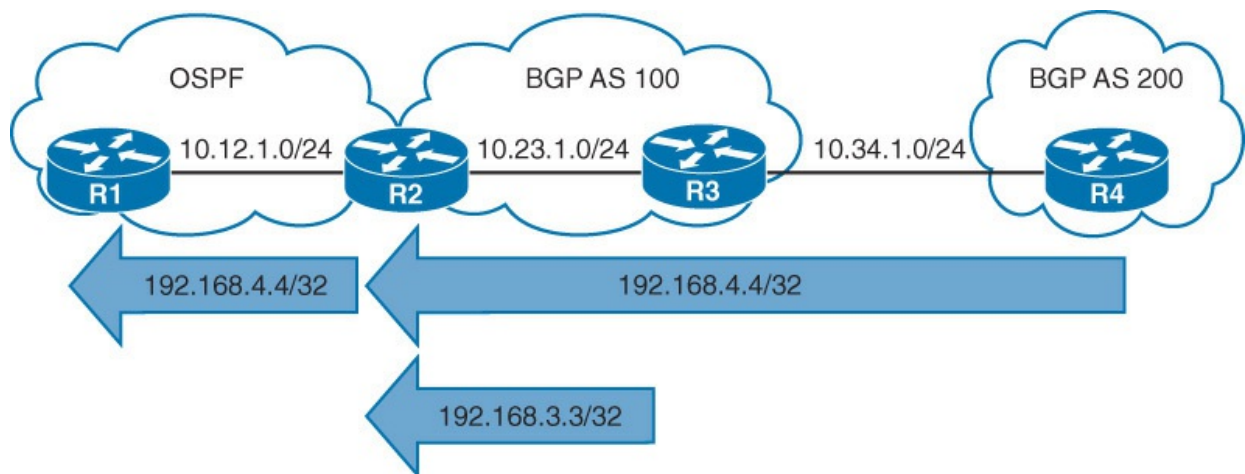


Figure 13-8 BGP Redistribution Topology

### Note

BGP is designed to handle a large routing table, whereas IGP's are not. Redistributing BGP into an IGP on a router with a larger BGP table (for example, the Internet table with 500,000+ routes) should use selective route redistribution. Otherwise, the IGP can become unstable in the routing domain and lead to packet loss.

BGP behavior can be changed so that all BGP routes are redistributed with the BGP configuration command **redistribute-internal** for IOS nodes, and IOS XR nodes use the command **bgp redistribute-internal**. To allow for the iBGP route 192.168.3.3/32 to redistribute into OSPF, the **redistribute-internal** command is required on R2.

### Note

Redistributing iBGP routes into an IGP could result in routing loops. A more logical solution is to advertise the network into the IGP.

## Destination-Specific Behaviors

The following sections explain specific behaviors for each protocol from a destination perspective. Redistributing between different processes of the same routing protocol or when redistributing into different routing protocol. The following section addresses all the protocols.

### EIGRP

External EIGRP routes are given an AD of 170 and use a default seed metric of infinity, which prevents the installation of the routes into the EIGRP topology table. The exception is that if an EIGRP autonomous system redistributes into another EIGRP autonomous system because all the path metrics are included during redistribution.

The default path metric can be changed from infinity to specific values for bandwidth, load, delay, reliability, and maximum transmission unit (MTU), thereby allowing for the installation into the EIGRP topology table. IOS and IOS XR routers can set the default metric with the address family configuration command **default-metric** *bandwidth delay reliability load mtu*. Delay is entered in tens of microseconds ( $\mu$ s).

Redistributing into EIGRP on IOS routers uses the command **redistribute** *source-protocol* [*metric bandwidth delay reliability load mtu*] [**route-map** *route-map-name*].

IOS XR routers redistribute into EIGRP with address-family configuration command **redistribute** *source-protocol* **route-policy** *route-policy-name*. Using a route map or route policy allows for metrics to be changed for each route, or the selective filtering of routes.

#### Note

Named mode EIGRP configuration occurs in the topology base configuration.

Figure 13-9 provides a topology example where XR2 mutually redistributes OSPF into EIGRP and where R3 mutually redistributes IS-IS into EIGRP. XR1 is advertising the Loopback 0 address of 192.168.1.1/32, and R4 is advertising the Loopback 0 address of 192.168.4.4/32.

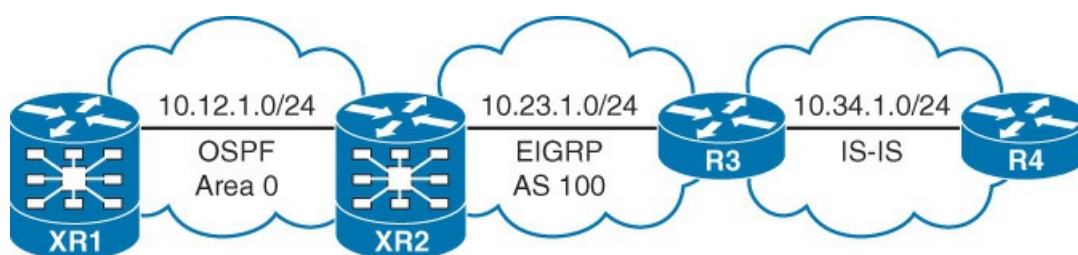


Figure 13-9 EIGRP Redistribution Topology

Example 13-7 demonstrates the relevant EIGRP configuration. XR2 and R3 use the **default-metric** configuration command, which overrides the seed value with the K values specified. All redistributed routes will use the **default-metric** values unless modified in a route map or route policy. Autonomous system classic and named mode configuration has been provided for R3. Notice that R3 specifies that IS-IS includes L1-L2 routes during redistribution; otherwise, no routes would have been redistributed because they are all L1 routes.

### Example 13-7 EIGRP Redistribution Configuration

[Click here to view code image](#)

**XR2**

```
router eigrp 100
  address-family ipv4
    default-metric 1000000 1 255 1 1500
    redistribute ospf 1
  interface GigabitEthernet0/0/0/1
```

**R3 (AS Classic Configuration)**

```
router eigrp 100
  default-metric 1000000 1 255 1 1500
  network 10.23.1.0 0.0.0.255
  redistribute isis level-1-2
```

**R3 (Named Mode Configuration)**

```
router eigrp NAMED-MODE
  !
  address-family ipv4 unicast autonomous-system 100
  !
  topology base
    default-metric 1000000 1 255 1 1500
    redistribute isis level-1-2
  exit-af-topology
  network 10.23.1.0 0.0.0.255
  exit-address-family
```

EIGRP seed metrics are overwritten by setting K values with the RPL command **set eigrp-metric bandwidth delay reliability load mtu** or the route map command **set metric bandwidth delay reliability load mtu**. Setting the metric on a prefix-by-prefix basis during redistribution provides a method of traffic engineering. [Example 13-8](#) shows the configuration without the use of the **default-metric** command, but by setting the EIGRP metric via route map or route policy.

**Example 13-8 EIGRP Redistribution with Route Map and Route Policy Configuration**

[Click here to view code image](#)

**XR2**

```
router eigrp 100
  address-family ipv4
    redistribute ospf 1 route-policy OSPF-2-EIGRP
  interface GigabitEthernet0/0/0/1
  !
  route-policy OSPF-2-EIGRP
    set eigrp-metric 1000000 1 255 1 1500
  end-policy
```

**R3**

```
router eigrp 100
  default-metric 1000000 1 255 1 1500
  network 10.23.1.0 0.0.0.255
  redistribute isis level-1-2 route-map ISIS-2-EIGRP
!
route-map ISIS-2-EIGRP permit 10
  set metric 1000000 1 255 1 1500
```

**Example 13-9** displays the EIGRP topology table with the redistributed routes highlighted. Notice that the 10.34.1.0/24 network is missing because IS-IS does not include connected networks as a source protocol.

### **Example 13-9** EIGRP Topology Table of Redistributed Routes

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show eigrp topology
IPv4-EIGRP Topology Table for AS(100)/ID(10.23.1.2)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.12.1.0/24, 1 successors, FD is 1310720, RIB is 10240
   via Redistributed (1310720/0)
P 10.23.1.0/24, 1 successors, FD is 1310720, RIB is 10240
   via Connected, GigabitEthernet0/0/0/1
P 192.168.1.1/32, 1 successors, FD is 1310720, RIB is 10240
   via Redistributed (1310720/0)
P 192.168.4.4/32, 1 successors, FD is 1966080, RIB is 15360
   via 10.23.1.3 (1966080/1310720), GigabitEthernet0/0/0/1
```

The redistributed routes are shown in the routing table with D EX and an AD of 170, as shown in **Example 13-10**.

### **Example 13-10** Verification of Routes

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route
! Output omitted for brevity

C   10.12.1.0/24 is directly connected, 00:45:25, GigabitEthernet0/0/0/0
C   10.23.1.0/24 is directly connected, 00:45:25, GigabitEthernet0/0/0/1
O   192.168.1.1/32 [110/2] via 10.12.1.1, 00:41:28, GigabitEthernet0/0/0/0
D EX 192.168.4.4/32 [170/15360] via 10.23.1.3, 00:32:04, GigabitEthernet0/0/0/1
```

```
R3#show ip route
```

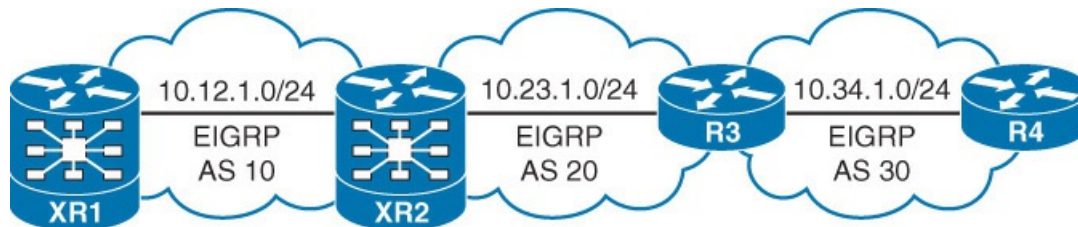
```
! Output omitted for brevity
```

```
Gateway of last resort is not set
```

```
C      10.11.12.0/24 is directly connected, GigabitEthernet0/0
D EX   10.12.1.0/24 [170/15360] via 10.23.1.2, 00:26:19, GigabitEthernet0/1
C      10.23.1.0/24 is directly connected, GigabitEthernet0/1
C      10.34.1.0/24 is directly connected, GigabitEthernet0/2
D EX   192.168.1.1 [170/15360] via 10.23.1.2, 00:26:19, GigabitEthernet0/1
C      192.168.3.3 is directly connected, Loopback0
i L1   192.168.4.4 [115/20] via 10.34.1.4, 00:26:42, GigabitEthernet0/2
```

### EIGRP-to-EIGRP Redistribution

Redistributing routes between EIGRP autonomous systems will preserve the path metrics during redistribution. **Figure 13-10** demonstrates a topology of multiple EIGRP autonomous systems. XR2 mutually redistributes routes between AS 10 and AS 20, and R3 mutually redistributes routes between AS 20 and AS 30. XR1 advertises Loopback 0 interface (192.168.1.1/32) into EIGRP AS 10, and R4 advertises the Loopback 0 interface (192.168.4.4/32) into EIGRP AS 30.



**Figure 13-10** Mutual EIGRP Redistribution Topology

**Example 13-11** demonstrates the configuration for XR2 and R3. The default seed metrics do not need to be set because they are maintained between EIGRP AS to EIGRP AS. R3 is using autonomous system classic configuration for AS 20 and EIGRP named mode configuration for AS 30.

### Example 13-11 EIGRP Mutual Redistribution Configuration

[Click here to view code image](#)

```
XR2
router eigrp 10
 address-family ipv4
  redistribute eigrp 20
 interface GigabitEthernet0/0/0/0
!
router eigrp 20
 address-family ipv4
  redistribute eigrp 10
 interface GigabitEthernet0/0/0/1
```



```

R3
router eigrp 20
 network 10.23.1.0 0.0.0.255
 redistribute eigrp 30
router eigrp NAMED-MODE
 !
 address-family ipv4 unicast autonomous-system 30
 !
 topology base
 redistribute eigrp 20
 exit-af-topology
 network 10.34.1.0 0.0.0.255
 exit-address-family

```

**Example 13-12** provides verification that XR1 has routes learned from AS 20 and AS 30, and R5 has routes learned from AS 10 and AS 20.

### **Example 13-12** *Verification of Routes*

[Click here to view code image](#)

```

RP/0/0/CPU0:XR1#show route eigrp

D EX 10.23.1.0/24 [170/15360] via 10.12.1.2, 00:14:57, GigabitEthernet0/0/0/0
D EX 10.34.1.0/24 [170/20480] via 10.12.1.2, 00:14:28, GigabitEthernet0/0/0/0
D EX 192.168.4.4/32 [170/25600] via 10.12.1.2, 00:09:51, GigabitEthernet0/0/0/0

```

```

R4#show ip route eigrp
! Output omitted for brevity

D EX    10.12.1.0/24 [170/3328] via 10.34.1.3, 00:15:43, GigabitEthernet0/1
D EX    10.23.1.0/24 [170/3072] via 10.34.1.3, 00:15:43, GigabitEthernet0/1
D EX    192.168.1.1 [170/3584] via 10.34.1.3, 00:15:43, GigabitEthernet0/1

```

**Example 13-13** displays the EIGRP topology table for the route 192.168.4.4/32 in AS 10 and AS 20. The EIGRP path metrics for bandwidth, reliability, load, and delay are the same between the autonomous systems. Notice that the feasible distance (2621440) is the same for both autonomous systems, but the reported distance is 0 for AS 10 and 1,966,080 for AS 20. The reported distance (RD) was reset when redistributed into AS 10.

### **Example 13-13** *Topology Table for 192.168.4.4/32*

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show eigrp topology 192.168.4.4/32
```

```
! Output omitted for brevity
```

```
IPv4-EIGRP AS(10): Topology entry for 192.168.4.4/32
```

```
Routing Descriptor Blocks:
```

```
10.23.1.3, from Redistributed, Send flag is 0x0
```

```
Composite metric is (2621440/0), Route is External
```

```
Minimum bandwidth is 1000000 Kbit
```

```
Total delay is 30000000 picoseconds
```

```
Reliability is 255/255
```

```
Load is 1/255
```

```
Minimum MTU is 1500
```

```
Hop count is 2
```

```
External data:
```

```
Originating router is 192.168.2.2 (this system)
```

```
AS number of route is 10
```

```
External protocol is EIGRP, external metric is 20480
```

```
IPv4-EIGRP AS(20): Topology entry for 192.168.4.4/32
```

```
Routing Descriptor Blocks:
```

```
10.23.1.3 (GigabitEthernet0/0/0/1), from 10.23.1.3, Send flag is 0x0
```

```
Composite metric is (2621440/1966080), Route is External
```

```
Minimum bandwidth is 1000000 Kbit
```

```
Total delay is 30000000 picoseconds
```

```
Reliability is 255/255
```

```
Load is 1/255
```

```
Minimum MTU is 1500
```

```
Hop count is 2
```

```
External data:
```

```
Originating router is 192.168.3.3
```

```
AS number of route is 30
```

```
External protocol is EIGRP, external metric is 15360
```

## OSPF

The AD for intra-area, interarea, and external OSPF routes are all set to 110. External OSPF routes are classified as Type 1 or Type 2, with Type 2 as the default setting. The seed metric is 1 for BGP-sourced routes and 20 for all other protocols. The exception is that if OSPF redistributes from another OSPF process, the path metric is transferred. The main differences between Type 1 and Type 2 External OSPF routes follow:

- Type 1 routes are preferred over Type 2.
- The Type 1 metric equals the redistribution metric plus the total path metric to the ASBR. In other words, as the LSA propagates away from the originating ASBR the metric increases.
- The Type 2 metric equals only the redistribution metric. The metric is the same for the router next to the Area System Border Router (ASBR) as the router 30 hops away from the originating ASBR. In the event that two Type 2 paths have the exact same metric, the lower forwarding cost is preferred. This is the default external metric type used by OSPF.

Redistributing into OSPF on IOS routers uses the command **redistribute source-protocol [subnets] [metric metric] [metric-type {1 | 2}] [tag 0-4294967295] [route-map route-map-name]**. If the optional **subnets** keyword is not included, only the classful networks are redistributed. The optional **tag** allows for a 32-bit route tag to be included on all redistributed

routes.

IOS XR routers redistribute into OSPF with the configuration command **redistribute** *source-protocol* [**metric** *metric*] [**metric-type** {**1** | **2**}] [**tag** *0-4294967295*] [**route-policy** *route-policy-name*]. Using a route map or route policy allows for metrics to be changed for each route, or the selective filtering of routes.

IOS and IOS XR allow for the metric and metric-type to be set during redistribution.

Figure 13-11 provides a topology example where XR2 will mutually redistribute EIGRP into OSPF and where R3 will mutually redistribute IS-IS into OSPF. XR1 is advertising the Loopback 0 interface (192.168.1.1/32), and R4 is advertising the Loopback 0 interface (192.168.4.4/32).



Figure 13-11 OSPF Redistribution Topology

Example 13-14 provides the relevant OSPF configuration. Notice that XR2 and R3 use different OSPF process numbers but are still able to form an adjacency. The OSPF process numbers are locally significant in linking the OSPF-enabled interfaces to a process, as shown later in this section.

### Example 13-14 OSPF Redistribution Configuration

[Click here to view code image](#)

```
XR2
router ospf 1
router-id 192.168.2.2
redistribute eigrp 100
area 1
interface GigabitEthernet0/0/0/2
!
```

```
R3
router ospf 2
router-id 192.168.3.3
redistribute isis level-1-2 subnets
network 10.23.1.3 0.0.0.0 area 1
```

Example 13-15 displays the Type 5 LSAs for the external networks in the OSPF domain. The routes 10.12.1.0/24, 192.168.1.1/32, and 192.168.4.4/32 successfully redistributed into OSPF, whereas the 10.34.1.0/24 network is missing because IS-IS does not redistribute connected networks. The redistributed networks are Type 2 with a metric of 20.

## Example 13-15 OSPF LSDB from R3

[Click here to view code image](#)

```
R3#show ip ospf database external
! Output omitted for brevity

OSPF Router with ID (192.168.3.3) (Process ID 2)

Type-5 AS External Link States

Link State ID: 10.12.1.0 (External Network Number )
Advertising Router: 192.168.2.2
Network Mask: /24
Metric Type: 2 (Larger than any link state path)
Metric: 20

Link State ID: 192.168.1.1 (External Network Number )
Advertising Router: 10.23.1.2
Network Mask: /32
Metric Type: 2 (Larger than any link state path)
Metric: 20

Link State ID: 192.168.4.4 (External Network Number )
Advertising Router: 192.168.3.3
Network Mask: /32
Metric Type: 2 (Larger than any link state path)
Metric: 20
```

In [Example 13-16](#), the redistributed routes appear in the routing table with O E2 for Type 2, and O E1 for Type 1 external routes. The routers do not explicitly set a metric type, so all the redistributed routes from the topology are type E2.

## Example 13-16 OSPF Route Redistribution Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route
! Output omitted for brevity

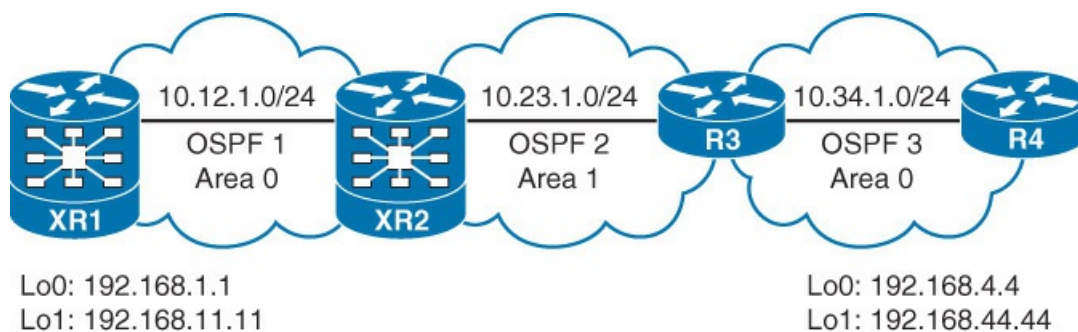
C 10.12.1.0/24 is directly connected, 00:24:15, GigabitEthernet0/0/0/0
C 10.23.1.0/24 is directly connected, 00:24:15, GigabitEthernet0/0/0/2
D 192.168.1.1/32 [90/3072] via 10.12.1.1, 00:17:36, GigabitEthernet0/0/0/0
O E2 192.168.4.4/32 [110/20] via 10.23.1.3, 00:04:39, GigabitEthernet0/0/0/2
```

```
R3#show ip route
! Output omitted for brevity

O E2 10.12.1.0/24 [110/20] via 10.23.1.2, 00:05:16, GigabitEthernet/0
C 10.23.1.0/24 is directly connected, GigabitEthernet0/0
C 10.34.1.0/24 is directly connected, GigabitEthernet0/1
O E2 192.168.1.1 [110/20] via 10.23.1.2, 00:05:16, GigabitEthernet0/0
i L1 192.168.4.4 [115/20] via 10.34.1.4, 00:25:56, GigabitEthernet0/1
```

## OSPF-to-OSPF Redistribution

Redistributing routes between OSPF processes will preserve the path metric during redistribution independence of the metric type. **Figure 13-12** demonstrates a topology of multiple OSPF processes and areas. XR2 redistributes routes between OSPF process 1 and OSPF process 2, and R3 redistributes between OSPF process 2 and OSPF process 3. XR2 and R3 set the metric type to 1 during redistribution so that the path metric increments. XR1 advertises the Loopback 0 interface (192.168.1.1/32) into OSPF process 1, and R4 advertises R4 advertises the Loopback 0 interface (192.168.4.4/32) into OSPF process 3.



**Figure 13-12** OSPF Multiprocess Redistribution

**Example 13-17** demonstrates the relevant configuration for XR2 and R3. Notice that the metric type is set at the time of redistribution into the destination protocol so that you can see the metric increase as the route travels between the OSPF processes.

## Example 13-17 OSPF Multiprocess Redistribution

[Click here to view code image](#)

```
XR2
router ospf 1
  redistribute ospf 2 metric-type 1
  area 0
  interface GigabitEthernet0/0/0/0
!
router ospf 2
  redistribute ospf 1 metric-type 1
  area 1
  interface GigabitEthernet0/0/0/2
```

```
R3
router ospf 2
  redistribute ospf 3 metric-type 1 subnets
  network 10.23.1.0 0.0.0.255 area 1
router ospf 3
  redistribute ospf 2 metric-type 1 subnets
  network 10.34.1.0 0.0.0.255 area 0
```

**Example 13-18** provides verification that XR1 has routes learned from OSPF process 3 and that R4 has routes learned from OSPF process 1.

## Example 13-18 Verification of OSPF Redistribution

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route ospf
```

```
! Output omitted for brevity
```

```
O E1 10.23.1.0/24 [110/21] via 10.12.1.2, 00:00:09, GigabitEthernet0/0/0/0
O E1 10.34.1.0/24 [110/3] via 10.12.1.2, 00:00:09, GigabitEthernet0/0/0/0
O E1 192.168.4.4/32 [110/4] via 10.12.1.2, 00:00:09, GigabitEthernet0/0/0/0
```

```
R4#show ip route ospf
```

```
! Output omitted for brevity
```

```
O E1 10.12.1.0/24 [110/22] via 10.34.1.3, 02:38:49, GigabitEthernet0/1
O E1 10.23.1.0/24 [110/2] via 10.34.1.3, 02:38:49, GigabitEthernet0/1
O E1 192.168.1.1 [110/4] via 10.34.1.3, 02:38:49, GigabitEthernet0/1
```

### OSPF Forwarding Address

OSPF Type 5 LSAs include a field known as the *forwarding address* that optimizes forwarding traffic when the source uses a shared network segment. The scenario defined in RFC 2328 is not common but is represented in [Figure 13-13](#). OSPF is enabled on all the links in Area 0 except for network 10.123.1.0/24. R1 forms an eBGP session with R2 (ASBR), which then redistributes the AS100 route 192.168.1.1/32 into the OSPF domain. R3 has direct connectivity to R1 but does not establish a BGP session with R1.

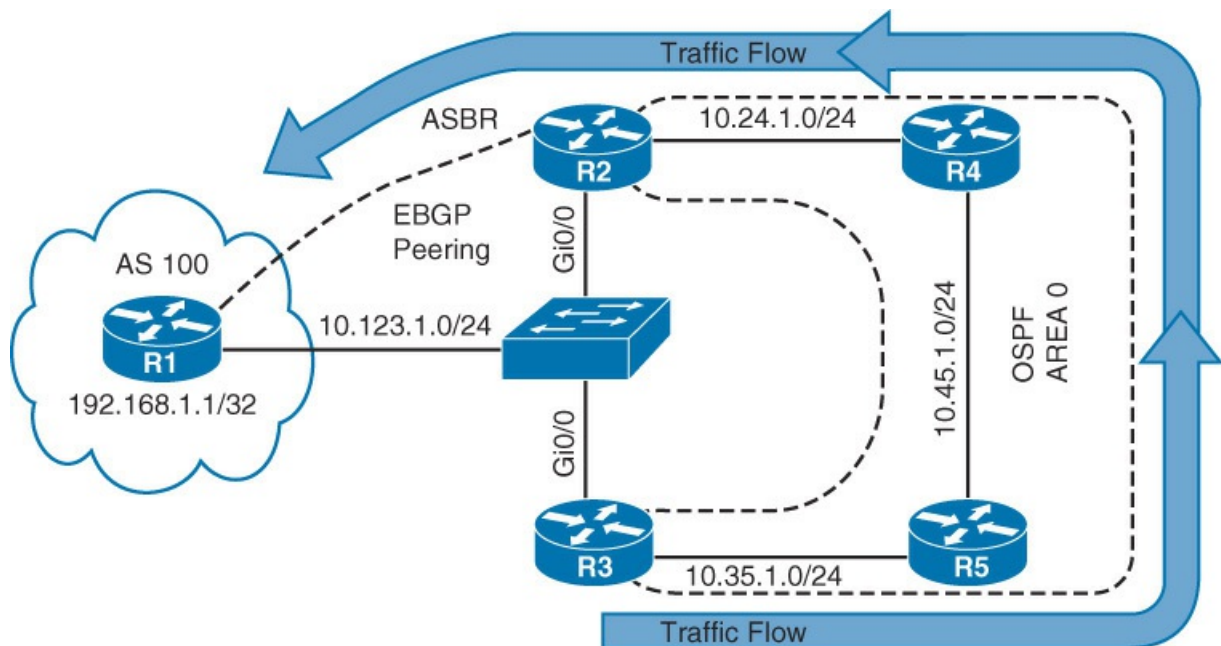


Figure 13-13 OSPF Forwarding Address Set to Default

[Example 13-19](#) displays the Type 5 LSA for the AS100 route for 192.168.1.1/32. The ASBR is identified as 10.123.1.2, which is the IP address that all OSPF routers will forward packets to reach the 192.168.1.1/32 network. Notice that the forwarding address is the default value of 0.0.0.0.

### Example 13-19 OSPF External LSA with Forwarding Address of 0.0.0.0

[Click here to view code image](#)

```

R3#show ip ospf database external
! Output omitted for brevity
      Type-5 AS External Link States

Routing Bit Set on this LSA in topology Base with MTID 0
LS Type: AS External Link
Link State ID: 192.168.1.1 (External Network Number )
Advertising Router: 10.123.1.2
Network Mask: /32
      Metric Type: 2 (Larger than any link state path)
      Metric: 1
      Forward Address: 0.0.0.0

```

Traffic from R3 (and R5) take the suboptimal route (R3 --> R5 --> R4 --> R2 --> R1), as shown in [Example 13-20](#). The optimal route would use the directly connected 10.123.1.0/24 network.

### Example 13-20 Verification of Suboptimal Routing

[Click here to view code image](#)

```

R3#trace 192.168.1.1
Tracing the route to 192.168.1.1
  0 10.35.1.5 0 msec 0 msec 1 msec
  1 10.45.1.4 0 msec 0 msec 0 msec
  2 10.24.1.2 1 msec 0 msec 0 msec
  3 10.123.1.1 1 msec * 0 msec

```

```

R5#trace 192.168.1.1
Tracing the route to 192.168.1.1
  0 10.45.1.4 0 msec 0 msec 0 msec
  1 10.24.1.2 1 msec 0 msec 0 msec
  2 10.123.1.1 1 msec * 0 msec

```

The forwarding address in OSPF Type 5 LSAs is specified in RFC 2328 for scenarios such as this. When the forwarding address is 0.0.0.0, all routers will forward packets to the ASBR, introducing the potential for suboptimal routing.

The OSPF forwarding address changes from 0.0.0.0 to the next-hop IP address in the source routing protocol when

- OSPF is enabled on the ASBR's interface that points to the next-hop IP address.
- That interface is not set to passive.
- That interface is a broadcast or nonbroadcast OSPF network type.

When the forwarding address is set to a value besides 0.0.0.0, the OSPF routers will forward traffic only to the forwarding address. OSPF has been enabled on the R2's and R3's Ethernet interface connected to the 10.123.1.0/24 network, as shown in [Figure 13-14](#). The interface is

Ethernet, which defaults to the broadcast OSPF network type and all conditions have been met.

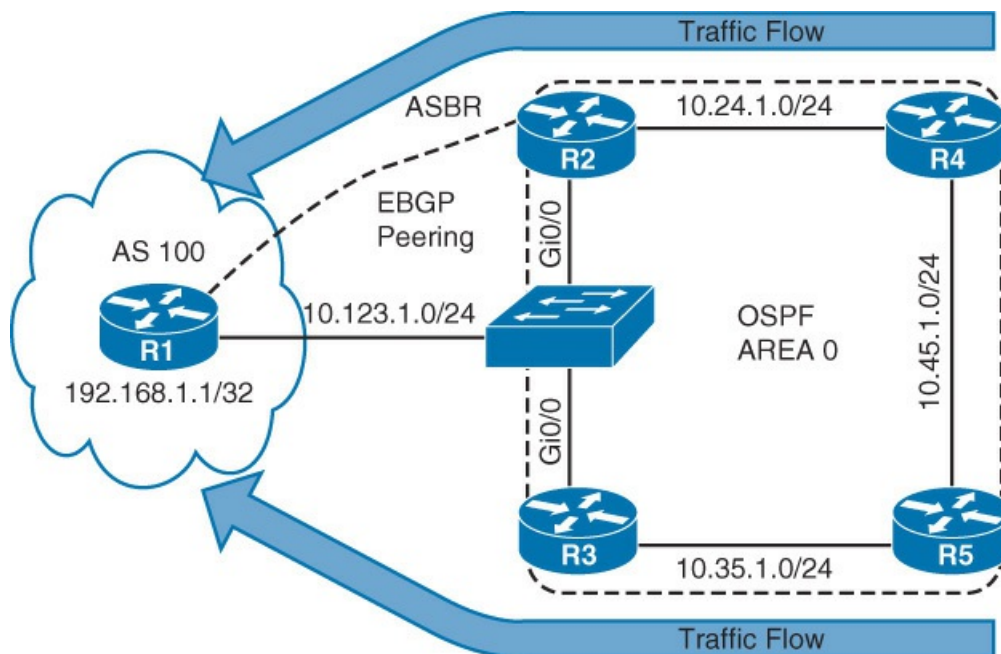


Figure 13-14 OSPF Forwarding Address Set to Nondefault

Example 13-21 provides the Type 5 LSA for the 192.168.1.1/32 network. Now that OSPF has been enabled on R2's 10.123.1.2 interface and the interface is a broadcast network type, the forwarding address has changed from 0.0.0.0 to 10.123.1.1.

### Example 13-21 OSPF External LSA with a Forwarding Address of 10.123.1.1

[Click here to view code image](#)

```
R3#show ip ospf database external
! Output omitted for brevity
Type-5 AS External Link States1

Options: (No TOS-capability, DC)
LS Type: AS External Link
Link State ID: 192.168.1.1 (External Network Number )
Advertising Router: 10.123.1.2
Network Mask: /32
Metric Type: 2 (Larger than any link state path)
Metric: 1
Forward Address: 10.123.1.1
```

Example 13-22 verifies that connectivity from R3 and R5 now takes the optimal path because the forwarding address has changed to 10.123.1.1.

### Example 13-22 Verification of Optimal Routing

[Click here to view code image](#)



```
R3#trace 192.168.1.1
Tracing the route to 192.168.1.1
 1 10.123.1.1 0 msec * 1 msec
```

```
R5#trace 192.168.1.1
Tracing the route to 192.168.1.1
 1 10.35.1.3 0 msec 0 msec 1 msec
 2 10.123.1.1 0 msec * 1 msec
```

If the Type 5 LSA forwarding address is not a default value, the address must be an intra-area or interarea OSPF route. If the route does not exist, the LSA is ignored and does not install into the RIB. This ensures that there are at least two routes directly connected to the external next-hop address. Otherwise, there is no reason to include the forwarding address in the LSA.

#### Note

The OSPF forwarding address optimizes forwarding toward the destination network, but return traffic is unaffected. In [Figure 13-14](#), outbound traffic from R3 or R5 still exists in R3's Gi0/0 interface, but return traffic is sent directly to R1.

## IS-IS

External routes are learned from outside of the IS-IS routing domain. The AD for IS-IS does not differentiate between internal and external routes and is set to 115. External IS-IS routes are redistributed as L2 routes into IS-IS and use an internal metric with a default seed metric of 0. IS-IS does not include connected networks during redistribution.

IOS routers perform redistribution with the IS-IS configuration command **redistribute source-protocol** [**metric** *metric*] [**metric-type** {**internal** | **external**}] [**route-map** *route-map-name*] [**level-1** | **level-1-2** | **level-2**]. IOS XR routers redistribute into IS-IS with configuration command **redistribute source-protocol** [**metric** *metric*] [**metric-type** {**internal** | **external**}] [**route-policy** *route-policy-name*].

#### Note

External routes use an **internal** or **external** metric as part of the IS-IS best path calculation. External routes with internal metrics are directly comparable in the best path algorithm with internal routes. External routes with external metrics are compared differently from internal routes and are less preferred to external routes with internal metrics. IS-IS best path calculation is explained in [Chapter 9, "Advanced IS-IS."](#)

[Figure 13-15](#) provides a topology example where XR2 mutually redistributes between EIGRP and IS-IS and where R3 mutually redistributes between IS-IS and OSPF. XR1 is advertising the Loopback 0 interface (192.168.1.1/32), and R4 is advertising the Loopback 0 interface (192.168.4.4/32).

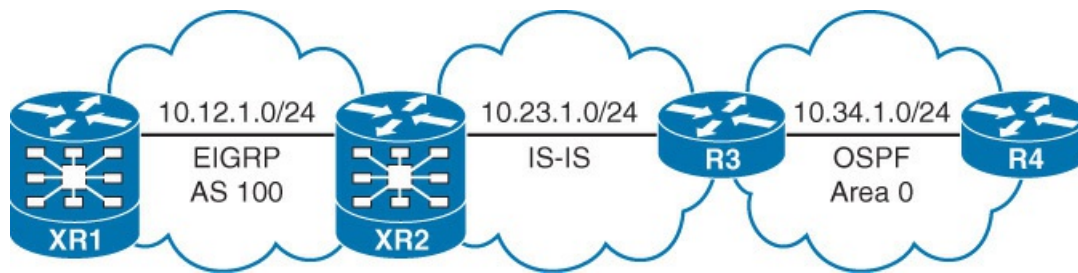


Figure 13-15 IS-IS Interarea Topology

Example 13-23 provides the relevant IS-IS configuration.

### Example 13-23 IS-IS Redistribution Configuration

[Click here to view code image](#)

```
XR2
router isis ISIS
 net 49.0023.0000.0000.0002.00
 address-family ipv4 unicast
  redistribute eigrp 100
 !
 interface GigabitEthernet0/0/0/2
  address-family ipv4 unicast
```

```
R3
router isis
 net 49.0023.0000.0000.0003.00
 redistribute ospf 1
```

Example 13-24 verifies that the 192.168.1.1/32 and 192.168.4.4/32 routes were redistributed into IS-IS. Notice that the metric is 0, but the path metric will still increment from router to router using the IS-IS topology.

### Example 13-24 IS-IS LSPDB

[Click here to view code image](#)

```
R3#show isis database level-2 detail
! Output omitted for brevity

IS-IS Level-2 Link State Database:
LSPID          LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
XR2.00-00      0x00000007   0xA63B        950            0/0/0
  Hostname: XR2
  Metric: 0
  IP-External 192.168.1.1 255.255.255.255
R3.00-00      * 0x00000006   0x02D5        1016           0/0/0
  Hostname: R3
  IP Address: 10.23.1.3
  Metric: 0
  IP-External 192.168.4.4 255.255.255.255
```

Example 13-25 verifies the routes are distributed into XR2's and R3's routing table. IS-IS

redistributes routes into the L2 LSPDB by default.

### Example 13-25 IS-IS Redistribution Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route
! Output omitted for brevity

C   10.12.1.0/24 is directly connected, 00:11:47, GigabitEthernet0/0/0/0
C   10.23.1.0/24 is directly connected, 00:11:47, GigabitEthernet0/0/0/2
i L2 10.34.1.0/24 [115/10] via 10.23.1.3, 00:01:34, GigabitEthernet0/0/0/2
D   192.168.1.1/32 [90/130816] via 10.12.1.1, 00:03:56, GigabitEthernet0/0/0/0
i L2 192.168.4.4/32 [115/10] via 10.23.1.3, 00:01:34, GigabitEthernet0/0/0/2
```

```
R3#show ip route
! Output omitted for brevity

i L2   10.12.1.0/24 [115/10] via 10.23.1.2, 00:01:14, GigabitEthernet0/0
C      10.23.1.0/24 is directly connected, GigabitEthernet0/0
C      10.34.1.0/24 is directly connected, GigabitEthernet0/1
i L2   192.168.1.1 [115/10] via 10.23.1.2, 00:01:14, GigabitEthernet0/0
O      192.168.4.4 [110/2] via 10.34.1.4, 00:06:31, GigabitEthernet0/1
```

### IS-IS to IS-IS Redistribution

Redistributing routes between IS-IS processes does not preserve the path metric. The path metric is always set to 0 regardless of the source protocol. IOS routers use the command **redistribute isis instance-id ip [metric metric] [route-map route-map-name]** for redistribution between other IS-IS processes. IOS XR routers use the IS-IS address family configuration command **redistribute isis instance-id [level-1 | level-1-2 | level-2] [metric metric] [metric-type {external | internal}] route-policy route-policy-name** for redistribution between other IS-IS processes. Because IS-IS selects only routes from the L2 LSPDB as a source protocol, a route map is required to select routes from the L1 LSPDB.

Figure 13-16 provides a topology example where XR2 mutually redistributes between IS-IS instance INSTANCE-A and INSTANCE-B and where R3 mutually redistributes between IS-IS instance INSTANCE-B and INSTANCE-C. XR1 is advertising the Loopback 0 interface (192.168.1.1/32), and R4 is advertising the Loopback 0 interface (192.168.4.4/32).

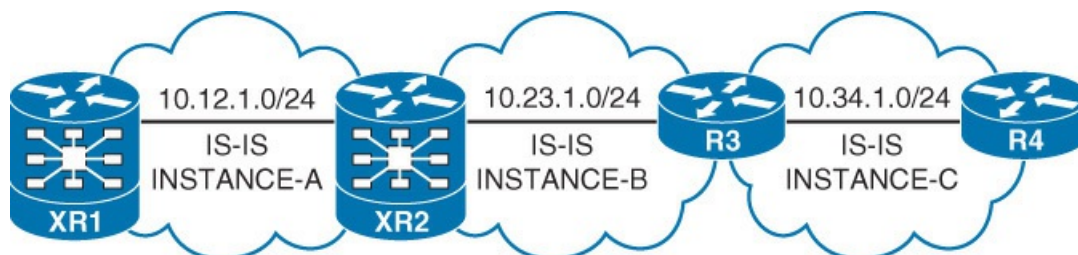


Figure 13-16 IS-IS Multiprocess Topology

Example 13-26 demonstrates the IS-IS configuration. R4's Loopback 0 interface is advertised as an L1 route. R3 uses a route map to allow L1 and L2 routes to redistribute into the other IS-IS process. If the route map were omitted, R4's Loopback 0 interface would not be redistributed into IS-IS INSTANCE-B.

## Example 13-26 IS-IS Multiprocess Redistribution Configuration

[Click here to view code image](#)

```
XR2
router isis INSTANCE-A
 net 49.0012.1000.0000.0002.00
 address-family ipv4 unicast
  redistribute isis INSTANCE-B
 !
 interface GigabitEthernet0/0/0/0
  address-family ipv4 unicast
  !
router isis INSTANCE-B
 net 49.0023.2000.0000.0002.00
 address-family ipv4 unicast
  redistribute isis INSTANCE-A
 !
 interface GigabitEthernet0/0/0/2
  address-family ipv4 unicast
```

```
R3
router isis INSTANCE-B
 net 49.0023.2000.0000.0003.00
 redistribute isis INSTANCE-C ip route-map ISISLEVEL
 !
router isis INSTANCE-C
 net 49.0034.3000.0000.0003.00
 redistribute isis INSTANCE-B ip route-map ISISLEVEL
 !
route-map ISISLEVEL permit 10
 match route-type level-1
 !
route-map ISISLEVEL permit 20
 match route-type level-2
```

Example 13-27 verifies that the routes have fully redistributed between all the IS-IS instances on the routes XR1, XR2, R3, and R4.

## Example 13-27 IS-IS Multitopology Redistribution Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show route
! Output omitted for brevity

C   10.12.1.0/24 is directly connected, 01:30:17, GigabitEthernet0/0/0/0
i L2 10.23.1.0/24 [115/10] via 10.12.1.2, 00:11:14, GigabitEthernet0/0/0/0
i L2 192.168.4.4/32 [115/10] via 10.12.1.2, 00:02:35, GigabitEthernet0/0/0/0

RP/0/0/CPU0:XR2#show route
! Output omitted for brevity
```

```
C 10.12.1.0/24 is directly connected, 01:30:47, GigabitEthernet0/0/0/0
C 10.23.1.0/24 is directly connected, 01:30:47, GigabitEthernet0/0/0/2
i L1 192.168.1.1/32 [115/20] via 10.12.1.1, 01:30:31, GigabitEthernet0/0/0/0
i L2 192.168.4.4/32 [115/10] via 10.23.1.3, 00:02:53, GigabitEthernet0/0/0/2
```

```
R3#show ip route
```

```
! Output omitted for brevity
```

```
i L2 10.12.1.0/24 [115/10] via 10.23.1.2, 00:07:36, GigabitEthernet0/0
C 10.23.1.0/24 is directly connected, GigabitEthernet0/0
C 10.34.1.0/24 is directly connected, GigabitEthernet0/1
i L2 192.168.1.1 [115/10] via 10.23.1.2, 00:07:36, GigabitEthernet0/0
i L1 192.168.4.4 [115/20] via 10.34.1.4, 00:07:12, GigabitEthernet0/1
```

```
R4#show ip route
```

```
! Output omitted for brevity
```

```
i L2 10.12.1.0/24 [115/10] via 10.34.1.3, 00:03:11, GigabitEthernet0/1
C 10.34.1.0/24 is directly connected, GigabitEthernet0/1
i L2 192.168.1.1 [115/10] via 10.34.1.3, 00:03:11, GigabitEthernet0/1
C 192.168.4.4 is directly connected, Loopback0
```

Note

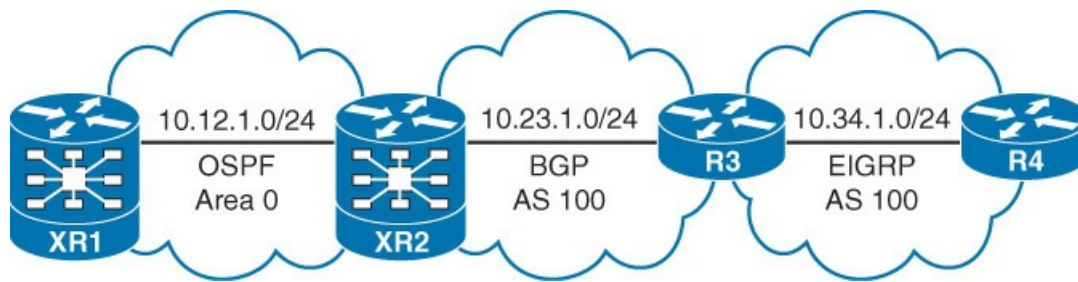
Chapter 9 explains IS-IS *route leaking*, which is redistribution between IS-IS levels in the same IS-IS instance. The usage of a route map or route policy allows for selective route leaking.

## BGP

Redistributing routes into BGP does not require a seed metric because it is a path vector protocol. Redistributed routes have the following BGP attributes set:

- Origin is set to incomplete.
- Next-hop address is set to the IP address of the source protocol.
- The weight is set to 32,768.
- The MED is set to the path metric of the source protocol.

Figure 13-17 displays a topology example where XR2 mutually redistributes between OSPF and BGP and where R3 mutually redistributes between EIGRP AS100 and BGP. XR1 is advertising the Loopback 0 interface (192.168.1.1/32), and R4 is advertising the Loopback 0 interface (192.168.4.4/32) into the appropriate protocols.



**Figure 13-17** BGP Redistribution Topology

**Example 13-28** demonstrates the BGP configuration with XR2 redistributing OSPF into BGP on XR2 and R3 redistributing EIGRP into BGP.

### **Example 13-28** BGP Redistribution Configuration

[Click here to view code image](#)

#### **XR2**

```
router bgp 100
  bgp redistribute-internal
  address-family ipv4 unicast
  redistribute ospf 1
  !
  neighbor 10.23.1.3
  remote-as 100
  address-family ipv4 unicast
```

#### **R3**

```
router bgp 100
  no bgp default ipv4-unicast
  neighbor 10.23.1.2 remote-as 100
  !
  address-family ipv4
  bgp redistribute-internal
  redistribute eigrp 100
  neighbor 10.23.1.2 activate
  exit-address-family
```

**Example 13-29** displays the BGP table for AS100. Notice that the 192.168.1.1/32 and 192.168.4.4/32 networks have installed into the BGP table.

### **Example 13-29** BGP Route Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 10.12.1.0/24	0.0.0.0	0		32768	?
*>i10.34.1.0/24	10.23.1.3	0	100	0	?
*> 192.168.1.1/32	10.12.1.1	2		32768	?
*>i192.168.4.4/32	10.34.1.4	130816	100	0	?

```
Processed 4 prefixes, 4 paths
```

**Example 13-30** displays detailed BGP path information for the redistributed routes. Notice that the origin is incomplete and that the metric matches the IGP metric at the time of redistribution.

### Example 13-30 Verification of BGP Routes

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show bgp ipv4 unicast 192.168.1.1
```

```
! Output omitted for brevity
```

```
Paths: (1 available, best #1)
```

```
Path #1: Received by speaker 0
```

```
Local
```

```
10.12.1.1 from 0.0.0.0 (192.168.2.2)
```

```
Origin incomplete, metric 2, localpref 100, weight 32768, valid,  
redistributed, best, group-best
```

```
Received Path ID 0, Local Path ID 1, version 5
```

```
R3#show bgp ipv4 unicast 192.168.4.4
```

```
BGP routing table entry for 192.168.4.4/32, version 3
```

```
Paths: (1 available, best #1, table default)
```

```
Advertised to update-groups:
```

```
1
```

```
Refresh Epoch 1
```

```
Local
```

```
10.34.1.4 from 0.0.0.0 (10.34.1.3)
```

```
Origin incomplete, metric 130816, localpref 100, weight 32768, valid, sourced, best
```

#### Note

Redistributing routes from OSPF to BGP does not include OSPF external routes by default. The optional **match external [1 | 2]** keyword is required to redistribute OSPF external routes.

## CHALLENGES WITH REDISTRIBUTION

Highly available network designs remove single points of failure through redundancy. When redistributing routes between protocols there must be at least two points of redistribution in the network to ensure that connectivity is maintained during a failure. When performing redistribution the following issues may arise:

- Route feedback

- Suboptimal routing

## ■ Invalid routing tables

## ■ Routing loops

### Route Feedback

Redistributing between different routing protocols causes a loss of path information preventing a complete topology of the routers between the source and destination networks. This can cause *route feedback*, which is a redistributed route that is redistributed back into the original routing domain at a different point in the network topology.

In [Figure 13-18](#), two routing domains exist (domain A and domain B), and R2 and R4 are mutually redistributing between the two domains. The 10.1.1.0/24 route from domain A is redistributed into domain B by R2 and eventually reaches R4. R4 then redistributes the 10.1.1.0/24 network back into domain A as route feedback. Incomplete routing topologies and route feedback introduce the possibility for suboptimal routing or routing loops in a topology. For example, does R5 use the routing information from R4 or R1 to forward packets to the 10.1.1.0/24 network?

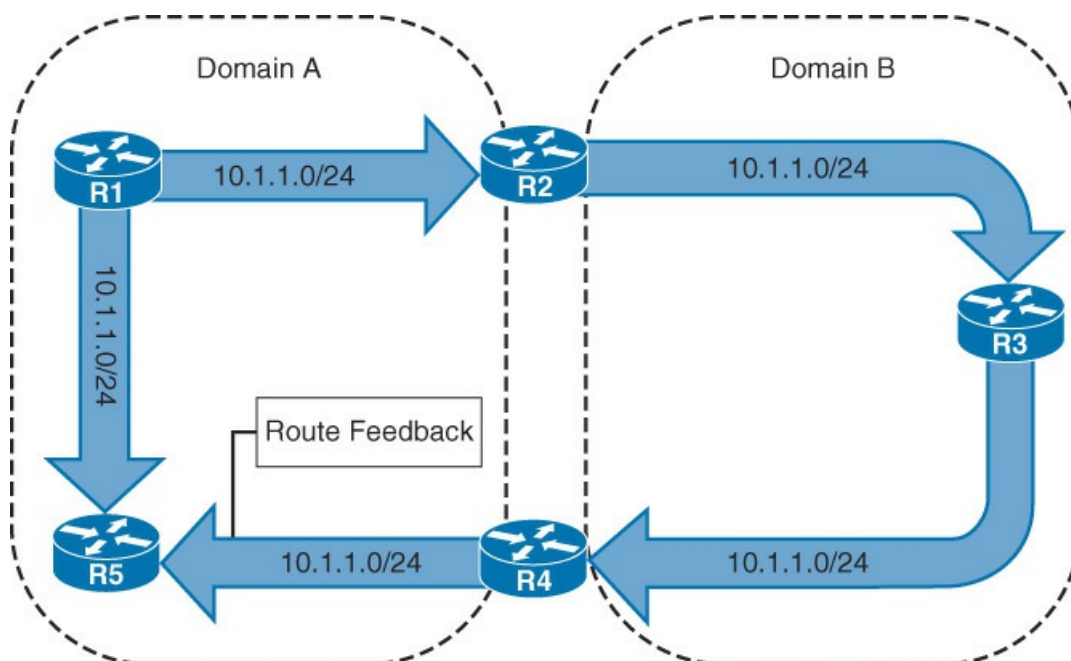


Figure 13-18 Route Feedback

The routing protocols discussed in this book (EIGRP, OSPF, IS-IS, and BGP) incorporate logic that drastically reduces the challenges observed with route feedback when compared to other protocols such as RIP, but problems can still arise.

Overcoming these challenges often involves making the routing protocol that initially advertises the network dominant over the other routing protocol. The dominant routing protocol is known as the *primary routing protocol*, and the other protocol is known as the *secondary routing protocol*.

### Suboptimal Routing

Routing protocols try to find the best path to the destination network but require a complete topology to find the best path to the destination. If all the routers between the source and destination networks do not use the same routing protocol, it is possible that packets do not use the optimal path and so forward traffic suboptimally.



Figure 13-19 displays a network topology that uses EIGRP on R1, R3, and R5 and OSPF on R1, R2, and R5. The path selected by R5 that connects it to R1 is suboptimal. R5 will select the path (R5 --> R4 --> R3 --> R1) that uses the slower serial links over the faster Ethernet links because EIGRP's AD (90) is lower than OSPF's AD (110). If all five routers were to use the same routing protocol, the optimal path with the Ethernet links would be used.

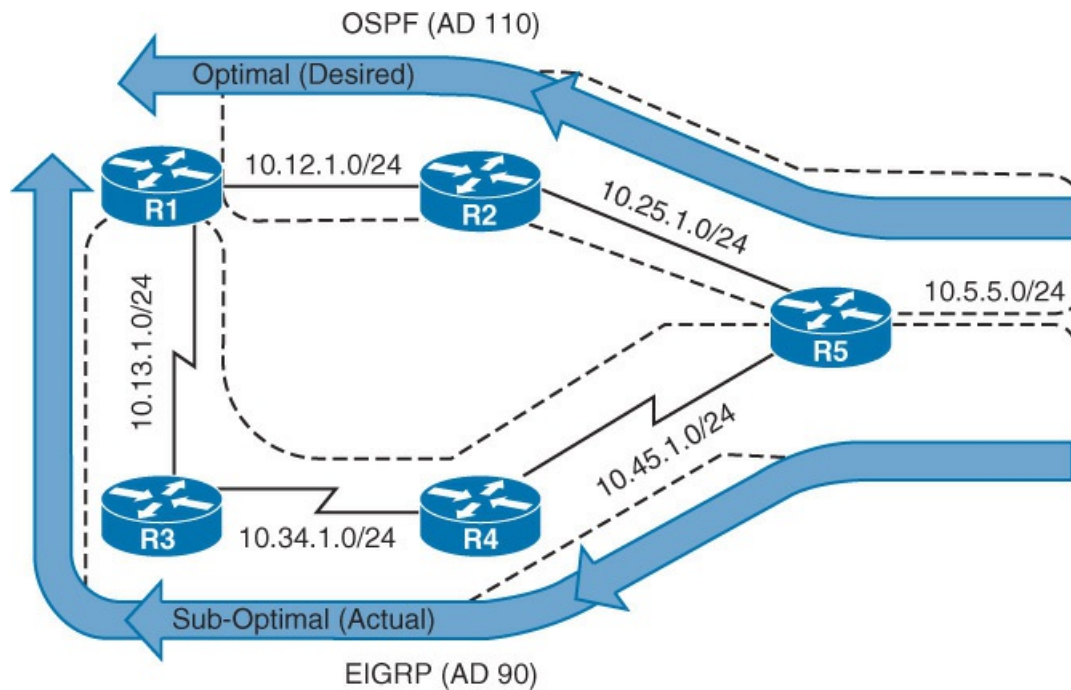


Figure 13-19 Suboptimal Routing Because of Administrative Distance

Figure 13-20 is a simple topology with two routing domains (EIGRP and OSPF) to explain the problems caused by route feedback. R1, R2, and XR3 use EIGRP as the routing protocol, and R2, XR3, R4, and R5 use OSPF for their routing protocol. R1 is redistributing the 10.1.1.0/24 connected network into EIGRP.

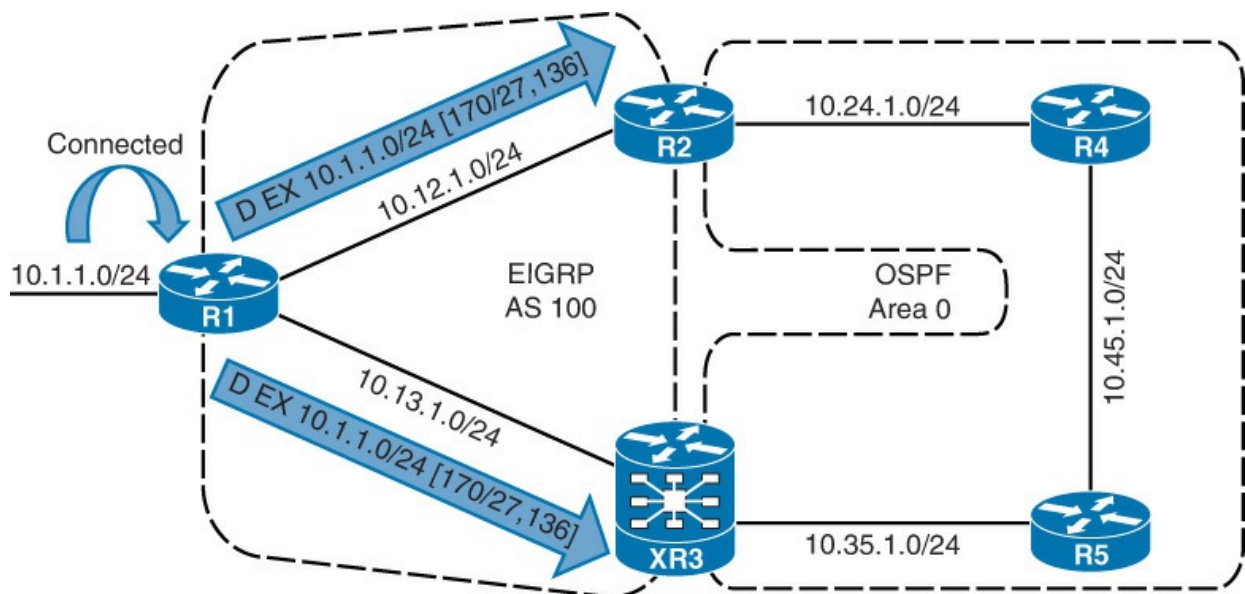


Figure 13-20 Multipoint Redistribution Topology

Example 13-31 provides R1's EIGRP configuration that is performing the redistribution of the

10.1.1.0/24 connected network. The delay K value was explicitly set to 5 to represent a 50 μs delay during redistribution to demonstrate some of the challenges with mutual redistribution on multiple routers. The incremental delay simulates the 10.1.1.0/24 route being further away from R2 and XR3 without having to add routers to the topology to simplify the following illustrations.

### Example 13-31 R1's EIGRP Configuration

[Click here to view code image](#)

```
router eigrp 100
 network 10.12.0.0 0.0.255.255
 network 10.13.0.0 0.0.255.255
 redistribute connected metric 100000 5 255 1 1500
```

Figure 13-21 builds upon the previous topology with the mutual redistribution between OSPF and EIGRP on R2 and XR3. R2 redistributes the 10.1.1.0/24 route into OSPF with an AD of 110, and is eventually advertised to XR3 via OSPF as an O E2 route.

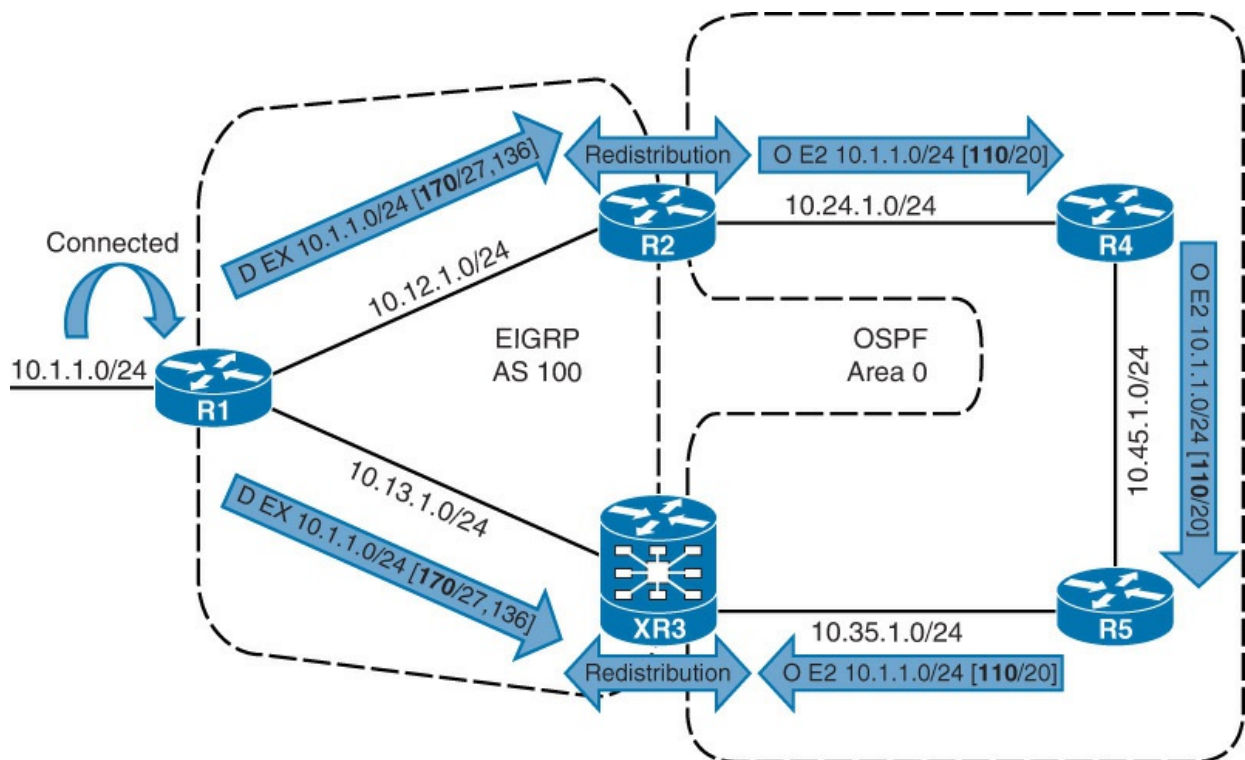


Figure 13-21 Suboptimal Routing Because of Route Feedback

Upon receipt of R1's EIGRP advertisement, XR3 installs the 10.1.1.0/24 route with a next hop of 10.13.1.1 learned via EIGRP with an AD of 170 into the RIB. XR3 eventually receives R2's Type 5 LSA for the 10.1.1.0/24 network, and because OSPF has a lower AD, XR3 removes the EIGRP-learned route from the RIB and installs the OSPF-learned route instead.

Example 13-32 displays XR3's RIB entry for the 10.1.1.0/24 network. The O E2 route was redistributed into OSPF by R2 (10.24.1.2) and advertised from R5 (10.35.1.5).

### Example 13-32 R3's Routing Entry for 10.1.1.0/24

[Click here to view code image](#)

```
RP/0/0/CPU0:XR3#show route 10.1.1.0

Routing entry for 10.1.1.0/24
  Known via "ospf 1", distance 110, metric 20, type extern 2
  Routing Descriptor Blocks
    10.35.1.5, from 10.24.1.2, via GigabitEthernet0/0/0/3
      Route metric is 20
  No advertising protos.
```

Traffic from XR3 takes the suboptimal path (XR3 --> R5 --> R4 --> R2 --> R1) instead of the optimal path (XR3 --> R1) to reach the 10.1.1.0/24 network. [Example 13-33](#) displays XR3's traceroute to the 10.1.1.0/24 network.

### **Example 13-33** *Suboptimal Path from R3 to 10.1.1.1*

[Click here to view code image](#)

```
RP/0/0/CPU0:XR3#traceroute 10.1.1.1

Tracing the route to 10.1.1.1

 1  10.35.1.5 1 msec  0 msec  0 msec
 2  10.45.1.4 0 msec  0 msec  0 msec
 3  10.24.1.2 0 msec  0 msec  0 msec
 4  10.12.1.1 0 msec  *  1 msec
```

The scope of the suboptimal routing is not restricted to XR3. XR3 cannot redistribute the 10.1.1.0/24 EIGRP route into OSPF because the EIGRP route is not installed in the RIB. R5 will use a suboptimal path too (R5 --> R4 --> R2 --> R1).

Another important concept in [Example 13-34](#) demonstrates the EIGRP topology for the 10.1.1.0/24 network where there are two EIGRP paths. The first path originates from R1 (192.168.1.1), and the second path is redistributed by XR3 (192.168.3.3). XR3 is able to redistribute the 10.1.1.0/24 network into EIGRP because of its mutual redistribution configuration and the fact that OSPF installs the 10.1.1.0/24 network in the RIB instead of EIGRP.

XR3's path has a lower path metric (2,816) than R1's path, which has a path metric of 27,136. Imagine that other routers were inserted into the topology between R1 and XR3. They would receive an advertisement for the 10.1.1.0/24 network from R1 and XR3. If XR3's metrics are lower, the suboptimal routing is further extended into the EIGRP routing domain.

### **Example 13-34** *EIGRP Topology Table*

[Click here to view code image](#)

```
RP/0/0/CPU0:XR3#show eigrp topology 10.1.1.0/24
```

```
! Output omitted for brevity
```

```
IPv4-EIGRP AS(100): Topology entry for 10.1.1.0/24
```

```
State is Passive, Query origin flag is 1, 1 Successor(s), FD is 2816
```

```
Routing Descriptor Blocks:
```

```
10.35.1.5, from Redistributed, Send flag is 0x0
```

```
Composite metric is (2816/0), Route is External
```

```
External data:
```

```
Originating router is 192.168.3.3 (this system)
```

```
AS number of route is 100
```

```
External protocol is OSPF, external metric is 20
```

```
10.13.1.1 (GigabitEthernet0/0/0/2), from 10.13.1.1, Send flag is 0x0
```

```
Composite metric is (27136/26880), Route is External, not in RIB
```

```
External data:
```

```
Originating router is 192.168.1.1
```

```
AS number of route is 0
```

```
External protocol is OSPFv3, external metric is 0
```

#### Note

The timing of route advertisements dictates the point of suboptimal routing between R2 and XR3. If the R1-R2 EIGRP adjacency was delayed during establishment, the suboptimal routing would occur from R2's perspective. For this section, XR3 will always be the timed failure point.

Suboptimal routing can be resolved by preventing the secondary routing protocol from installing the route in the RIB on the redistributing router. [Example 13-35](#) provides R2's and XR3's OSPF configuration, where the OSPF distribute list prevents OSPF from installing the 10.1.1.0/24 route into the RIB, which provides the following benefits:

- EIGRP will always install the 10.1.1.0/24 entry into the RIB.
- The EIGRP route will always redistribute into OSPF because EIGRP installs the route into the RIB.

This change allows XR3 and R5 to use the optimal path toward the 10.1.1.0/24 network. The negative aspect to this solution is that R2 and XR3 lose redundancy to the 10.1.1.0/24 network while R4 and R5 maintain redundancy.

### **Example 13-35** Configuration to Resolve Suboptimal Routing

[Click here to view code image](#)

#### R2

```
router ospf 1
 redistribute eigrp 100 subnets
 network 10.24.1.0 0.0.0.255 area 0
 distribute-list prefix BLOCK-TEN-ONE in
!
ip prefix-list BLOCK-TEN-ONE seq 5 deny 10.1.1.0/24
ip prefix-list BLOCK-TEN-ONE seq 10 permit 0.0.0.0/0 le 32
```

XR3

```
router ospf 1
  distribute-list route-policy BLOCK-TEN-ONE in
  redistribute eigrp 100
  area 0
    interface GigabitEthernet0/0/0/3
  !
route-policy BLOCK-TEN-ONE
  if not destination in (10.1.1.0/24) then
    pass
  endif
end-policy
```

Note

Distribute lists were explained earlier in Chapter 12, "Advanced Route Manipulation."

### Invalid Routing Tables

Route feedback from the secondary routing domain can lead to an invalid routing table in the primary routing domain. It is possible for the route feedback to preempt the originating network advertisement so that topology changes are not accurately reflected resulting in an *invalid routing table*.

In Figure 13-22, R1 is advertising the 10.1.1.0/24 network into domain A. R3 redistributes the route into domain B, which is then redistributed back into domain A. If R1 withdraws the 10.1.1.0/24 network from domain A, it is possible for R5 to advertise the route back to domain A after R1 withdraws it. The propagation of the route keeps the original route in the routing table for both domains, even though the network is not available. This is known as an *invalid routing table*.

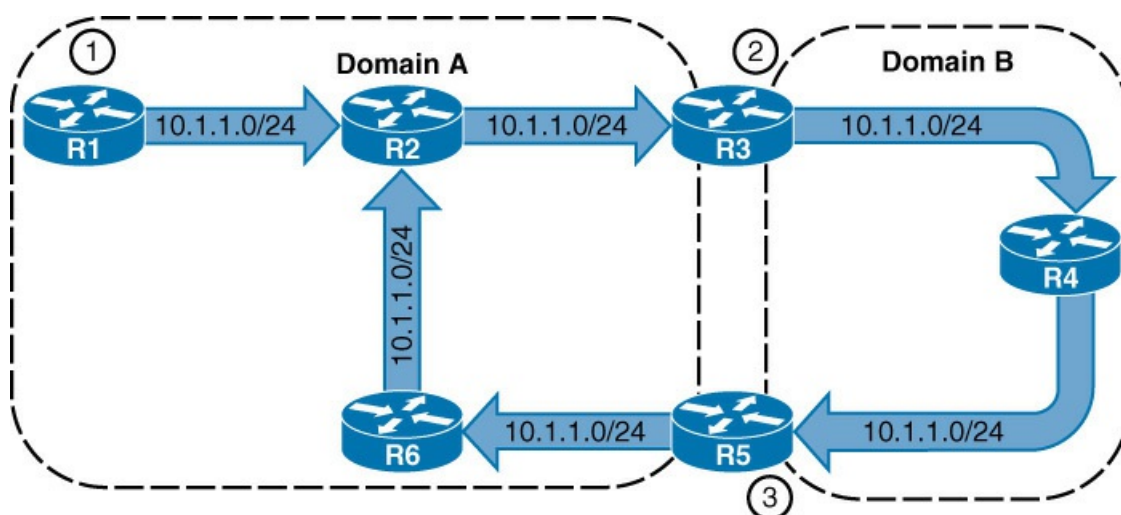


Figure 13-22 Concept of Invalid Routing Tables

Figure 13-23 revisits the previous topology where XR3 is redistributing the 10.1.1.0/24 back into EIGRP. If the 10.1.1.0/24 network fails, the route still remains in the routing table.





path metric for the 10.1.1.0/24 network redistributed by XR3 into EIGRP was significantly lower from the advertisement from R1. R2 receives both advertisements from R1 (27,136) and XR3 (3072) and identifies XR3's route as more preferable.

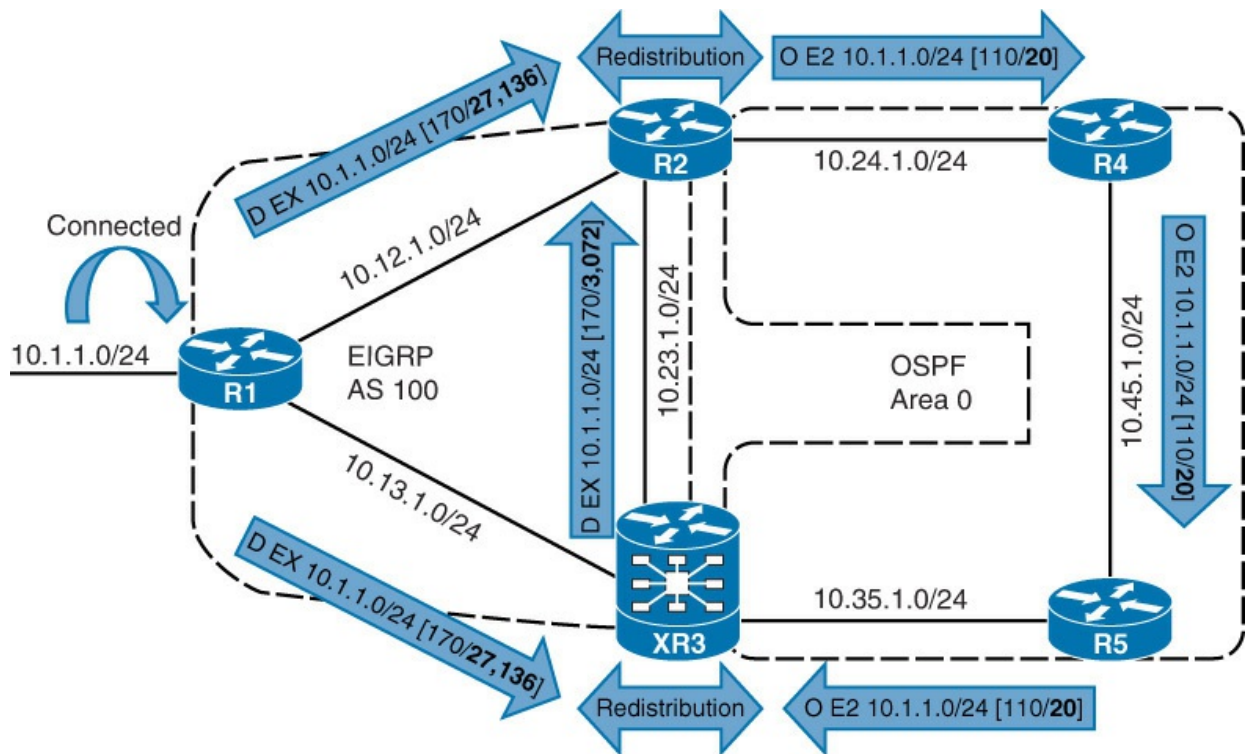


Figure 13-24 Routing Loop

Example 13-37 provides the routing table for R2, XR3, R4, and R5. Notice the next-hop addresses for the 10.1.1.0/24 on R2 points toward XR3, XR3 points toward R5, and R4 points toward R2 completing the routing loop.

### Example 13-37 R2's, XR3's, R4's, and R5's Routing Table

[Click here to view code image](#)

```
R2#show ip route
! Output omitted for brevity

D EX    10.1.1.0/24 [170/3072] via 10.23.1.3, 00:12:06, GigabitEthernet0/2
```

```
RP/0/0/CPU0:XR3#show route
! Output omitted for brevity

O E2 10.1.1.0/24 [110/20] via 10.35.1.5, 00:02:51, GigabitEthernet0/0/0/3
```

```
R5#show ip route
! Output omitted for brevity

O E2    10.1.1.0/24 [110/20] via 10.45.1.4, 00:15:13, GigabitEthernet0/2
```

```
R4#show ip route
! Output omitted for brevity

O E2    10.1.1.0/24 [110/20] via 10.24.1.2, 00:14:22, GigabitEthernet0/1
```

Traffic toward the 10.1.1.0/24 network will fall into the R2 --> XR3 --> R5 --> R4 --> R2 routing loop. [Example 13-38](#) provides verification with XR3's traceroute to 10.1.1.1.

### Example 13-38 XR4 Traceroute to 10.1.1.1

[Click here to view code image](#)

```
RP/0/0/CPU0:XR3#traceroute 10.1.1.1
! Output omitted for brevity
Tracing the route to 10.1.1.1
  1 10.35.1.5 0 msec 1 msec 0 msec
  2 10.45.1.4 0 msec 0 msec 1 msec
  3 10.24.1.2 0 msec 0 msec 1 msec
  4 10.23.1.3 0 msec 0 msec 0 msec
  5 10.35.1.5 0 msec 1 msec 0 msec
  .
  .
 29 10.35.1.5 1 msec 0 msec 1 msec
 30 10.45.1.4 1 msec 1 msec 1 msec
```

[Example 13-39](#) displays R2's EIGRP topology for the 10.1.1.0/24 network. Notice that XR3's path to the 10.1.1.0/24 network is preferred over R1's, which is causing the routing loop.

### Example 13-39 EIGRP Topology Table with Routing Loop

[Click here to view code image](#)

```
R2#show ip eigrp topology 10.1.1.0/24
! Output omitted for brevity

EIGRP-IPv4 Topology Entry for AS(100)/ID(192.168.2.2) for 10.1.1.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 3072
  Descriptor Blocks:
  10.23.1.3 (GigabitEthernet0/2), from 10.23.1.3, Send flag is 0x0
    Composite metric is (3072/2816), route is External
    External data:
      Originating router is 192.168.3.3
      AS number of route is 100
      External protocol is OSPF, external metric is 20
      Administrator tag is 0 (0x00000000)
  10.12.1.1 (GigabitEthernet0/0), from 10.12.1.1, Send flag is 0x0
    Composite metric is (27136/26880), route is External
    Originating router is 192.168.1.1
    External data:
      AS number of route is 0
      External protocol is Connected, external metric is 0
      Administrator tag is 0 (0x00000000)
```





## Example 13-40 Redistribution with Prefix Filtering Configuration

[Click here to view code image](#)

```
R2
router eigrp 100
  default-metric 100000 1 255 1 1500
  network 10.12.0.0 0.0.255.255
  network 10.23.1.0 0.0.0.255
  network 10.24.0.0 0.0.255.255
  redistribute ospf 1 route-map OSPF2EIGRP
!
router ospf 1
  redistribute eigrp 100 subnets
  network 10.24.1.0 0.0.0.255 area 0
!
ip prefix-list TEN-ONE seq 5 permit 10.1.1.0/24
!
route-map OSPF2EIGRP deny 10
  match ip address prefix-list TEN-ONE
!
route-map OSPF2EIGRP permit 20
```

```
XR3
router eigrp 100
  address-family ipv4
  default-metric 1000000 1 255 1 1500
  redistribute ospf 1 route-policy OSPF2EIGRP
  interface GigabitEthernet0/0/0/2
  !
  interface GigabitEthernet0/0/0/4
!
router ospf 1
  redistribute eigrp 100
  area 0
  interface GigabitEthernet0/0/0/3
!
route-policy OSPF2EIGRP
  if not destination in (10.1.1.0/24) then
    pass
  endif
end-policy
```

Example 13-41 provides R2's routing table. The next hop for the 10.1.1.0/24 route is R1 (10.12.1.1) instead of the previous setting toward XR3. This ensures that R2's routing table is programmed properly.

## Example 13-41 R2 and XR3 Routing Table

[Click here to view code image](#)

```
R2#show ip route
! Output omitted for brevity

D EX    10.1.1.0/24 [170/ 27136] via 10.12.1.1, 00:06:58, GigabitEthernet0/0
```

**Example 13-42** provides the EIGRP topology table for the 10.1.1.0/24 network. Notice that the originating router is R1 (192.168.1.1) for both paths. The path from XR3 (192.168.3.3) was prevented from being redistributed back into EIGRP from OSPF.

### Example 13-42 EIGRP Topology Table After Prefix Filtering

[Click here to view code image](#)

```
R2#show ip eigrp topology 10.1.1.0/24
! Output omitted for brevity

EIGRP-IPv4 Topology Entry for AS(100)/ID(192.168.2.2) for 10.1.1.0/24
 10.12.1.1 (GigabitEthernet0/0), from 10.12.1.1, Send flag is 0x0
   Composite metric is (27136/26880), route is External
   Originating router is 192.168.1.1
 10.23.1.3 (GigabitEthernet0/2), from 10.23.1.3, Send flag is 0x0
   Composite metric is (27392/27136), route is External
   Originating router is 192.168.1.1
```

### Tagging

The concept of route tags was briefly introduced in Chapter 11, “Route Maps and Route Policy,” for conditional matching. A route tag is a 32-bit number (0–4294967295) associated with a network prefix that allows for identification by other routers in the routing domain. The route tag is visible when looking at a specific entry in the routing protocols database or topology table.

A common concept is to use a unique identifier for the router (that is, router number) combined with the last three digits to identify the source routing protocol. Table 13-5 provides sample router identifiers, route source protocols, and suggested route tag using the previous logic.

Router Identifier	Source Protocol	Suggested Route Tag
R1 (1)	OSPF (AD 110)	1110
R1 (1)	BGP (AD 20)	1020
R2 (2)	IS-IS (AD)	2115

**Table 13-5** Sample Route Tag Chart

In Figure 13-26, R2 is setting the route tag of 2170 (R2, external EIGRP) for routes as they advertise into OSPF, and XR3 is setting the route tag of 3170 (XR3, external EIGRP) for routes as they advertise into OSPF. R2 and XR3 can selectively discard routes with the matching tags, as they redistribute from OSPF into EIGRP.

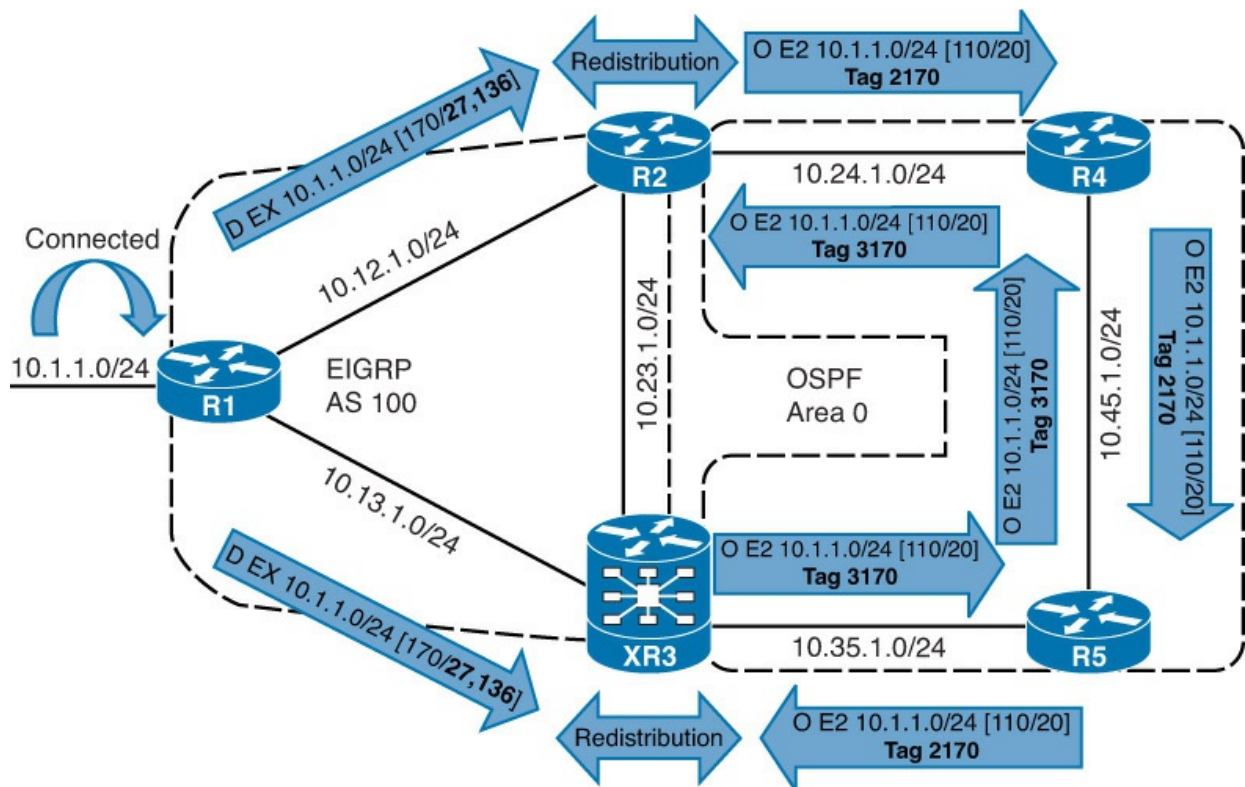


Figure 13-26 Routing Loop Prevention Using Tags

Note

BGP does not support route tags. IS-IS requires wide metrics because the tag is stored in the Extended IP Reachability TLV 135 as a sub-TLV.

Example 13-43 provides the configuration for R2 and XR3. Redistributing into OSPF allows for the route tags to be set without the need of a route map or route policy. This approach sets the route tag for all routes that are redistributed and will overwrite any existing route tags. R2 will not redistribute OSPF routes with the 3170 tag into EIGRP, and XR3 will not redistribute OSPF routes with the 2170 tag into EIGRP.

**Example 13-43** Route Tag Redistribution Configuration

[Click here to view code image](#)

**R2**

```
router eigrp 100
  default-metric 100000 1 255 1 1500
  network 10.12.0.0 0.0.255.255
  network 10.23.1.0 0.0.0.255
  network 10.24.0.0 0.0.255.255
  redistribute ospf 1 route-map OSPF2EIGRP
!
router ospf 1
  redistribute eigrp 100 subnets tag 2170
  network 10.24.1.0 0.0.0.255 area 0
!
route-map OSPF2EIGRP deny 10
  match tag 3170
route-map OSPF2EIGRP permit 20
```

**XR3**

```
router eigrp 100
  address-family ipv4
  default-metric 1000000 1 255 1 1500
  redistribute ospf 1 route-policy OSPF2EIGRP
  interface GigabitEthernet0/0/0/2
  !
  interface GigabitEthernet0/0/0/4
  !
router ospf 1
  redistribute eigrp 100 tag 3170
  area 0
  interface GigabitEthernet0/0/0/3
  !
route-policy OSPF2EIGRP
  if not tag is 2170 then
    pass
  endif
  drop
end-policy
```

**Note**

Setting route tags in EIGRP requires the use of a route map and route policy and was not shown in the configuration for the sake of brevity.

**Example 13-44** displays the LSDB on R2 where the route tag can be seen for both of the link-state advertisements (LSAs) redistributed on R2 and XR3.

**Example 13-44** *OSPF External LSAs for 10.1.1.0/24*

[Click here to view code image](#)

```

R2#show ip ospf database external
! Output omitted for brevity

OSPF Router with ID (192.168.2.2) (Process ID 1)

Type-5 AS External Link States

LS Type: AS External Link
Link State ID: 10.1.1.0 (External Network Number )
Advertising Router: 192.168.2.2
Network Mask: /24
    External Route Tag: 2170

LS Type: AS External Link
Link State ID: 10.1.1.0 (External Network Number )
Advertising Router: 192.168.3.3
Network Mask: /24
    External Route Tag: 3170

```

**Example 13-45** confirms that the EIGRP topology only contains entries for the 10.1.1.0/24 network that are advertised from R1. This provides verification that the routing loop was removed.

### **Example 13-45** *EIGRP Topology Table*

[Click here to view code image](#)

```

R2#show ip eigrp topology 10.1.1.0/24
! Output omitted for brevity

EIGRP-IPv4 Topology Entry for AS(100)/ID(192.168.2.2) for 10.1.1.0/24
  10.12.1.1 (GigabitEthernet0/0), from 10.12.1.1, Send flag is 0x0
    Composite metric is (27136/26880), route is External
    Originating router is 192.168.1.1
  10.23.1.3 (GigabitEthernet0/2), from 10.23.1.3, Send flag is 0x0
    Composite metric is (27392/27136), route is External
    Originating router is 192.168.1.1

```

### **Increase Seed Metrics**

The seed metric provides a reference value to accommodate for a portion of the topology that was lost due to route redistribution. Elevating the seed metric makes a path less preferred when compared to an identical route with a lower metric.

**Figure 13-27** illustrates two routing domains, where R3 and R5 mutually redistribute between each other. If only the path metric is used to identify the best path, increasing the seed metric on R3 and R5 to a number higher than the source of the route on R1 will ensure R2 always selects R1 as the source of the route.

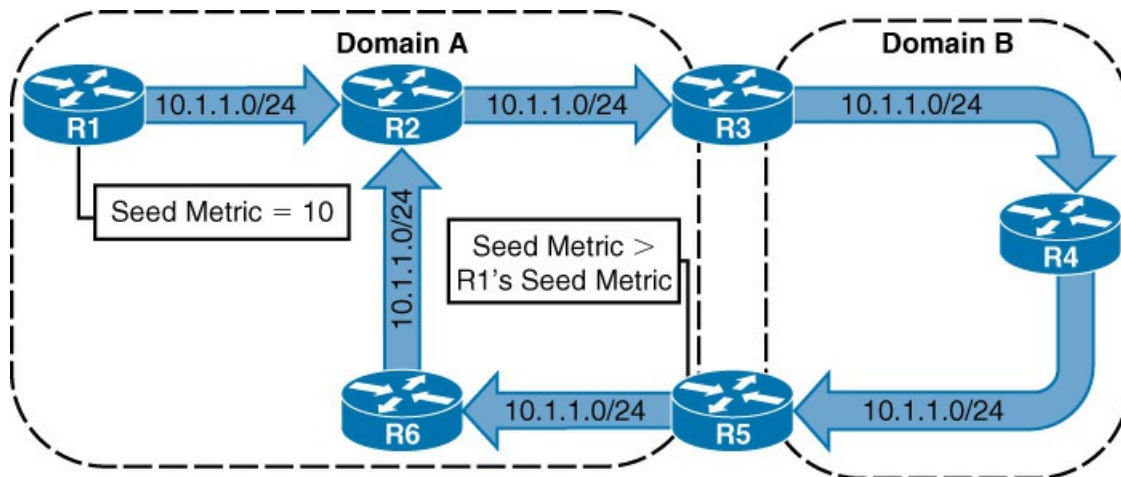


Figure 13-27 Optimal Usage of Seed Metrics

In Figure 13-28, R1's configuration uses a delay of 5 microseconds versus 1 microsecond (the normal value associated with at Gigabit Ethernet interface). This was done to simulate an increase in distance for this topology. The same concept can be applied during redistribution so that the seed metric is increased drastically making other native routes more preferred.

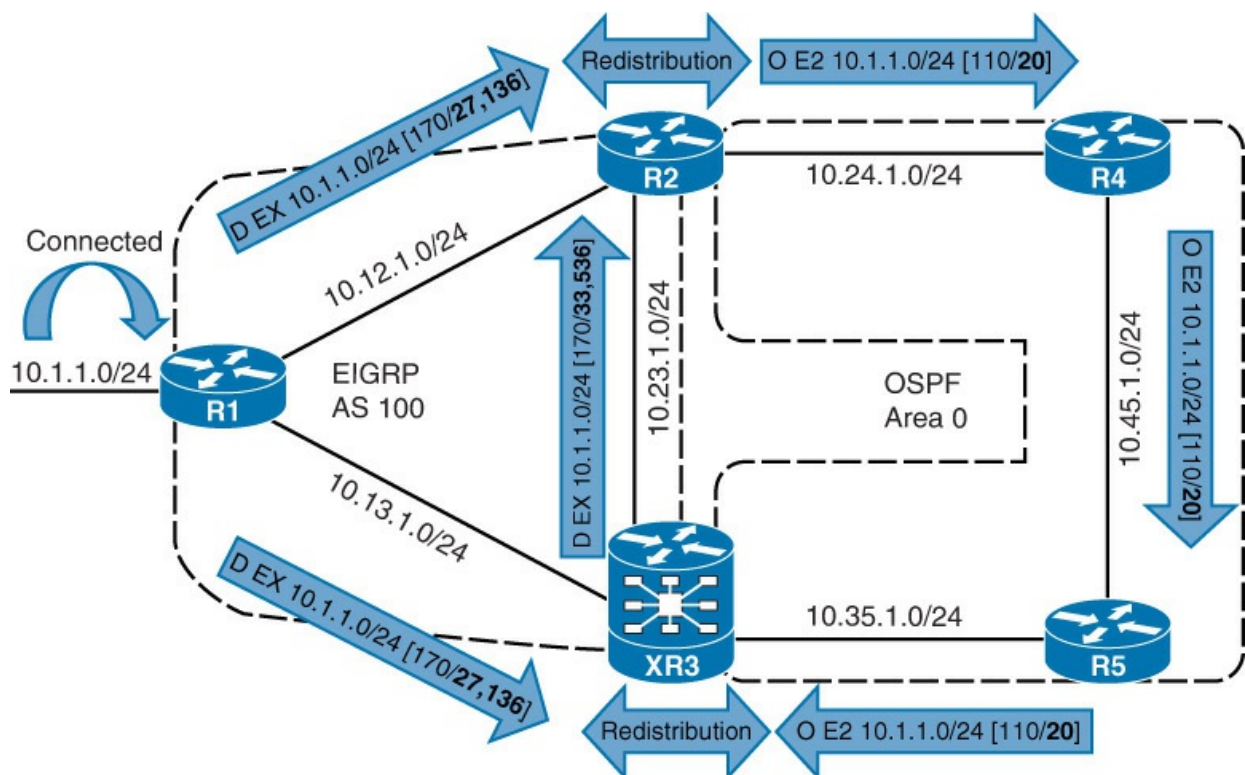


Figure 13-28 Increased Seed Metric Topology

The default seed metric is changed on R2 and XR3 so that the delay uses a value of 300 microseconds ( $\mu$ s) instead of 10 microseconds ( $\mu$ s). XR3 redistributes the OSPF 10.1.1.0/24 route into EIGRP with a path metric of 33,536, which is significantly higher than R1's path metric value of 27,136. R2 will always select the path advertised from R1 because it is a lower metric.

Example 13-46 provides R2 and XR3's configuration where the delay seed metric was incremented from 10 to 300 microseconds ( $\mu$ s) in the EIGRP default-metric. Remember that the delay K value is configured in tens of microseconds.

## Example 13-46 Increased Seed Metric Redistribution Configuration

[Click here to view code image](#)

### R2

```
router eigrp 100
  default-metric 1000000 30 255 1 1500
  network 10.12.0.0 0.0.255.255
  network 10.23.1.0 0.0.0.255
  network 10.24.0.0 0.0.255.255
  redistribute ospf 1
!
router ospf 1
  redistribute eigrp 100 subnets
  network 10.24.1.0 0.0.0.255 area 0
```

### XR3

```
router eigrp 100
  address-family ipv4
  default-metric 1000000 30 255 1 1500
  redistribute ospf 1
  interface GigabitEthernet0/0/0/2
  !
  interface GigabitEthernet0/0/0/4
  !
router ospf 1
  redistribute eigrp 100
  area 0
  interface GigabitEthernet0/0/0/3
```

**Example 13-47** confirms that the EIGRP topology contains only the original entry advertised by R1. The redistribute route from XR3 did not meet the feasible successor condition because the reported distance was higher than the feasible distance. XR3's route is not maintained in R2's EIGRP topology table.

## Example 13-47 EIGRP Topology Table

[Click here to view code image](#)

```
R2#show ip eigrp topology 10.1.1.0/24
! Output omitted for brevity
EIGRP-IPv4 Topology Entry for AS(100)/ID(10.24.1.2) for 10.1.1.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 27392
  10.23.1.3 (GigabitEthernet0/2), from 10.23.1.3, Send flag is 0x0
  Composite metric is (27392/27136), route is External
  External data:
    Originating router is 192.168.1.1
```

## Administrative Distance

Separating internal routes from external routes ensures that routes with a complete topology are preferred to the external routes that do not have the complete routing topology.



Remember that each protocol uses its own logic for identifying the best path it sends to the Routing Information Base (RIB), and the administrative distance (AD) is checked only when a conflict in the RIB exists. In most of the routing protocols' algorithms, an internal route is preferred to an external path; but if the AD does not change, there is not a differentiation between an internal route with one protocol and an external route from a different protocol. Only the AD differentiation between the two protocols is used.

EIGRP automatically sets the AD for external routes to 170, but other protocols do not automatically change the AD. Elevating the AD for external routes on other routing protocols ensures that internal routes are separated from external routes.

Instead of removing the OSPF route from the RIB with a distribute list, the AD could be modified so that external OSPF routes have a higher AD than external EIGRP routes. This technique ensures that the EIGRP installs into the RIB, but the OSPF route could install if both the EIGRP links fail.

Figure 13-29 demonstrates the topology where the OSPF routers have modified the AD for external OSPF routes to 180. On XR3, the 10.1.1.0/24 network uses EIGRP for installing the route into the RIB and eliminates the routing loop. A distribute list for the OSPF process is not required to ensure optimal routing.

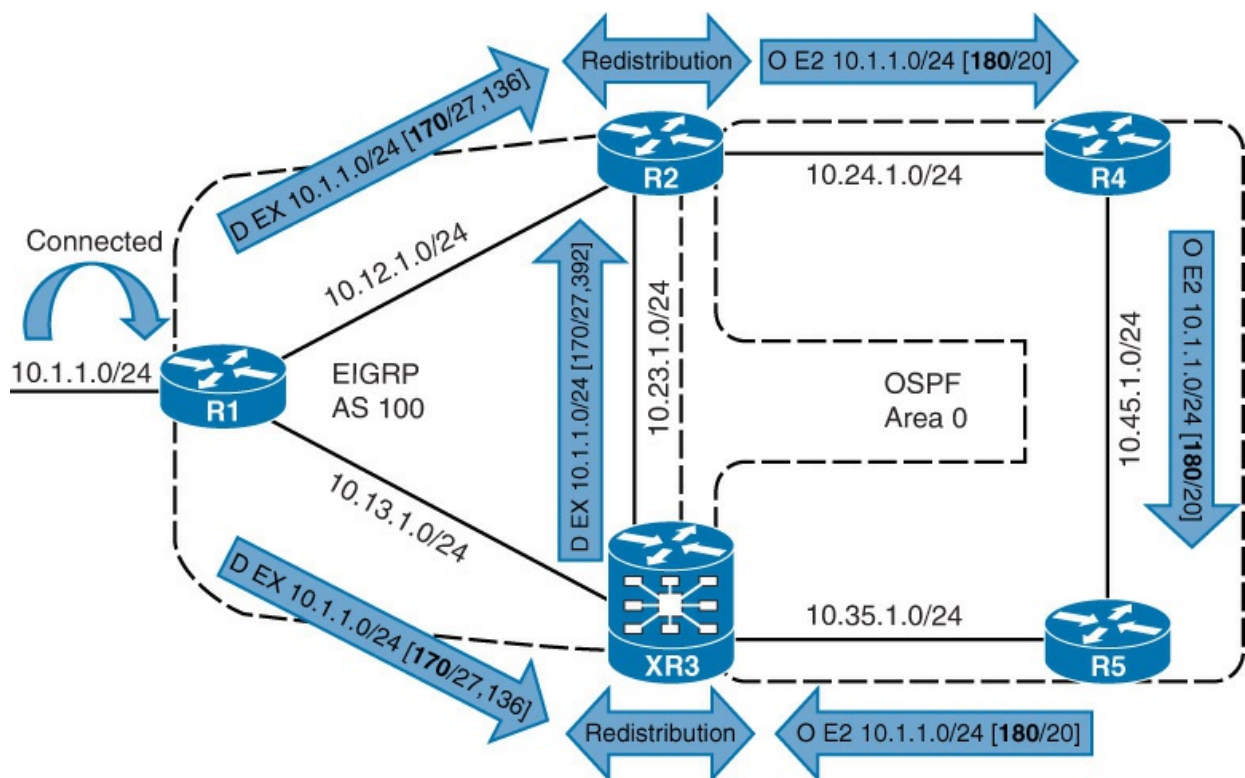


Figure 13-29 Administrative Distance Routing Loop Topology

Example 13-48 demonstrates the configuration for manipulating OSPF's external routes to 180. The AD change is not required on R4 or R5 for the function to work but was done for consistency purposes.

### Example 13-48 AD Manipulation for OSPF Configuration

[Click here to view code image](#)

```
R2
router eigrp 100
  default-metric 100000 1 255 1 1500
  network 10.12.0.0 0.0.255.255
  network 10.23.1.0 0.0.0.255
  network 10.24.0.0 0.0.255.255
  redistribute ospf 1
  eigrp router-id 192.168.2.2
!
router ospf 1
  redistribute eigrp 100 subnets
  network 10.24.1.0 0.0.0.255 area 0
  distance ospf external 180
```

```
XR3
router eigrp 100
  address-family ipv4
  default-metric 1000000 1 255 1 1500
  redistribute ospf 1
  interface GigabitEthernet0/0/0/2
  !
  interface GigabitEthernet0/0/0/4
  !
router ospf 1
  distance ospf external 180
  redistribute eigrp 100
  area 0
  interface GigabitEthernet0/0/0/3
```

**Example 13-49** provides the routing table for R2 and XR3. Notice that both routers use R1 as a next-hop address for this route.

### **Example 13-49** R2's and XR3's Routing Tables

[Click here to view code image](#)

```
R2#show ip route
! Output omitted for brevity

D EX    10.1.1.0/24 [170/ 27136] via 10.12.1.1, 00:06:58, GigabitEthernet0/0
```

```
RP/0/0/CPU0:XR3#show route
! Output omitted for brevity

D EX 10.1.1.0/24 [170/27136] via 10.13.1.1, 00:00:03, GigabitEthernet0/0/0/2
```

**Example 13-50** displays R4's OSPF routing table. Notice that the OSPF AD was modified to 180 to ensure that it is higher than EIGRP's external AD of 170.

## Example 13-50 R4's Routing Table

[Click here to view code image](#)

```
R4#show ip route ospf
! Output omitted for brevity

O E2 10.1.1.0/24 [180/20] via 10.24.1.2, 00:00:07, GigabitEthernet0/1
O E2 10.12.1.0/24 [180/20] via 10.24.1.2, 00:00:07, GigabitEthernet0/1
O E2 10.13.1.0/24 [180/20] via 10.24.1.2, 00:00:07, GigabitEthernet0/1
O E2 10.23.1.0/24 [180/20] via 10.24.1.2, 00:00:07, GigabitEthernet0/1
O 10.35.1.0/24 [110/2] via 10.45.1.5, 00:00:07, GigabitEthernet0/2
```

### Note

IS-IS does not allow for the modification of AD based on internal or external routes. This requires that the routes be explicitly identified, which is acceptable usage for all routing protocols, too.

### Summarization on Redistributing Router

Summarizing on the redistributing router minimizes the impact of route feedback. Connectivity is still maintained in the secondary routing domain through a less-specific prefix. When the summarized prefix redistributes into the original routing domain, routers will still use the more-specific routes for forwarding traffic.

Figure 13-30 demonstrates the same topology where R2 and XR3 are summarizing external routes in the 10.1.0.0/16 network range. As the 10.1.1.0/24 network redistributes into OSPF, the ASBRs advertise the 10.1.0.0/16 in lieu of the more-specific 10.1.1.0/24 network prefix. There is not a conflict for the 10.1.1.0/24 network protocol for installation in the RIB, and the 10.1.0.0/16 route is redistributed into EIGRP.

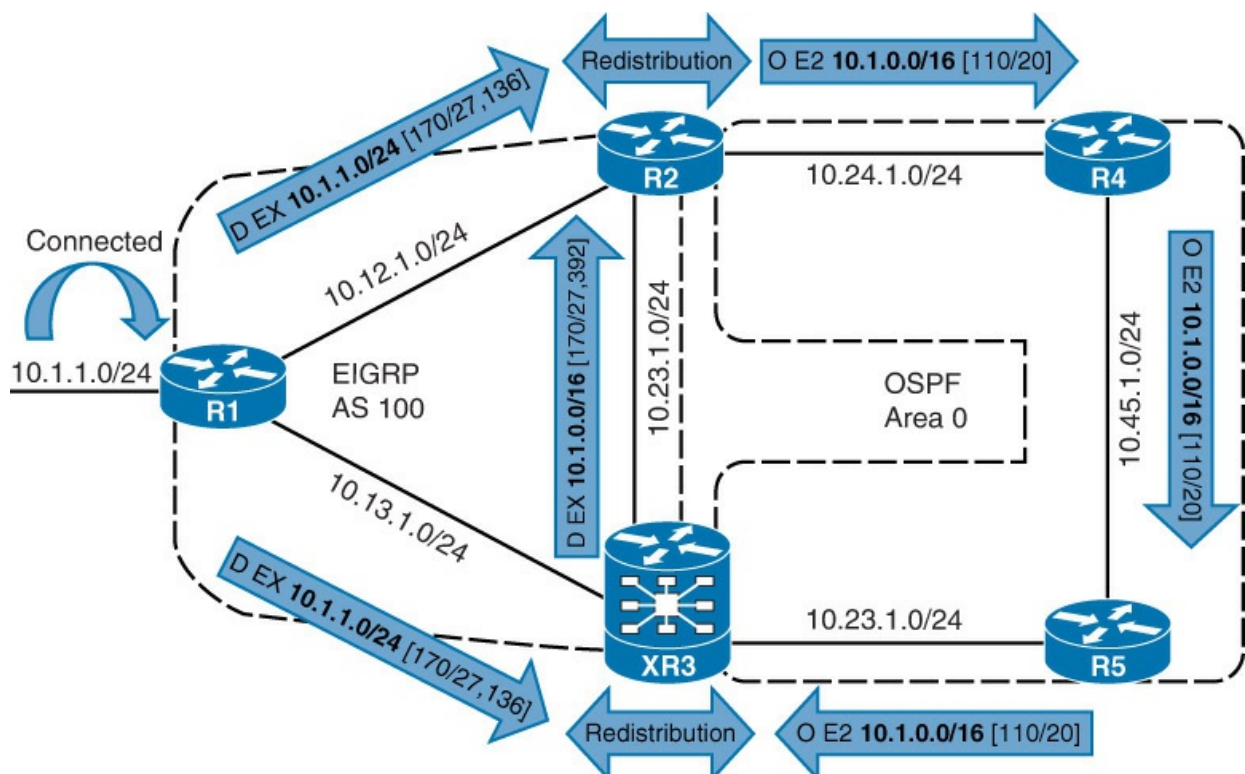


Figure 13-30 Redistribution with Summarization Topology

**Example 13-51** provides R2's and XR3's configuration with OSPF external summarization for the 10.1.0.0/16 network range.

### **Example 13-51** Summarization on Redistributing Router Configuration

[Click here to view code image](#)

```
R2
router eigrp 100
  default-metric 100000 1 255 1 1500
  network 10.12.0.0 0.0.255.255
  network 10.23.1.0 0.0.0.255
  network 10.24.0.0 0.0.255.255
  redistribute ospf 1
!
router ospf 1
  summary-address 10.1.0.0 255.255.0.0
  redistribute eigrp 100 subnets
  network 10.24.1.0 0.0.0.255 area 0
```

```
XR3
router eigrp 100
  address-family ipv4
  default-metric 1000000 1 255 1 1500
  redistribute ospf 1
  interface GigabitEthernet0/0/0/2
  !
  interface GigabitEthernet0/0/0/4
  !
router ospf 1
  router-id 192.168.3.3
  summary-prefix 10.1.0.0/16
  redistribute eigrp 100
  area 0
  interface GigabitEthernet0/0/0/3
```

**Example 13-52** displays R2 and XR3's routing table for the 10.1.1.0/24 and 10.1.0.0/16 networks. Even though the 10.1.0.0/16 network range is not optimal, the routers will use the 10.1.1.0/24 network entry because it is more specific.

### **Example 13-52** R2's and XR3's Route Table

[Click here to view code image](#)

```
R2#show ip route
! Output omitted for brevity

D EX    10.1.0.0/16 [170/3072] via 10.23.1.3, 00:02:32, Ethernet0/2
D EX    10.1.1.0/24 [170/27136] via 10.12.1.1, 00:02:32, Ethernet0/0
```

```
RP/0/0/CPU0:XR3#show route
! Output omitted for brevity
```

```
O E2 10.1.0.0/16 [254/0] via 0.0.0.0, 00:04:42, Null0
D EX 10.1.1.0/24 [170/27136] via 10.13.1.1, 00:05:28, GigabitEthernet0/0/0/2
```

**Example 13-53** displays R4's OSPF routing table. There is not an entry for the 10.1.1.0/24 network, but routers will use the 10.1.0.0/16 entry to forward traffic toward the 10.1.1.0/24 network.

### Example 13-53 R4's OSPF Routing Table

[Click here to view code image](#)

```
R4#show ip route ospf
! Output omitted for brevity
```

```
10.0.0.0/8 is variably subnetted, 9 subnets, 3 masks
O E2 10.1.0.0/16 [110/20] via 10.24.1.2, 00:05:54, Ethernet0/1
O E2 10.12.1.0/24 [110/20] via 10.24.1.2, 00:05:55, Ethernet0/1
O E2 10.13.1.0/24 [110/20] via 10.24.1.2, 00:05:54, Ethernet0/1
O E2 10.23.1.0/24 [110/20] via 10.24.1.2, 00:05:55, Ethernet0/1
10.35.1.0/24 [110/2] via 10.45.1.5, 00:03:04, Ethernet0/2
```

### Solutions to Redistribution Challenges

Networks that mutually redistribute between routing protocols on multiple routers can introduce problems through route feedback. Routing loops, invalid routing tables, and suboptimal routing are detriments to the health of a network, but the redundancy eliminates single points of failure, too. The following guidelines help eliminate issues caused by route feedback caused because of mutual route redistribution on multiple routers:

- Document the physical and logical topology. Include the routing protocols and desired traffic flows.
- Avoid overlapping routing protocols on the same networks links or between multiple continuous router segments. A clear delineation between routing protocols simplifies troubleshooting.
- Routers should differentiate and prefer internal routes over external routes where feasible. Modify the AD for the routing protocols so that external routes have a higher AD than internal routes.
- Increase the seed metrics at redistribution to ensure that the source is preferred over redistributing routers.
- Use distribute lists where feasible while allowing redundancy in connectivity.
- Summarize the external routes on the redistributing router.

### SUMMARY

Redistribution is the method of injecting routes from one routing protocol (or domain) into another routing protocol.

Redistribution uses the following rules:

- Redistribution is not transitive.
- Only routes installed in the RIB are eligible for redistribution.
- Redistributed routes from a source protocol with a higher AD will not preempt an existing route from a routing protocol with a lower AD.
- If a metric is not defined, the seed metric is used.

Every routing protocol has a unique behavior during route redistribution. [Table 13-6](#) identifies the source selection criteria for a protocol and the default seed metric.

Protocol	Source Selection	Seed Metric
Static	Any static route that installs into the RIB. * This protocol can only be a source protocol.	—
Connected	Any interface in an up state that is not associated with the destination protocol. * This protocol can only be a source protocol.	—
EIGRP	Any routes in EIGRP including EIGRP-enabled connected networks.	Infinity
OSPF	Any routes in the OSPF LSDB, including OSPF-enabled interfaces.	1 for BGP, 20 for all other protocols
IS-IS	Any routes in the IS-IS L2 LSPDB, but does not include IS-IS-enabled interfaces.	0
BGP	Any routes in the BGP Loc-RIB table learned externally. iBGP routes require the command <b>redistribute internal</b> for redistribution into IGP routing protocols.	Origin set to incomplete

**Table 13-6** Source Protocol Chart Summary

Highly available network designs use multiple points of redistribution to ensure redundancy, which increases the probability of route feedback. Route feedback can cause suboptimal routing or routing loops, but can be resolved with the techniques explained in [Chapter 12](#) and this chapter.

## REFERENCES IN THIS CHAPTER

Moy, John. RFC 2328, *OSPF Version 2*, IETF, <http://www.ietf.org/rfc/rfc2328.txt>, April 1998

Cisco. Cisco IOS Software Configuration Guides. <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides. <http://www.cisco.com>

## **Part IV: Advanced BGP**

## Chapter 14. Advanced BGP

This chapter covers the following topics:

- [BGP communities](#)
- [Route summarization](#)
- Default routes
- Prefix manipulation and acceptance
- Configuration scaling

Border Gateway Protocol (BGP) can support hundreds of thousands of routes making it the ideal choice for the Internet. Organizations also use BGP for its flexibility and traffic engineering properties. This chapter expands upon the first BGP chapter, [Chapter 10, “Border Gateway Protocol \(BGP\)”](#), and explains BGP’s advanced features and concepts involved with the BGP routing protocol such as BGP communities, route summarization, outbound route filtering (ORF), and configuration scalability.

### BGP COMMUNITIES

BGP communities provide additional capability for tagging routes and for modifying BGP routing policy on upstream and downstream routers. BGP communities can be appended, removed, or modified selectively on each attribute, as the route travels from router to router.

BGP communities are an optional transitive BGP attribute that can traverse from autonomous system to autonomous system. A BGP community is a 32-bit number that can be included with a route. A BGP community can be displayed as a full 16-bit number (0–4,294,967,295) or as two 16-bit numbers (0–65535):(0–65535), commonly referred to as *new format*.

Note

BGP path attribute classifications are explained in [Chapter 10](#).

Private BGP communities follow the convention that the first 16-bits represent the autonomous system of the community origination, and the second 16-bits represent a pattern defined by the originating autonomous system. The private BGP community pattern could vary from organization to organization, do not need to be registered, and could signify geographic locations for one autonomous system while signifying a method of route advertisement in another autonomous system. Some organizations publish their private BGP community patterns on websites such as <http://www.onesc.net/communities/>.

In 2006, RFC 4360 expanded BGP communities’ capabilities by providing an extended format. Extended BGP communities provide structure for various classes of information, and are commonly used for Multiprotocol Label Switching (MPLS) virtual private network (VPN)



services.

IOS XR displays BGP communities in new format by default, and IOS routers display communities in decimal format by default. IOS routers can display communities in new-format with the global configuration command **ip bgp-community new-format**.

[Example 14-1](#) displays the BGP community in decimal format on top and in new format on bottom.

### Example 14-1 BGP Community Formats

[Click here to view code image](#)

```
! DECIMAL FORMAT
R3#show bgp 192.168.1.1
! Output omitted for brevity
BGP routing table entry for 192.168.1.1/32, version 6
Community: 6553602 6577023
```

```
! New Format
R3#show bgp 192.168.1.1
! Output omitted for brevity
BGP routing table entry for 192.168.1.1/32, version 6
Community: 100:2 100:23423
```

### Enabling BGP Community Support

IOS does not advertise BGP communities to peers by default. Communities are enabled on a neighbor-by-neighbor basis with the BGP address family configuration command **neighbor ip-address send-community [standard | extended | both]**. Standard communities are sent by default, unless the optional **extended** or **both** keywords are used.

IOS XR advertises BGP communities to interior BGP (iBGP) peers by default, but external BGP (eBGP) peers require the neighbor address family configuration command **send-community-ebgp** for advertising standard BGP communities, and the command **send-extended-community-ebgp** to advertise extended BGP communities. Both commands are required if both community formats are to be sent to an eBGP peer.

### Well-Known Communities

RFC 1997 defined a set of global communities (known as *well-known communities*) that use the community range of 4,294,901,760 (0xFFFF0000) to 4,294,967,295 (0xFFFFFFFF). All routers that are capable of sending/receiving BGP communities must implement well-known communities. Currently there are only four:

- Internet
- No\_Export
- No\_Advertise
- No\_Export\_SubConfed

## Internet

This community is a standardized community for identifying that a route should be advertised on the Internet. In larger networks that deploy BGP into the core, advertised routes should be advertised to the Internet should have this community set. Using this technique allows for the edge BGP routers to selectively advertise BGP routes with the Internet community. Filtering is not automatic, but can be done with an outbound route maps or route policies.

## No\_Export

The No\_Export community (0xFFFFF01 or 4,294,967,041) specifies that when a route is received with this community that the route is not advertised to any eBGP peer. If the router receiving the No\_Export route is a confederation member, the route can be advertised to other sub autonomous systems in the confederation.

The No\_Export community is set with the command **set community no-export** within the route map on IOS routers. IOS XR routers set the community inline in the route policy with the command **set community (no-export)**, or with the command **no-export** in a community set.

Figure 14-1 demonstrates the No\_Export community being set on routes as they enter the BGP confederation (AS200):

- R1 is advertising the 192.168.1.0/24 network, and R6 is advertising the 172.16.1.0/24 network.
- XR2 and R5 set the No\_Export community for routes learned from AS100 and AS300, respectively.
- AS100 and AS300 routes are visible to all sub-autonomous systems (member autonomous systems) in the confederation, but AS100 routes are not advertised to AS300, and AS300 routes are not advertised to AS100.

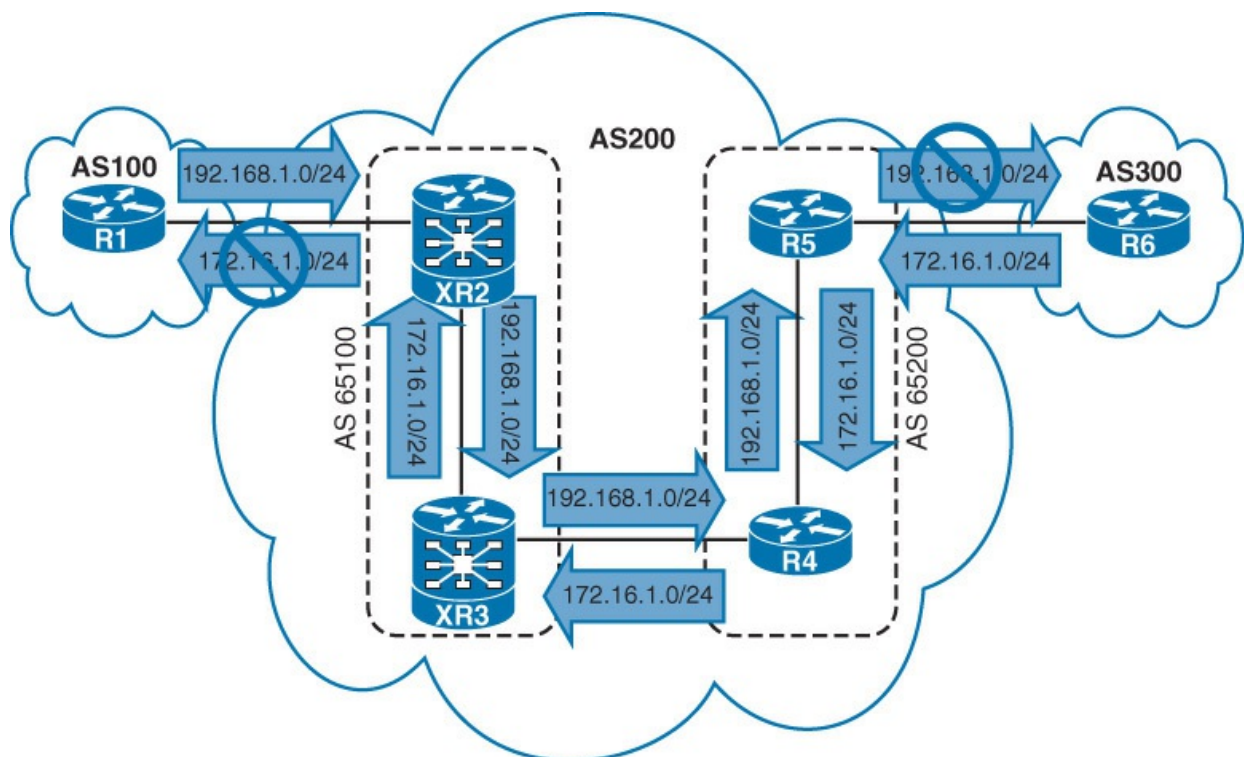


Figure 14-1 BGP No\_Export Community Topology

Example 14-2 demonstrates setting the No\_Export community on XR2 and R5 so that routes

from AS100 and AS300 are not advertised to the neighbor autonomous systems.

### Example 14-2 Configuration for Setting No\_Export BGP Community

[Click here to view code image](#)

```
XR2
route-policy AS100
  set community (no-export)
end-policy
!
router bgp 65100
  bgp confederation identifier 200
  address-family ipv4 unicast
    network 192.168.2.2/32
  !
  neighbor 10.12.1.1
    remote-as 100
    address-family ipv4 unicast
      send-community ebgp
      route-policy AS100 in
      route-policy PASSALL out
  !
  neighbor 192.168.3.3
    remote-as 65100
    update-source Loopback0
    address-family ipv4 unicast
      next-hop-self
```

```
R5
router bgp 65200
  bgp confederation identifier 200
  no bgp default ipv4-unicast
  neighbor 10.56.1.6 remote-as 300
  neighbor 192.168.4.4 remote-as 65200
  neighbor 192.168.4.4 update-source Loopback0
  !
  address-family ipv4
    neighbor 10.56.1.6 activate
    neighbor 10.56.1.6 send-community
    neighbor 10.56.1.6 route-map AS300 in
    neighbor 192.168.4.4 activate
    neighbor 192.168.4.4 send-community
    neighbor 192.168.4.4 next-hop-self
    network 192.168.5.5 mask 255.255.255.255
  exit-address-family

route-map AS300 permit 10
  set community no-export
```

Example 14-3 displays the BGP path attributes (PAs) for the 172.16.1.0/24 and 192.168.1.0/24 networks. Notice that the XR2 and R5 routes display “not advertised to eBGP peer.”

### Example 14-3 BGP Attributes for No\_Export Routes

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show bgp ipv4 unicast 172.16.1.0
! Output omitted for brevity
BGP routing table entry for 172.16.1.0/24
Paths: (1 available, best #1, not advertised to EBGp peer)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  (65200) 300
    192.168.5.5 (metric 13) from 192.168.3.3 (192.168.3.3)
    Community: no-export
```

```
R5#show bgp ipv4 unicast 192.168.1.0
! Output omitted for brevity
Paths: (1 available, best #1, table default, not advertised to EBGp peer)
  Not advertised to any peer
  (65100) 100
    192.168.2.2 (metric 22) from 192.168.4.4 (192.168.4.4)
    Community: no-export
```

**Example 14-4** verifies that R5 did not advertise the 192.168.1.0/24 network to R6 and that XR2 did not advertise the 172.16.1.0/24 network to R1.

### Example 14-4 Verification of Advertised Routes

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show bgp ipv4 unicast neighbors 10.12.1.1 advertised-routes
Network          Next Hop        From           AS Path
192.168.2.2/32   10.12.1.2      Local          i
192.168.3.3/32   10.12.1.2      192.168.3.3   i
192.168.4.4/32   10.12.1.2      192.168.3.3   (65200) i
192.168.5.5/32   10.12.1.2      192.168.3.3   (65200) i

Processed 4 prefixes, 4 paths
```

```
R5#show bgp ipv4 unicast neighbors 10.56.1.6 advertised-routes
Network          Next Hop        Metric LocPrf Weight Path
r>i192.168.2.2/32 192.168.2.2     0      100     0 (65100) i
r>i192.168.3.3/32 10.34.1.3       0      100     0 (65100) i
r>i192.168.4.4/32 192.168.4.4     0      100     0 i
*> 192.168.5.5/32 0.0.0.0         0              32768 i

Total number of prefixes 4
```

### No\_Advertise

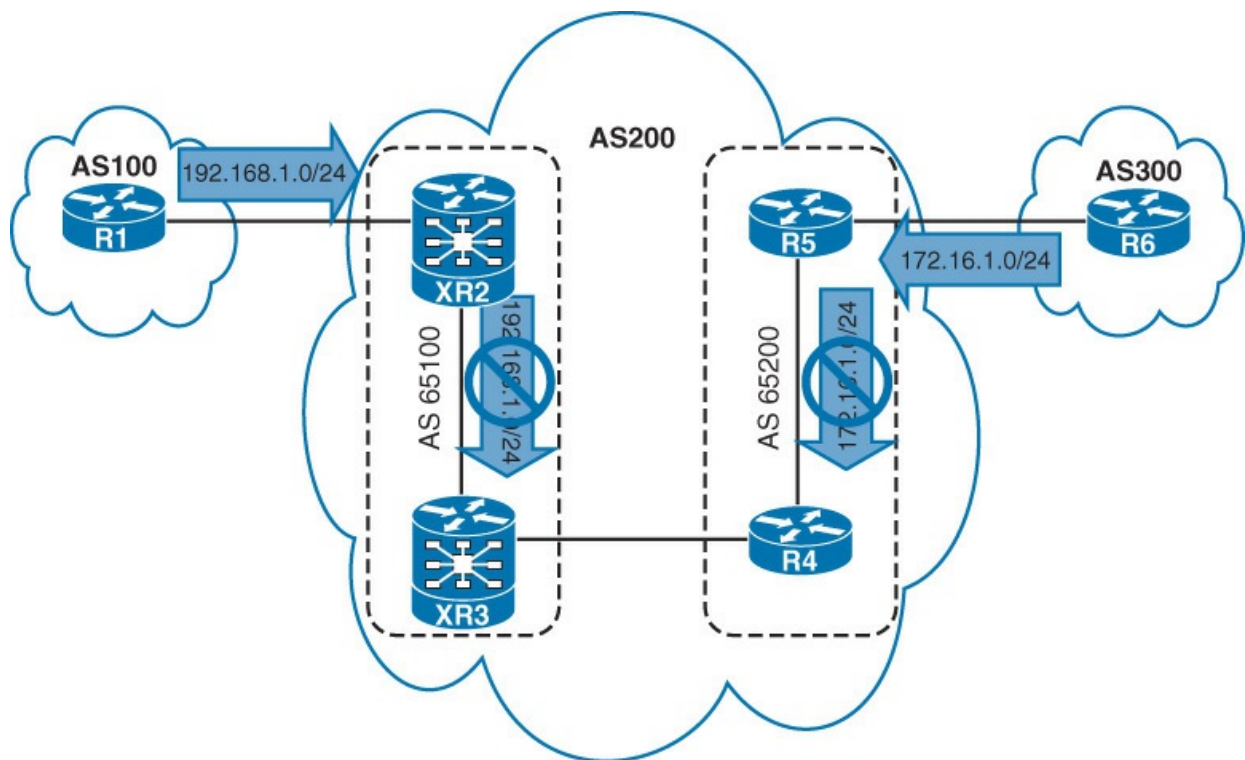
The No\_Advertise community (0xFFFFF02 or 4,294,967,042) specifies that routes with this community should not be advertised to any BGP peer.

The No\_Advertise community is set with the command **set community no-advertise** within

the route map on IOS routers. IOS XR routers set the community inline in the route policy with the command **set community (no-advertise)** or with the command **no-advertise** in a community set.

Figure 14-2 demonstrates the No\_Advertise community being set on routes from AS100 and AS300 as they enter the BGP confederation (AS200):

- R1 is advertising the 192.168.1.0/24 network into BGP, and R6 is advertising the 172.16.1.0/24 network into BGP.
- XR2 and R5 set the No\_Advertise BGP community for routes learned from AS100 or AS300 accordingly.
- The routes are in the Loc-RIB table only for XR2 and R5 but are not advertised to any BGP neighbor.



**Figure 14-2** BGP No\_Advertise Community Topology

Example 14-5 demonstrates setting the No\_Advertise community on XR2 and R5 so that routes are not advertised to any BGP peer.

#### **Example 14-5** Configuration Setting for No\_Advertise BGP Community

[Click here to view code image](#)

**XR2**

```
route-policy AS100
  set community (no-advertise)
end-policy

router bgp 65100
  bgp confederation identifier 200
  address-family ipv4 unicast
    network 192.168.2.2/32
  !
  neighbor 10.12.1.1
    remote-as 100
    address-family ipv4 unicast
      send-community-ebgp
      route-policy AS100 in
      route-policy PASSALL out
  !
  neighbor 192.168.3.3
    remote-as 65100
    update-source Loopback0
    address-family ipv4 unicast
      next-hop-self
```

**R5**

```
router bgp 65200
  bgp confederation identifier 200
  no bgp default ipv4-unicast
  neighbor 10.56.1.6 remote-as 300
  neighbor 192.168.4.4 remote-as 65200
  neighbor 192.168.4.4 update-source Loopback0
  !
  address-family ipv4
    neighbor 10.56.1.6 activate
    neighbor 10.56.1.6 send-community
    neighbor 10.56.1.6 route-map AS300 in
    neighbor 192.168.4.4 activate
    neighbor 192.168.4.4 send-community
    neighbor 192.168.4.4 next-hop-self
    network 192.168.5.5 mask 255.255.255.255
  exit-address-family

  route-map AS300 permit 10
    set community no-advertise
```

XR2 and R5 process the route with the No\_Advertise BGP community. [Example 14-6](#) verifies that the routes are set for “*not advertised to any peer.*”

**Example 14-6 BGP Attributes for No\_Advertise Routes**

[Click here to view code image](#)

```

RP/0/0/CPU0:XR2#show bgp ipv4 unicast 192.168.1.0
! Output omitted for brevity
BGP routing table entry for 192.168.1.0/24
Paths: (1 available, best #1, not advertised to any peer)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  100
    10.12.1.1 from 10.12.1.1 (192.168.1.1)
      Community: no-advertise

R5#show bgp ipv4 unicast 172.16.1.0
! Output omitted for brevity
BGP routing table entry for 172.16.1.0/24, version 12
Paths: (1 available, best #1, table default, not advertised to any peer)
  Not advertised to any peer
  300
    10.56.1.6 from 10.56.1.6 (172.16.1.1)
      Community: no-advertise

```

**Example 14-7** verifies that XR2 and R5 do not advertise the 192.168.1.0/24 and 172.16.1.0/24 network to other routers.

### Example 14-7 Verification of Advertised Routes

[Click here to view code image](#)

```

RP/0/0/CPU0:XR2#show bgp ipv4 unicast neighbor 192.168.3.3 advertised-routes
Network          Next Hop        From           AS Path
192.168.2.2/32   192.168.2.2    Local          i

Processed 1 prefixes, 1 paths

```

```

R5#show bgp ipv4 unicast neighbors 192.168.4.4 advertised-routes
Network          Next Hop        Metric LocPrf Weight Path
*> 192.168.5.5/32 0.0.0.0         0          32768 i

Total number of prefixes 1

```

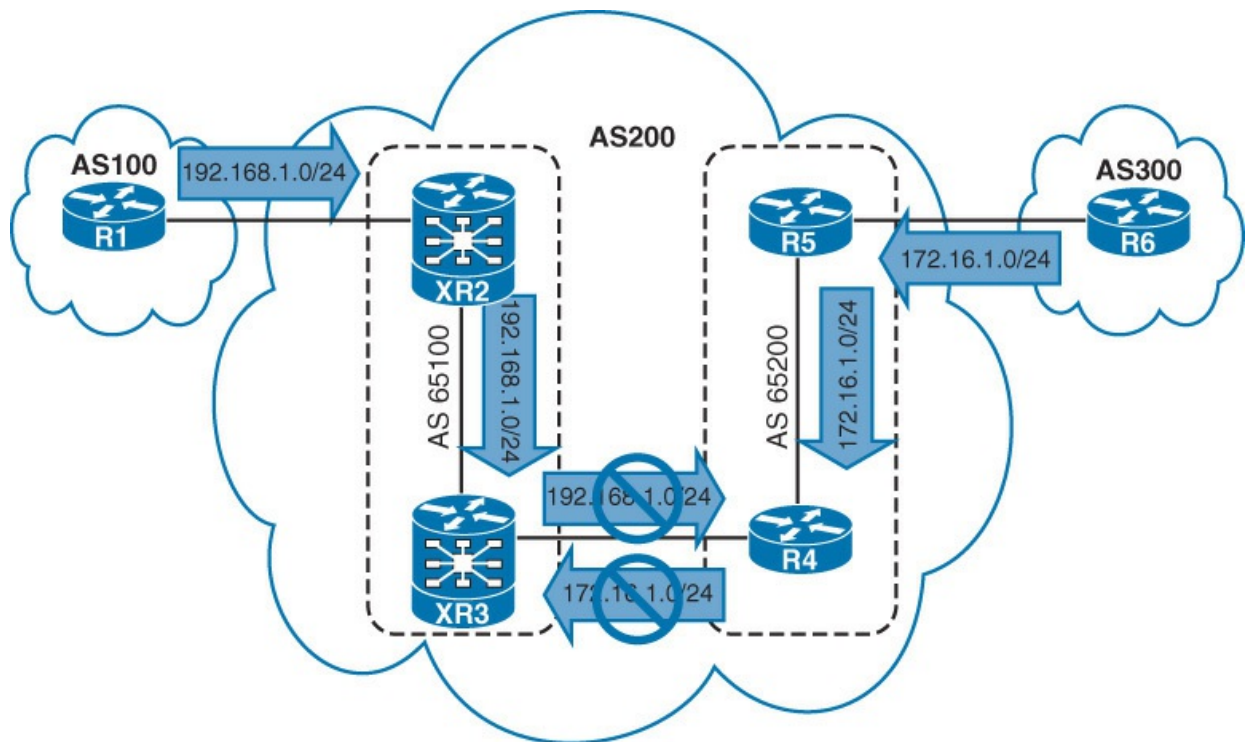
#### No\_Export\_SubConfed

The No\_Export\_SubConfed community (0xFFFFF03 or 4,294,967,043) known as the *Local-AS community* specifies that a route with this community is not advertised outside of the local autonomous system. If the router receiving a route with the Local-AS community is a confederation member, the route can be advertised only within the sub-autonomous system (member autonomous system) and is not advertised between member autonomous systems.

The Local-AS community is set with the command **set community local-as** within the route map on IOS routers. IOS XR routers set the community inline in the route policy with the command **set community (local-as)**, or with the command **local-as** in a community set.

**Figure 14-3** demonstrates the Local-AS community being set on routes from AS100 and AS300 as they enter the BGP confederation (AS200):

- R1 is advertising the 192.168.1.0/24 network and R6 is advertising the 172.16.1.0/24 network.
- XR2 and R5 set the Local-AS BGP community for routes learned from AS100 and AS300 accordingly.
- The routes are not exchanged between sub-autonomous systems (member autonomous systems).



**Figure 14-3** BGP Local-AS Community Topology

Example 14-8 demonstrates setting the Local-AS community on XR2 and R5 so that routes are not advertised to sub-autonomous systems (member autonomous systems).

### **Example 14-8** Configuration Setting for Local-AS BGP Community

[Click here to view code image](#)



**XR2**

```
route-policy AS100
  set community (local-AS)
end-policy
!
router bgp 65100
  bgp confederation identifier 200
  address-family ipv4 unicast
    network 192.168.2.2/32
  !
  neighbor 10.12.1.1
    remote-as 100
    address-family ipv4 unicast
      send-community-ebgp
      route-policy AS100 in
      route-policy PASSALL out
  !
  !
  neighbor 192.168.3.3
    remote-as 65100
    update-source Loopback0
    address-family ipv4 unicast
      next-hop-self
```

**R5**

```
router bgp 65200
  bgp confederation identifier 200
  no bgp default ipv4-unicast
  neighbor 10.56.1.6 remote-as 300
  neighbor 192.168.4.4 remote-as 65200
  neighbor 192.168.4.4 update-source Loopback0
  !
  address-family ipv4
    neighbor 10.56.1.6 activate
    neighbor 10.56.1.6 send-community
    neighbor 10.56.1.6 route-map AS300 in
    neighbor 192.168.4.4 activate
    neighbor 192.168.4.4 send-community
    neighbor 192.168.4.4 next-hop-self
    network 192.168.5.5 mask 255.255.255.255
  exit-address-family

  route-map AS300 permit 10
    set community local-AS
```

**Example 14-9** confirms that the routes are set for “*not advertised outside local AS*” and that the routes are not advertised to any peer.

**Example 14-9 BGP Attributes for Local-AS Routes**

[Click here to view code image](#)

```
RP/0/0/CPU0:XR3#show bgp ipv4 unicast 192.168.1.0
! Output omitted for brevity
BGP routing table entry for 192.168.1.0/24
Paths: (1 available, best #1, not advertised outside local AS)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  100
    192.168.2.2 (metric 2) from 192.168.2.2 (192.168.2.2)
      Community: local-AS
```

```
R4#show bgp ipv4 unicast 172.16.1.0
! Output omitted for brevity
BGP routing table entry for 172.16.1.0/24, version 16
Paths: (1 available, best #1, table default, not advertised outside local AS)
  Not advertised to any peer
  Refresh Epoch 3
  300
    192.168.5.5 (metric 11) from 192.168.5.5 (192.168.5.5)
      Community: local-AS
```

**Example 14-10** verifies that the edge router XR3 did not advertise the 192.168.1.0/24 network to R4 and that R4 did not advertise the 172.16.1.0/24 network to XR3.

### Example 14-10 Verification of Advertised Routes

[Click here to view code image](#)

```
RP/0/0/CPU0:XR3#show bgp ipv4 unicast neighbors 10.34.1.4 advertised-routes
Network          Next Hop          From              AS Path
192.168.2.2/32   192.168.2.2      192.168.2.2      i
192.168.3.3/32   10.34.1.3        Local             i

Processed 2 prefixes, 2 paths
```

```
R4#show bgp ipv4 unicast neighbors 10.34.1.3 advertised-routes
Network          Next Hop          Metric LocPrf Weight Path
*> 192.168.4.4/32 0.0.0.0           0      32768 i
r>i192.168.5.5/32 192.168.5.5      0      100    0 i

Total number of prefixes 2
```

#### Note

BGP confederations was used to demonstrate the various well-known communities, but well-known community usage is not restricted to BGP confederations.

## Conditionally Matching BGP Communities

Conditionally matching BGP communities allow for selection of routes based on the BGP communities within the route's path attributes so that selective processing can occur in IOS route map or IOS XR route policies as shown in [Chapter 11](#), “[Route Maps and Route Policy](#).” Conditionally matching on IOS routers requires the creation of a community list. A community list shares a similar structure to an ACL and can be standard or expanded and referenced via number or name. Standard community lists are numbered 1–99 and match either well-known communities or a private community number (*as-number:16-bit-number*). Expanded community lists are numbered 100–500, and use regular expression (regex) patterns.

The IOS configuration syntax for a community list is **ip community-list** {1-500 | **standard list-name** | **expanded list-name**} {**permit** | **deny**} *community-pattern*. When multiple communities are on the same **ip community list** statement, all communities for that statement must exist in the route. If only one out of many communities is required, then use multiple **ip community list** statements. After defining the community list, the community list is referenced in the route map with the command **match community 1-500**.

IOS XR route policies match BGP communities via two methods: inline and community sets.

### Community Set

IOS XR community sets are a component of IOS XR's routing protocol language (RPL) that contains a comma-delimited set of BGP communities. The structure is identical to a prefix set but contains different commands and content. Once a BGP community is matched, searching in the community set stops. Community set configuration uses the following steps:

#### Step 1. Define the community set.

Define the community set with the command **community-set community-set-name**.

#### Step 2. Enter the community set entries.

Configure each specific entry with one of the following commands:

■ *as-number:16-bit-number*

■ **local-AS**

■ **no-export**

■ **no-advertise**

Step 2 is repeated for multiple entries with every entry delimited with a comma except for the last one.

#### Step 3. Exit the community set.

Exit the community set with the command **end-set**.

[Example 14-11](#) provides a sample community set that matches the well-known Local-AS, 100:123, or 100:4569 communities.

### Example 14-11 Community Set Example Configuration

[Click here to view code image](#)

```
community-set SAMPLE-COMMUNITY
  local-AS,
  100:123,
  100:4569
end-set
```

#### Inline

Route policies supports the conditional matching of BGP communities with parameterization or explicitly stating the communities inline with these keywords:

- **is-empty:** Requires that there are no BGP communities associated with the route
- **matches-any:** Requires a route to match at least one of the BGP communities to qualify
- **matches-all:** Requires a route to match all BGP communities to qualify

Figure 14-4 provides a topology to demonstrate conditional matching and filtering with the use of BGP communities. The conditional matching is using multiple communities to demonstrate the Boolean **or** functionality to demonstrate the flexibility of BGP:

- XR1 advertises the 172.16.1.0/24 prefix with the No\_Export community and the 172.16.2.0/24 network with the private community 100:1.
- R4 advertises the 172.20.1.0/24 network with the No\_Export community and the 172.20.2.0/24 network with the private community 300:1.

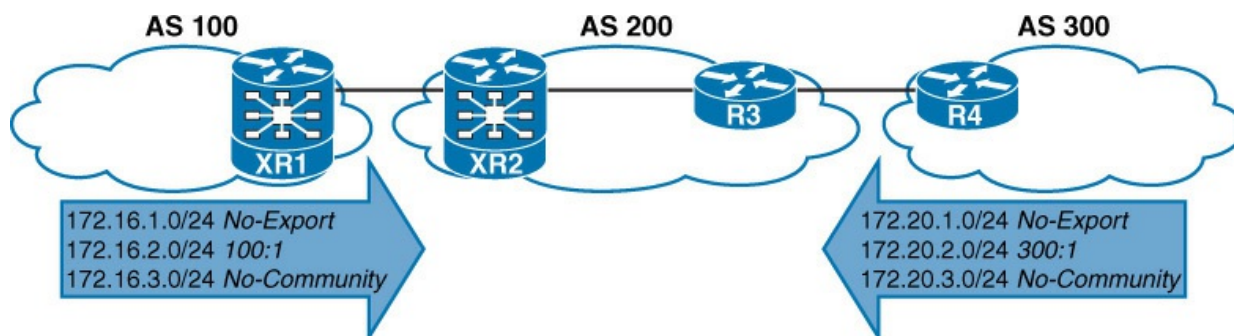


Figure 14-4 BGP Community Topology

Example 14-12 demonstrates the filtering of BGP private communities on XR2 and R3. XR2 blocks routes with the No\_Export or 100:1 private community, and R3 blocks routes with the No\_Export or 300:1 private community.

#### Example 14-12 BGP Community Configuration

[Click here to view code image](#)

**XR2 (Inline)**

```
route-policy COMMUNITY
  if community matches-any (100:1, no-export) then
    drop
  endif
  pass
end-policy
!
router bgp 200
  address-family ipv4 unicast
  !
  neighbor 10.12.1.1
    remote-as 100
    address-family ipv4 unicast
    send-community-ebgp
    route-policy COMMUNITY in
    route-policy PASSALL out
  soft-reconfiguration inbound
```

**XR2 (Community-Sets)**

```
route-policy COMMUNITY
  if community matches-any COMMUNITY then
    drop
  endif
  pass
end-policy
!
community-set COMMUNITY
  no-export,
  100:1
end-set
!
router bgp 200
  address-family ipv4 unicast
  !
  neighbor 10.12.1.1
    remote-as 100
    address-family ipv4 unicast
    send-community-ebgp
    route-policy COMMUNITY in
    route-policy PASSALL out
  soft-reconfiguration inbound
```

### R3

```
router bgp 200
  no bgp default ipv4-unicast
  neighbor 10.34.1.4 remote-as 300
  !
  address-family ipv4
    neighbor 10.34.1.4 activate
    neighbor 10.34.1.4 send-community
    neighbor 10.34.1.4 route-map COMMUNITIES in
  exit-address-family
  !
  route-map COMMUNITIES deny 10
  match community 1
  !
  route-map COMMUNITIES permit 20
  !
  ip bgp-community new-format
  ip community-list 1 permit 300:1
  ip community-list 1 permit no-export
```

[Example 14-13](#) displays XR2's Adj-RIB-In and Loc-RIB tables. Notice that 172.16.1.0/24 and 172.16.2.0/24 routes are filtered.

### Example 14-13 XR2's BGP Table

[Click here to view code image](#)

#### Adj-RIB Table

```
RP/0/0/CPU0:XR2#show bgp ipv4 unicast neighbors 10.12.1.1 received routes
```

```
! Output omitted for brevity
```

	Network	Next Hop	Metric	LocPrf	Weight	Path
*	172.16.1.0/24	10.12.1.1	0		0	100 i
*>	172.16.2.0/24	10.12.1.1	0		0	100 i
*	172.16.3.0/24	10.12.1.1	0		0	100 i

```
Processed 3 prefixes, 3 paths
```

#### Loc-RIB Table

```
RP/0/0/CPU0:XR2#show bgp ipv4 unicast neighbors 10.12.1.1 routes
```

```
! Output omitted for brevity
```

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	172.16.3.0/24	10.12.1.1	0		0	100 i

```
Processed 1 prefixes, 1 paths
```

[Example 14-14](#) displays R3's Adj-RIB-In and Loc-RIB tables. Notice that 172.20.1.0/24 and 172.20.2.0/24 routes are filtered.

### Example 14-14 R3's BGP Table

[Click here to view code image](#)

#### Adj-RIB Table

```
R3#show bgp ipv4 unicast neighbors 10.34.1.4 received-routes
```

```
! Output omitted for brevity
```

	Network	Next Hop	Metric	LocPrf	Weight	Path
*	172.20.1.0/24	10.34.1.4	0		0 300	i
*>	172.20.2.0/24	10.34.1.4	0		0 300	i
*	172.20.3.0/24	10.34.1.4	0		0 300	i

```
Total number of prefixes 3
```

#### Loc-RIB Table

```
R3#show bgp ipv4 unicast neighbors 10.34.1.4 routes
```

```
! Output omitted for brevity
```

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	172.20.3.0/24	10.34.1.4	0		0 300	i

```
Total number of prefixes 1
```

### Setting Private BGP Communities

A private BGP community is set in an IOS route map with the command **set community** *bgp-community* [**additive**]. IOS XR routers set the community in a route policy with the command **set community** *bgp-community* [**additive**]. By default, when setting a community, any existing communities will be overwritten but can be preserved by using the optional **additive** keyword.

Figure 14-5 demonstrates setting of BGP private communities:

- XR1 advertises the 172.16.1.0/24 network with the community of 100:1 and the 172.16.2.0/24 network with the community of 100:2.
- R3 advertises the 172.20.1.0/24 network with the community of 300:1 and the 172.20.2.0/24 network with the community 300:2.

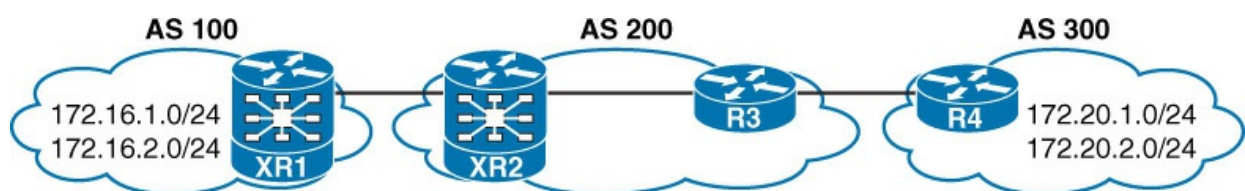


Figure 14-5 BGP Community Topology

Example 14-15 demonstrates the BGP configuration for XR2 and R3 in that

- XR2 sets the community 200:222 for routes with the existing community 100:1 and appends communities 200:250 200:2222 for routes matching the existing community 100:2.
- R3 sets the community 200:333 for routes with the existing community 300:1 and appends the communities 200:250 200:3333 for routes matching the existing community 300:2.

## Example 14-15 Setting BGP Community Configuration

[Click here to view code image](#)

### XR2

```
router bgp 200
  address-family ipv4 unicast
  !
  neighbor 10.12.1.1
    remote-as 100
    address-family ipv4 unicast
      route-policy COMMUNITY_ADDON in
      route-policy PASSALL out
    !
  !
  route-policy COMMUNITY_ADDON
    if community matches-any (100:1) then
      set community (200:150)
    endif
    if community matches-any (100:2) then
      set community (200:250, 200:2222) additive
    endif
    pass
  end-policy
```

### R3

```
router bgp 200
  no bgp default ipv4-unicast
  neighbor 10.34.1.4 remote-as 300
  !
  address-family ipv4
    neighbor 10.34.1.4 activate
    neighbor 10.34.1.4 send-community
    neighbor 10.34.1.4 route-map COMMUNITIES in
  exit-address-family
  !
  ip bgp-community new-format
  !
  ip community-list 1 permit 300:1
  ip community-list 2 permit 300:2
  !
  route-map COMMUNITIES permit 10
    match community 1
    set community 200:333
  !
  route-map COMMUNITIES permit 20
    match community 2
    set community 200:250 200:3333 additive
  !
  route-map COMMUNITIES permit 30
```

**Example 14-16** verifies that the original BGP communities for routes 172.16.1.0/24 and 172.20.1.0/24 were overwritten with the new BGP communities. Notice that the original communities 100:1 and 300:1 are not present in the community list.



## Example 14-16 BGP Community Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show bgp ipv4 unicast 172.16.1.0
! Output omitted for brevity
BGP routing table entry for 172.16.1.0/24
Paths: (1 available, best #1)
 100
 10.12.1.1 from 10.12.1.1 (192.168.1.1)
    Community: 200:222
```

```
R3#show bgp ipv4 unicast 172.20.1.0
BGP routing table entry for 172.20.1.0/24, version 2
Paths: (1 available, best #1, table default)
 300
 10.34.1.4 from 10.34.1.4 (192.168.4.4)
    Community: 200:333
```

Example 14-17 verifies that the original BGP communities for routes 172.16.2.0/24 and 172.20.2.0/24 were appended to the community list.

## Example 14-17 Additive BGP Community Verification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show bgp ipv4 unicast 172.16.2.0
 100
 10.12.1.1 from 10.12.1.1 (192.168.1.1)
    Received Path ID 0, Local Path ID 1, version 9
    Community: 100:2 200:250 200:2222
```

```
R3#show bgp ipv4 unicast 172.20.2.0
 300
 10.34.1.4 from 10.34.1.4 (192.168.4.4)
    Community: 300:2 200:250 200:3333
```

## ROUTE SUMMARIZATION

Summarizing prefixes conserves router resource and accelerates best path calculation by reducing the size of the table. Summarization also provides the benefit of stability by reducing routing churn and by hiding route flaps from downstream routers. While most service providers do not accept prefixes larger than /24 for IPv4 (/25 – /32), the Internet, at the time of this writing, still has more than 500,000 routes and continues to grow toward a million routes. Route summarization is required to reduce the size of the BGP table for Internet routers.

BGP route summarization on eBGP routers for nontransitive autonomous systems reduces route computation on routers in the core of the nontransitive autonomous system. In Figure 14-6, R3 summarizes all the eBGP routes received from AS100 and AS200 to reduce route computation on R4 during link flaps. In the event of a link flap on the 10.13.1.0/24 network, R3 will remove all AS100 routes learned directly from R1, and will identify the same networks via R2 with a different

(longer AS\_Path). R4 will process the same changes that R3 processes and is a waste of CPU cycles because R4 receives only connectivity from R3. If R3 summarized the network range, instead of running the best path algorithm against multiple routes, the best path algorithm would execute only once.

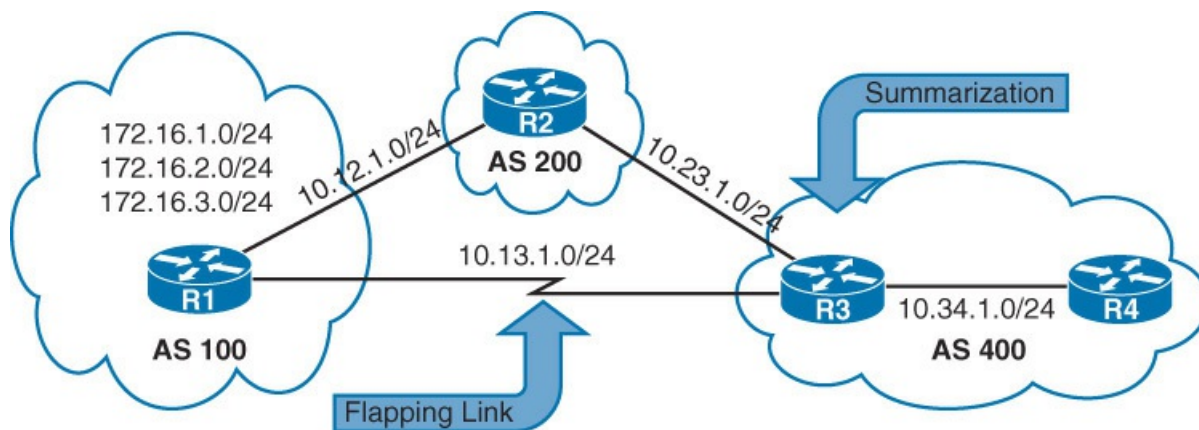


Figure 14-6 BGP Route Summarization

The two techniques for BGP summarization are as follows:

- **Static:** Create a static route to Null0 for the prefix, and then advertise the network via a network statement. The downfall to this technique is that the summary route will always be advertised even if the networks are not available.
- **Dynamic:** Configure an aggregation network range. When viable routes that match the network range enter the BGP table, an aggregate route is created. On the originating router, the aggregated prefix sets the next hop to Null0. The route to Null0 is automatically created by BGP as a loop-prevention mechanism.

In both methods, a new network prefix with a shorter prefix length is advertised into BGP. Because the aggregated prefix is a new route, the summarizing router is the originator for the new aggregate route.

### Aggregate Address

Dynamic route summarization is accomplished with the BGP address family configuration command **aggregate-address network subnet-mask [summary-only | supress-map route-map-name] [as-set] [advertise-map route-map-name]** for IOS routers and with the command **aggregate-address network/prefix-length [summary-only | route-policy route-policy-name] [as-set] [advertise-map route-policy-name]** on IOS XR routers. The aggregate-address command advertises the aggregated route in addition to the original networks. Using the optional **no-summary** keyword suppresses the networks in the summarized network range. BGP considers aggregated addresses as local routes.

Note

Aggregate addresses are local BGP routes when modifying BGP AD.

Figure 14-7 demonstrates BGP route summarization with selective route suppression:

- R1 is advertising four /24 networks. XR2 and R3 will summarize the networks to the 172.16.0.0/17 and 172.16.128.0/17 network ranges.

- XR2 will use the **summary-only** option for the 172.16.0.0/17 range, and R3 will use the **summary-only** option for the 172.16.128.0/17 range.

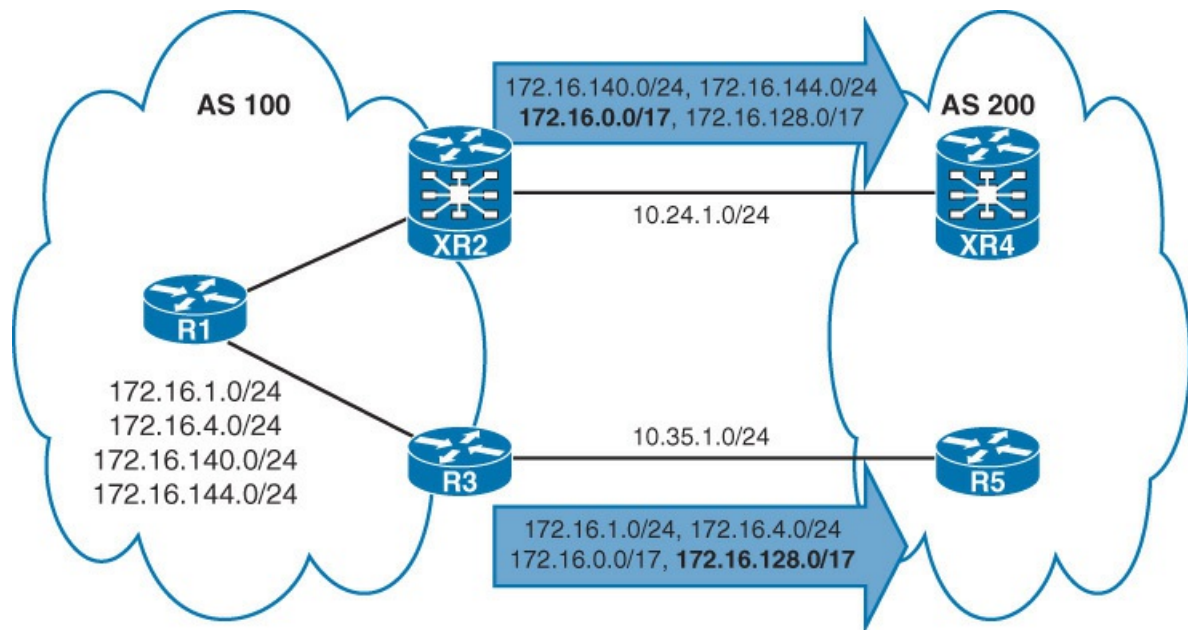


Figure 14-7 BGP Summarization Topology

Note

Figure 14-7 illustrates a common technique for load-balancing inbound traffic on multiple links. Smaller specific networks (for instance, /24) are advertised to only one eBGP peer, and a different set of smaller networks is advertised to a different eBGP peer. To ensure availability in the event of a link failure, a larger network range (/22) that encompasses all the smaller network ranges is advertised out of both links.

**Example 14-18** demonstrates the BGP configuration on XR2 and R3 for the BGP summarization. Notice that the **summary-only** keyword is applied to 172.16.0.0/17 range on XR2 and to 172.16.128.0/17 on R3.

### Example 14-18 BGP Summarization Configuration

[Click here to view code image](#)

**XR2**

```
router bgp 100
address-family ipv4 unicast
aggregate-address 172.16.0.0/17 summary-only
aggregate-address 172.16.128.0/17
```

**R3**

```
router bgp 100
no bgp default ipv4-unicast
neighbor 10.13.1.1 remote-as 100
neighbor 10.35.1.5 remote-as 200
!
address-family ipv4
aggregate-address 172.16.128.0 255.255.128.0 summary-only
aggregate-address 172.16.0.0 255.255.128.0
```

**Example 14-19** displays the BGP table and RIB for XR2 and R3. XR2 is suppressing the 172.16.1.0/24 and 172.16.4.0/24 routes because of the **summary-only** keyword, and R3 is suppressing the 172.16.140.0/24 and 172.16.144.0/24 routes because of the **summary-only** keyword. Notice that XR2 and R3 install a BGP route to Null0 in the Routing Information Base (RIB) for the summarized routes (172.16.0.0/17 and 172.16.128.0/17). A suppressed BGP route is not advertised to any BGP peer.

### Example 14-19 XR2 and R3 RIB and BGP Tables

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show bgp ipv4 unicast
! Output omitted for brevity

Status codes: s suppressed, d damped, h history, * valid, > best
               i - internal, r RIB-failure, S stale

   Network          Next Hop           Metric LocPrf Weight Path
*> 172.16.0.0/17    0.0.0.0                32768 i
s>i172.16.1.0/24    10.12.1.1              0     100     0 i
s>i172.16.4.0/24    10.12.1.1              0     100     0 i
*> 172.16.128.0/17 0.0.0.0                32768 i
*>i172.16.140.0/24 10.12.1.1              0     100     0 i
*>i172.16.144.0/24 10.12.1.1              0     100     0 i

Processed 6 prefixes, 6 paths

RP/0/0/CPU0:XR2#show route | i Null0
B    172.16.0.0/17 [200/0] via 0.0.0.0, 00:07:51, Null0
B    172.16.128.0/17 [200/0] via 0.0.0.0, 00:07:51, Null0

R3#show bgp ipv4 unicast
BGP table version is 9, local router ID is 10.35.1.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,

   Network          Next Hop           Metric LocPrf Weight Path
*> 172.16.0.0/17    0.0.0.0                32768 i
*>i172.16.1.0/24    10.13.1.1              0     100     0 i
*>i172.16.4.0/24    10.13.1.1              0     100     0 i
*> 172.16.128.0/17 0.0.0.0                32768 i
s>i172.16.140.0/24 10.13.1.1              0     100     0 i
s>i172.16.144.0/24 10.13.1.1              0     100     0 i

R3#show ip route | i Null0
B    172.16.0.0/17 [200/0] via 0.0.0.0, 00:10:15, Null0
B    172.16.128.0/17 [200/0] via 0.0.0.0, 00:10:15, Null0
```

**Example 14-20** displays the BGP table for XR4 and R5. XR4 does not contain the 172.16.1.0/24 and 172.16.4.0/24 networks because they were suppressed by XR2. The 172.16.140.0/24 and 172.16.144.0/24 networks are not present in R5's BGP table because they were suppressed by R3.

### Example 14-20 XR4's and R5's BGP Tables

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/17	10.24.1.2			0 100	i
*> 172.16.128.0/17	10.24.1.2			0 100	i
*> 172.16.140.0/24	10.24.1.2			0 100	i
*> 172.16.144.0/24	10.24.1.2			0 100	i

```
Processed 4 prefixes, 4 paths
```

```
R5#show bgp ipv4 unicast
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/17	10.35.1.3	0		0 100	i
*> 172.16.1.0/24	10.35.1.3			0 100	i
*> 172.16.4.0/24	10.35.1.3			0 100	i
*> 172.16.128.0/17	10.35.1.3	0		0 100	i

### Flexible Route Suppression

Some traffic engineering designs require “leaking” routes, which is the advertisement of a subset of more specific routes in addition to performing the summary. Leaking routes can be done at the process by explicitly stating the prefixes to suppress, or on a per neighbor level by indicating which prefixes should not be suppressed.

### Selective Prefix Suppression

Selective prefix suppression explicitly lists the networks that should not be advertised along with the summary route to neighbor routers.

IOS uses a suppress map, which uses the keyword **suppress-map** *route-map-name* instead of using the **summary-only** keyword. In the referenced route map, only the prefixes that should be suppressed are permitted. IOS XR routers use the keyword **route-policy** *route-policy-name* in lieu of the **summary-only** keyword. In the route policy, the action command **suppress** is used after conditionally matching the prefixes that should be suppressed.

Note

The **suppress** command is allowed only in an RPL when used with the **aggregate-address** command.

Figure 14-8 demonstrates BGP selective prefix suppression with the following logic:

- R1 is advertising the 172.16.1.0/24 and 172.16.2.0/24 networks.
- XR2 and R3 summarize R1’s networks to the 172.16.0.0/16 network range.
- XR2 and R3 suppress the 172.16.1.0/24 network.

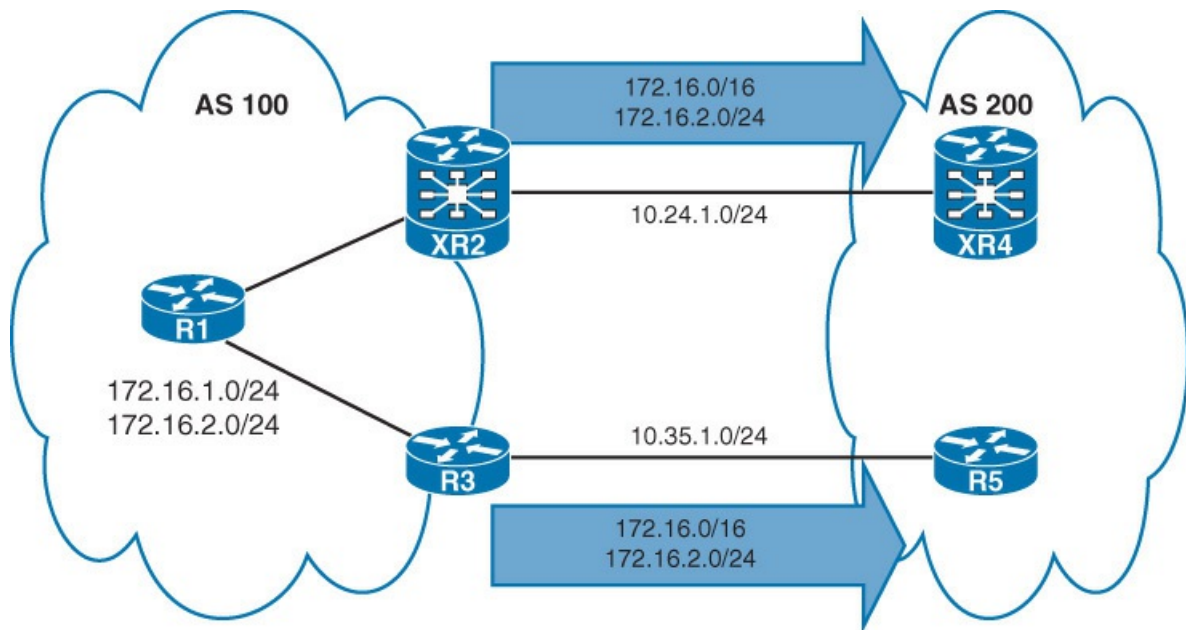


Figure 14-8 BGP Route Suppression

Example 14-21 demonstrates the BGP configuration on XR2 and R3 for the BGP suppression of the 172.16.1.0/24 network. R3's route map identifies only the routes that should be suppressed.

### Example 14-21 BGP Route Suppression Configuration

[Click here to view code image](#)

#### XR2

```
router bgp 100
 address-family ipv4 unicast
   aggregate-address 172.16.0.0/16 route-policy SUPPRESS-MAP
 !
 route-policy SUPPRESS-MAP
   if destination in (172.16.1.0/24) then
     suppress-route
   endif
 end-policy
```

#### R3

```
router bgp 100
 no bgp default ipv4-unicast
 neighbor 10.13.1.1 remote-as 100
 neighbor 10.35.1.5 remote-as 200
 !
 address-family ipv4
   aggregate-address 172.16.0.0 255.255.0.0 suppress-map SUPPRESS-MAP
 !
 route-map SUPPRESS-MAP permit 10
 match ip address prefix-list SUPPRESS-LIST
 !
 ip prefix-list SUPPRESS-LIST permit 172.16.1.0/24
```

Example 14-22 displays the BGP table and RIB for XR2 and R3. Notice that both routers have

suppressed the 172.16.1.0/24 network from being advertised.

### Example 14-22 XR2's and R3's RIB and BGP Tables

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show bgp ipv4 unicast
! Output omitted for brevity

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.0.0/16    0.0.0.0                      32768 i
s>i172.16.1.0/24   10.12.1.1             0    100    0 i
*>i172.16.2.0/24   10.12.1.1             0    100    0 i

Processed 3 prefixes, 3 paths
```

```
R3#show bgp ipv4 unicast
BGP table version is 9, local router ID is 10.35.1.3
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.0.0       0.0.0.0                      32768 i
s>i172.16.1.0/24   10.13.1.1             0    100    0 i
*>i172.16.2.0/24   10.13.1.1             0    100    0 i
```

Example 14-23 displays XR4's and R5's BGP table confirming that the 172.16.1.0/24 prefix is suppressed.

### Example 14-23 XR4's and R5's BGP Tables

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.0.0/16    10.24.1.2                      0 100 i
*> 172.16.2.0/24    10.24.1.2                      0 100 i

Processed 2 prefixes, 2 paths
```

```
R5#show bgp ipv4 unicast

   Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.0.0       10.35.1.3             0                      0 100 i
*> 172.16.2.0/24    10.35.1.3             0                      0 100 i
```

## Leaking Suppressed Routes

The **summary-only** keyword suppresses all the more specific routes of an aggregate address from being advertised. Once a route is suppressed, it is possible to still advertise the suppressed route to a specific neighbor.

IOS routers use an unsuppress map with the BGP neighbor address family configuration command **neighbor ip-address unsuppress-map route-map-name**. In the referenced route map, only the prefixes that should be leaked are permitted. IOS XR routers use an outbound route policy with the action command **unsuppress** to indicate which prefixes should be leaked.

### Note

The **unsuppress** command is allowed only in an RPL attached to a BGP neighbor's address family. To prevent unintentional filtering of other routes, be sure to grant a ticket to other routes so that they can pass the default drop in the route policy.

Figure 14-9 demonstrates leaking of BGP suppressed routes with the following logic:

- R1 is advertising the 172.16.1.0/24 and 172.16.2.0/24 networks.
- XR2 and R3 summarize and suppress R1's networks to the 172.16.0.0/16 range.
- XR2 and R3 will leak the 172.16.1.0/24 network to AS200.

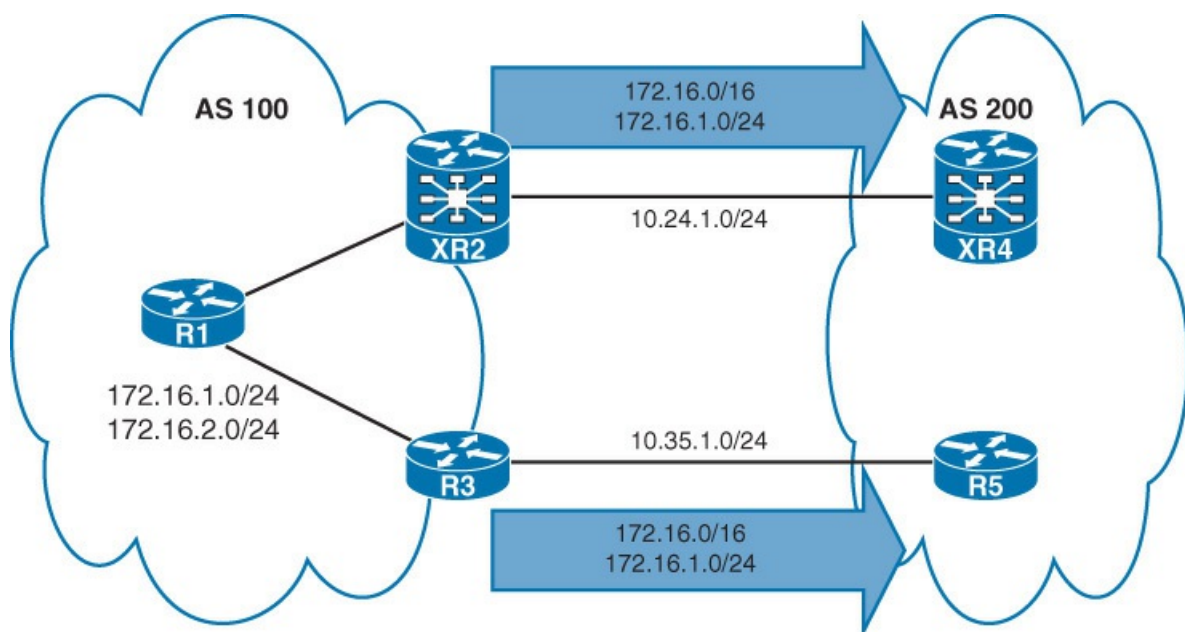


Figure 14-9 Leaking of BGP Suppressed Routes

Example 14-24 demonstrates the BGP configuration on XR2 and R3 for leaking the 172.16.1.0/24 network to the AS200 BGP peers. Notice that the route policy on XR2 includes an additional **pass** parameter to advertise routes that are not matched by the unsuppressed statement. This is required to prevent regular routes from being dropped by implicit deny in the route policy.

### Example 14-24 Leaked Suppressed BGP Route Configuration

[Click here to view code image](#)



**XR2**

```
router bgp 100
  address-family ipv4 unicast
    aggregate-address 172.16.0.0/16 summary-only
  !
  neighbor 10.24.1.4
    remote-as 200
    address-family ipv4 unicast
    route-policy PASSALL in
    route-policy UNSUPPRESS out
  !
  route-policy UNSUPPRESS
    if destination in (172.16.1.0/24) then
      unsuppress-route
    else
      pass
    endif
  end-policy
```

**R3**

```
router bgp 100
  no bgp default ipv4-unicast
  neighbor 10.13.1.1 remote-as 100
  neighbor 10.35.1.5 remote-as 200
  !
  address-family ipv4
    aggregate-address 172.16.0.0 255.255.0.0 summary-only
    neighbor 10.13.1.1 activate
    neighbor 10.35.1.5 activate
    neighbor 10.35.1.5 unsuppress-map UNSUPPRESS-MAP
  exit-address-family
  !
  route-map UNSUPPRESS-MAP permit 10
  match ip address prefix-list UNSUPPRESS-LIST
  !
  ip prefix-list UNSUPPRESS-LIST seq 5 permit 172.16.1.0/24
```

**Example 14-25** displays the BGP table and RIB for XR2 and R3. Notice that the 172.16.1.0/24 and 172.16.2.0/24 networks are suppressed on both routers.

**Example 14-25 XR2's and R3's RIB and BGP Tables**

**Click here to view code image**

```
RP/0/0/CPU0:XR2#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

```
Status codes: s suppressed, d damped, h history, * valid, > best  
i - internal, r RIB-failure, S stale
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/16	0.0.0.0			32768	i
s>i172.16.1.0/24	10.12.1.1	0	100	0	i
s>i172.16.2.0/24	10.12.1.1	0	100	0	i

```
Processed 3 prefixes, 3 paths
```

```
R3#show bgp ipv4 unicast
```

```
BGP table version is 9, local router ID is 10.35.1.3
```

```
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,  
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,  
x best-external, a additional-path, c RIB-compressed,
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0	0.0.0.0			32768	i
s>i172.16.1.0/24	10.13.1.1	0	100	0	i
s*>i172.16.2.0/24	10.13.1.1	0	100	0	i

**Example 14-26** displays the BGP table for XR4 and R5. The suppressed 172.16.1.0/24 network was successfully advertised to the AS200 routers, but the other suppressed network (172.16.2.0/24) was not advertised.

### Example 14-26 XR4's and R5's BGP Tables

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/16	10.24.1.2			0 100	i
*> 172.16.1.0/24	10.24.1.2			0 100	i

```
Processed 2 prefixes, 2 paths
```

```
R5#show bgp ipv4 unicast
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0	10.35.1.3	0		0 100	i
*> 172.16.1.0/24	10.35.1.3			0 100	i

### Atomic Aggregate

Aggregated routes act like new BGP routes with a shorter prefix length. When a BGP router summarizes a route, it will not advertise the AS\_Path information from before the aggregation. BGP path attributes like AS\_Path, Multi-Exit Discriminator (MED), and BGP communities are not included in the new BGP advertisement. The atomic aggregate attribute indicates that a loss of path information has occurred.

Figure 14-10 demonstrates the loss of BGP path attributes in the aggregate route:

- R1 and R2 are advertising the 172.16.1.0/24 and 172.16.2.0/24 networks.
- R3 is aggregating the routes into the 172.16.0.0/22 network range, which is advertised to all of R3's peers.
- R1 and R2 install the BGP aggregated route because the AS\_Path attribute was not copied to the aggregate route.

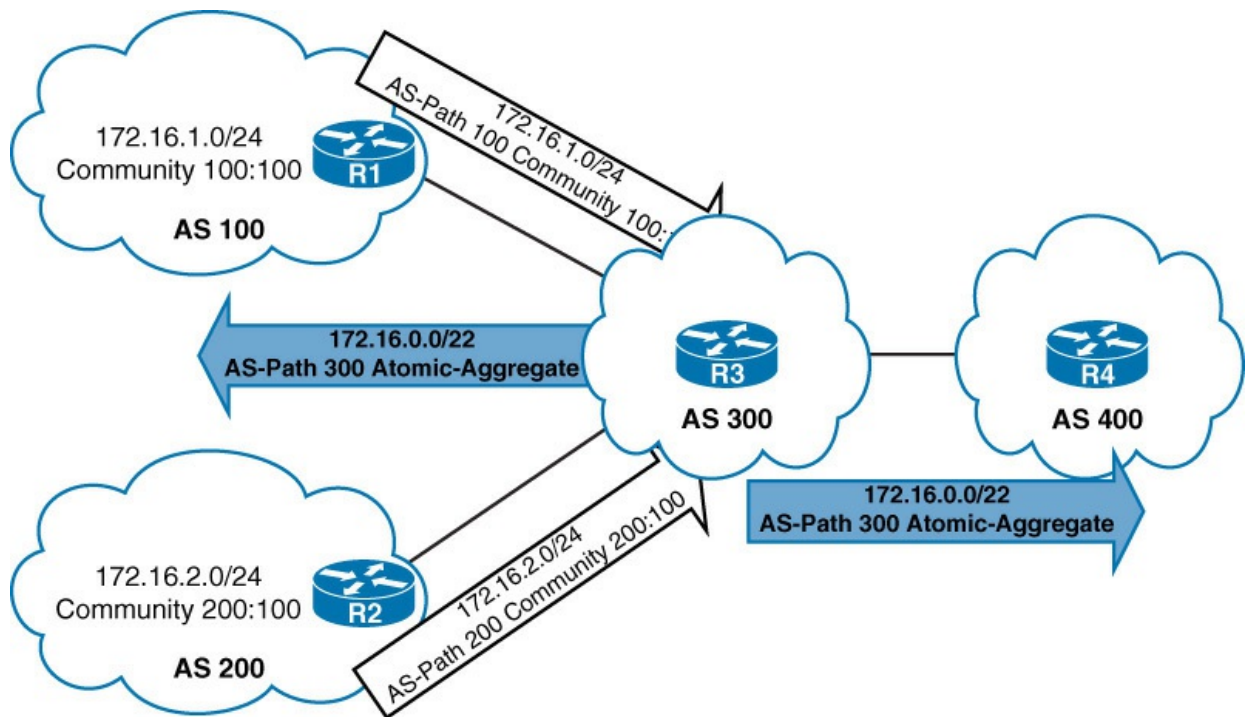


Figure 14-10 Atomic Aggregate Topology

Example 14-27 displays R1's BGP route 172.16.1.0/24 advertised to R3. Notice that the AS\_Path of 100 and BGP community of 100:100.

### Example 14-27 172.16.1.0/24 BGP Path Information

[Click here to view code image](#)

```
R3#show bgp ipv4 unicast 172.16.1.0
BGP routing table entry for 172.16.1.0/24, version 13
Paths: (1 available, best #1, table default, Advertisements suppressed by an aggregate.)
Not advertised to any peer
Refresh Epoch 1
100
  10.13.1.1 from 10.13.1.1 (192.168.1.1)
  Origin IGP, metric 0, localpref 100, valid, external, best
  Community: 100:100
```

R3's aggregate route (summary) does not include the BGP communities (including AS\_Path history) for the routes in the summarization range. R1 and R2 will install the 172.16.0.0/22 summary route because their AS\_Path is not listed in the AS\_Path attribute and passes the

AS\_Path loop check.

**Example 14-28** displays the BGP path information for the 172.16.0.0/22 summary network. The AS\_Path of the aggregated route displays only the aggregating router but does not include the AS\_Path of the routes being summarized (AS100 or AS200), nor is the BGP community included in the routes being summarized. The BGP path information indicates that this is an aggregated prefix and was aggregated by R3 (192.168.3.3). The *atomic-aggregate* in the route indicates a loss of information occurred during aggregation on the aggregating router.

### **Example 14-28** 172.16.0.0/22 BGP Path Information

[Click here to view code image](#)

```
R1#show bgp ipv4 unicast 172.16.0.0
BGP routing table entry for 172.16.0.0/21, version 5
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 1
  300, (aggregated by 300 192.168.3.3)
    10.13.1.3 from 10.13.1.3 (192.168.3.3)
      Origin IGP, metric 0, localpref 100, valid, external, atomic-aggregate, best
```

### **Route Aggregation with AS\_SET**

To keep the BGP path information history, the optional **as-set** keyword may be used with the **aggregate-address** command. As the router generates the aggregate route, BGP attributes from the summarized routes are copied over to it. The AS\_Path settings from the original prefixes are stored in the AS\_Set portion of the AS\_Path. The AS\_Set is displayed within brackets and counts only as one hop, even if multiple autonomous systems are listed.

**Figure 14-11** revisits the previous topology but demonstrates that all the path attributes from R1 and R2 are contained in the aggregated route:

- R1 and R2 are advertising the 172.16.1.0/24 and 172.16.2.0/24 networks.
- R3 is aggregating the routes into the 172.16.0.0/22 network range with the **as-set** and **summary-only** keywords, which is advertised to all of R3's peers.
- R1 and R2 discard the BGP aggregated route as a loop-prevention mechanism because their autonomous system number (ASN) is listed in the AS\_Path of the aggregate route.

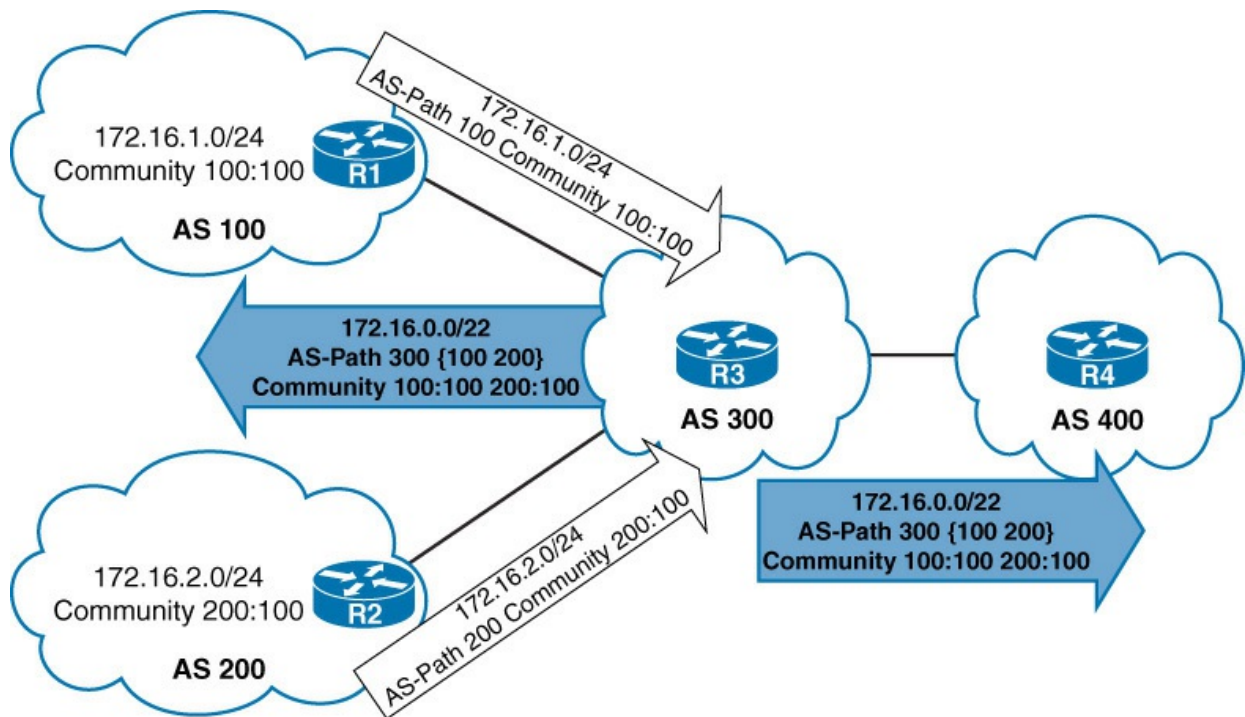


Figure 14-11 AS\_Set Topology

Example 14-29 demonstrates the BGP configuration for route aggregation with the **asset** keyword to ensure that all BGP path attributes are copied to the aggregated route. The IOS and IOS XR configuration provide for R3.

### Example 14-29 BGP Configuration for Route Aggregation with AS\_Set

[Click here to view code image](#)

```
IOS XR
router bgp 300
address-family ipv4 unicast
aggregate-address 172.16.0.0/22 summary-only as-set
```

```
IOS
router bgp 300
no bgp default ipv4-unicast
!
address-family ipv4
aggregate-address 172.16.0.0 255.255.248.0 summary-only as-set
```

Example 14-30 displays the 172.16.0.0/22 BGP aggregate route. Notice that the route contains the AS\_Path information for AS100 and AS200 in brackets (AS\_Set fields) and the BGP communities are included, too. The atomic aggregate flag is not present.

### Example 14-30 BGP Aggregate Route with Path Information Maintained

[Click here to view code image](#)

```

R4#show bgp ipv4 unicast 172.16.0.0
BGP routing table entry for 172.16.0.0/22, version 13
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 1
  300 {100,200}, (aggregated by 300 192.168.3.3)
    10.34.1.3 from 10.34.1.3 (192.168.3.3)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Community: 100:100 100:200

```

### Route Aggregation with Selective Advertisement of AS\_Set

Using the AS\_Set feature with network aggregation combines all of the attributes of the original prefixes into the aggregated prefixes. This might cause issues with your routing policy. For example, if one of the prefixes contains the No\_Export BGP community, the aggregate address will not be exported. To resolve these types of problems, selectively choose the routes that the path attributes will copy to the aggregate route.

Figure 14-12 demonstrates the selective route selection for copying BGP path attributes to the aggregate route:

- R1 and R2 are advertising the 172.16.1.0/24 and 172.16.2.0/24 networks.
- R3 is aggregating routes into the 172.16.0.0/22 network range with selective usage of the **as-set** and **summary-only** keywords. R3 is denying any attributes from AS100 from being applied to the aggregate address.
- The end result is that the AS\_Set for the aggregate route is 200 and the community is 200:100.

To conditionally match routes for aggregation, use the optional **advertise-map route-map-name** keyword on IOS routers or add the optional **advertise-map route-policy-name** keyword to the aggregation statement for IOS XR routers. The route map/route policy should conditionally match and deny attributes that you do or do not want to include in your aggregated route.

Example 14-31 demonstrates the selective AS\_Set selection for R3 for routes that contain the BGP community 100:100. The IOS and IOS XR configuration is provided for R3.

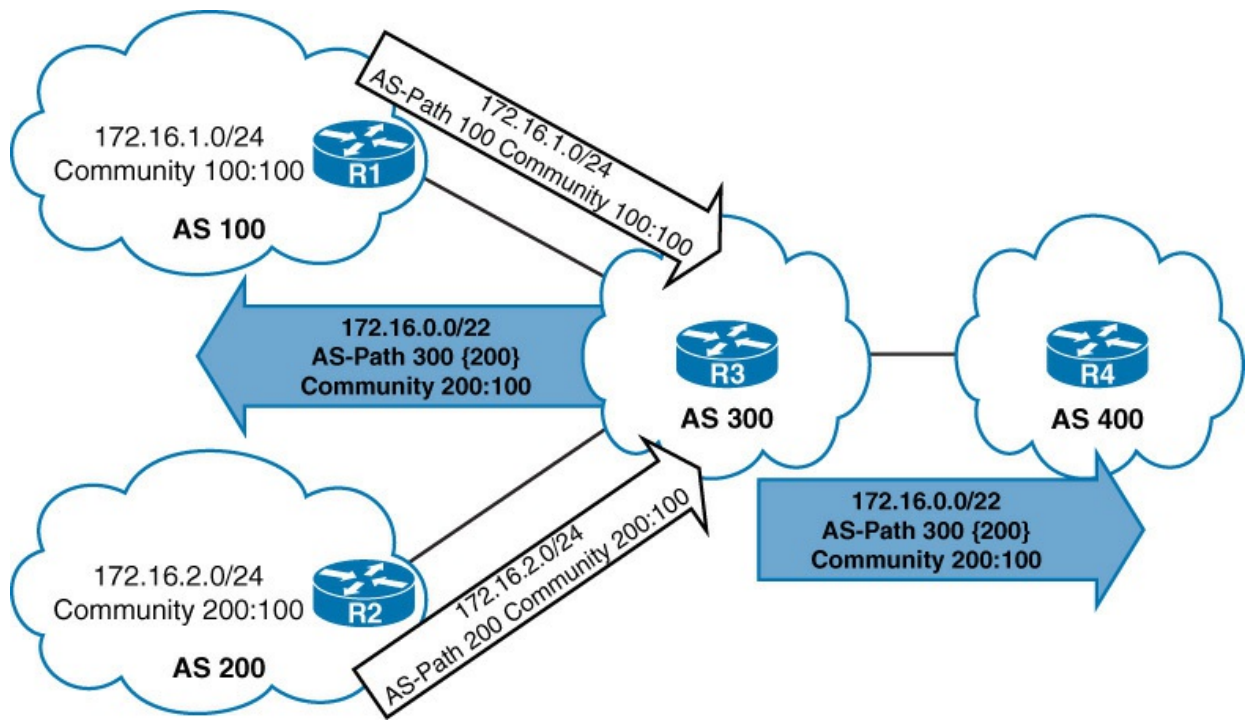


Figure 14-12 Route Aggregation with Selective AS\_Set

### Example 14-31 BGP Configuration for Route Aggregation with AS\_Set

[Click here to view code image](#)

```
IOS XR
route-policy SUMMARY
  if not community matches-any (100:100) then
    pass
  endif
end-policy
!
router bgp 300
address-family ipv4 unicast
  aggregate-address 172.16.0.0/22 summary-only as-set route-policy SUMMARY
```

```
IOS
ip community-list 1 permit 100:100
!
route-map SUMMARY deny 5
  match community 1
route-map SUMMARY permit 10
!
router bgp 300
  no bgp default ipv4-unicast
!
address-family ipv4
  aggregate-address 172.16.0.0 255.255.248.0 summary-only as-set advertise-map
  SUMMARY
```

Example 14-32 displays the aggregated prefix on R4. Notice that the AS\_Set information only contains the BGP attributes from AS200 and that the *atomic-aggregate* attribute is set because

path information from AS100 is not retained by the summary route.

### Example 14-32 BGP Aggregate Route with Path Information Maintained

[Click here to view code image](#)

```
R4#show bgp ipv4 unicast 172.16.0.0
BGP routing table entry for 172.16.0.0/22, version 13
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 1
  300 {200}, (aggregated by 300 192.168.3.3)
    10.34.1.1.3 from 10.34.1.3 (192.168.3.3)
      Origin IGP, metric 0, localpref 100, valid, external, atomic-aggregate, best
      Community: 100:200
```

## DEFAULT ROUTE ADVERTISEMENT

Advertising a default route into the BGP table requires that the default route to exist in the RIB and that the BGP configuration command **default-information originate** is used. The redistribution of a default route or usage of a network 0.0.0.0/0 will not work without the **default-information originate** command.

[Example 14-33](#) demonstrates the necessary BGP configuration for IOS and IOS XR so that a default route is advertised into the BGP Loc-RIB table. A static default route to Null0 was used to install a default route into the RIB.

### Example 14-33 Global BGP Default Route Advertisement

[Click here to view code image](#)

```
IOS XR
router static
  address-family ipv4 unicast
    0.0.0.0/0 Null0
  !
!
router bgp 100
  default-information originate
  address-family ipv4 unicast
    network 0.0.0.0/0
```

```
IOS
ip route 0.0.0.0 0.0.0.0 Null0
!
router bgp 100
  no bgp default ipv4-unicast
  !
  address-family ipv4
    network 0.0.0.0
    default-information originate
  exit-address-family
```



## DEFAULT ROUTE ADVERTISEMENT PER NEIGHBOR

Some network topologies restrict the size of the BGP advertisements to a neighbor because the remote router does not have enough processing power or memory for the full BGP routing table. Connectivity is still required, so the peering routers only advertise the default route to the remote router.

Figure 14-13 demonstrates the advertisement of a default route to a BGP neighbor for remote routers that do not want to receive the full BGP routing table:

- R1, R5, and R6 require full connectivity to each other, but due to memory limitations can accept only one route.
- R1, R5, and R6 advertise their local network to the BGP peer in AS200, which maintains a complete BGP table.
- XR2, R3, and R4 should advertise only a default route to the BGP peers so that they can maintain connectivity and a small BGP table.

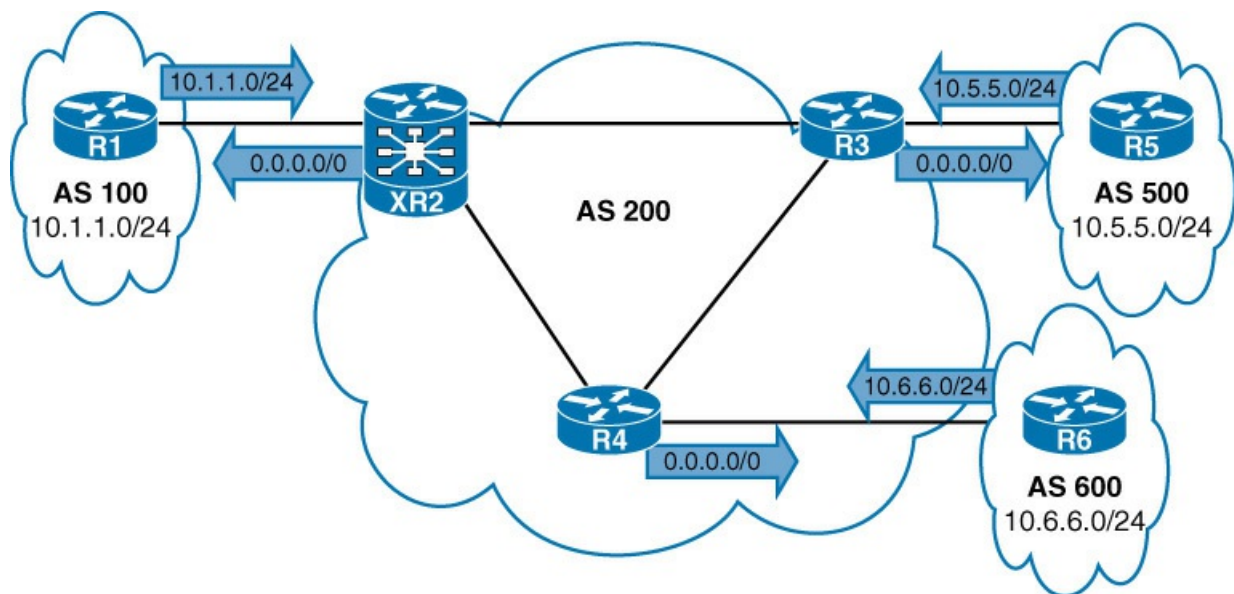


Figure 14-13 Default Route Advertisement per Neighbor

A default route is advertised to a BGP peer with the BGP address family configuration command **neighbor ip-address default-originate** for IOS routers or with the BGP neighbor address family configuration command **default-originate** for IOS XR routers. Default route advertisement to a specific neighbor does not require that a default route is present in the RIB or BGP Loc-RIB table.

Example 14-34 demonstrates the BGP configuration for XR2 and R3 where they are advertising a default route to their eBGP peers.

**Example 14-34** Configuration for Default Route Advertisement to a BGP Peer

[Click here to view code image](#)

### XR2

```
router bgp 200
  address-family ipv4 unicast
  !
  neighbor 10.12.1.1
    remote-as 100
  address-family ipv4 unicast
    route-policy PASSALL in
    route-policy PASSALL out
    default-originate
```

### R3

```
router bgp 200
  no bgp default ipv4-unicast
  neighbor 10.35.1.5 remote-as 500
  !
  address-family ipv4
    neighbor 10.35.1.5 activate
    neighbor 10.35.1.5 default-originate
  exit-address-family
```

### Note

A behavior difference between IOS and IOS XR occurs when a default route is already present in the BGP table. IOS routers will advertise the route as if it were the originating router. (None of the existing attributes are passed to the peer.) IOS XR routers will advertise the network to the peer as it exists in the BGP table with the entire default route attributes (AS\_Path and so on).

## CONDITIONAL ROUTE ADVERTISEMENT

Conditional route advertisement may be used for traffic engineering and controlling network traffic flow. [Figure 14-14](#) illustrates a topology where AS100 uses two service providers for Internet connectivity. Service provider B's rate (cost per megabyte) is significantly higher than service provider A's rate (cost per megabyte), and AS100 prefers to route all traffic through AS200 (service provider A). In the event that the 10.12.1.0/24 link fails, AS100 needs to advertise the network prefixes to AS300. Because AS300's rate is more expensive, the advertisement should be withdrawn upon restoration of service between AS100 and AS200.

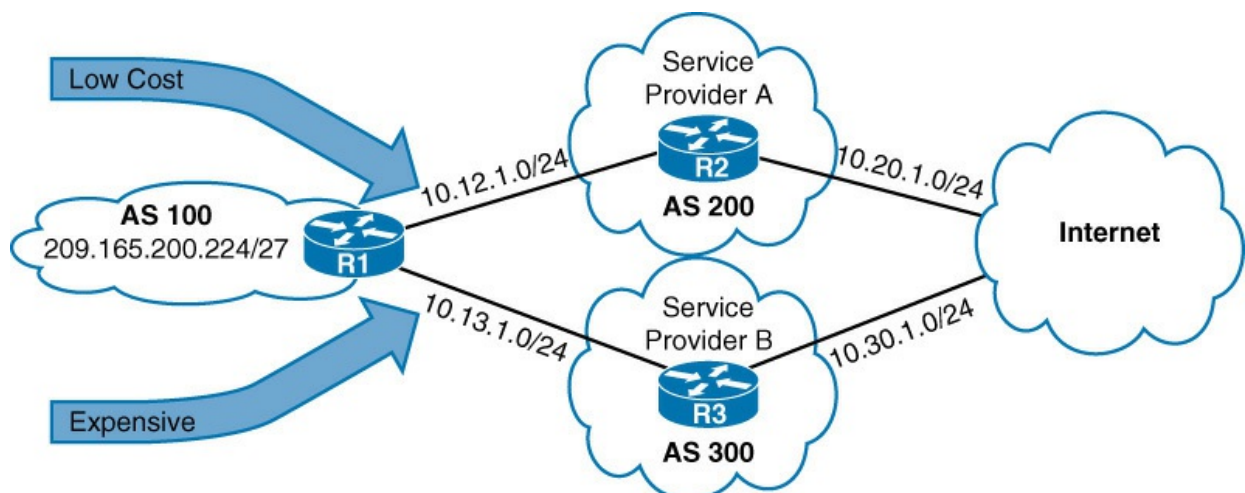


Figure 14-14 Conditional Route Advertisement

Conditional route advertisement uses the BGP address family configuration command

**neighbor ip-address advertise-map advertise-route-map {non-exist | exist} conditional-route-map-name** structure. The first route map indicates the routes that should be advertised, and the second route map checks to see if the routes are present or not present in the BGP table. If the **non-exist** keyword is used and the route is missing, the route is advertised. If the **exist** keyword is used and the route is present, the route is advertised.

**Example 14-35** demonstrates R1's BGP configuration for the conditional route advertisement of the 209.165.220.224/27 network if the 10.12.1.0/24 network is not in the BGP table.

### **Example 14-35** Configuration for Conditional Route Advertisement

[Click here to view code image](#)

```
IOS
router bgp 100
  no bgp default ipv4-unicast
  neighbor 10.12.1.2 remote-as 200
  neighbor 10.23.1.3 remote-as 300
  !
  address-family ipv4
    network 209.165.200.224 mask 255.255.255.224
    neighbor 10.12.1.2 activate
    neighbor 10.23.1.3 activate
    neighbor 10.23.1.3 advertise-map ADVERTISE non-exist NOEXIST
  exit-address-family
  !
  route-map ADVERTISE permit 10
    match ip address prefix-list LOCAL
  !
  Route-map NOEXIST permit 10
    match ip address prefix-list AS100TO200
  !
  ip prefix-list AS100TO200 permit 10.12.1.0/24
  !
  ip prefix-list LOCAL permit 209.165.200.224/27
```

#### Note

IOS XR does not support conditional route advertisement.

## OUTBOUND ROUTE FILTERING

Enterprises typically use a static default route or BGP peering to connect to the Internet. If they connect with BGP, the service provider usually gives the entire Internet table (500,000+ routes), only the routes for the service provider network blocks, or just a default route. If the customer wants only a small subsection of the Internet table on their router (customer edge [CE]), the CE router can filter out the routes that they do not want on an inbound route map/route policy, but the routes are still sent from the service provider (provider edge [PE]) router. This wastes bandwidth on the peering link, and consumes processing time on the CE and PE routers.

Customers may want the service provider to filter the routes before they are sent to reduce wasting bandwidth. The service provider must manage requests from the customer and update the policies on the PE routers accordingly. This is a burden to the service provider and is rarely agreed to.

Outbound route filtering (ORF) provides a solution for the customer while preventing an unnecessary burden on the service provider. ORF allows the CE device to provide the filter to the PE router dynamically. ORF is a session capability that is established when BGP establishes the peering between the two neighbors. Each peer must support route refresh and advertises its capability to send, receive, or send and receive (both) ORFs. The ORF senders will then send the prefix list used to filter the route advertisements.

By default, ORF session capability is not advertised during BGP session establishment.

IOS routers configure the ORF capability for a neighbor with the BGP address family configuration command **neighbor ip-address capability orf prefix {both | send | receive}**, and IOS XR routers use the BGP neighbor address family configuration command **capability orf prefix {both | none | receive | send}**.

IOS routers use the BGP address family configuration command **neighbor ip-address prefix-list prefix-list-name** to define the list of acceptable prefixes for advertisement via ORF. IOS XR routers use the equivalent BGP neighbor address family configuration command **orf route-policy route-policy-name**. The IOS XR route policy must use the conditional match of **if orf prefix in {prefix-set-name | (high-order-bit-pattern [/high-order-bit-count] [ge ge-value] [le le-value])** in an *if-then* programmatic structure. The only viable actions are pass or drop. This ensures that the route policy structure is easy to read and can easily be advertised to other non-  
IOS XR routers.

Note

A route refresh may be required for any changes to the ORF filter.

Example 14-36 demonstrates the ORF BGP configuration between XR1 and R2. XR1 is requesting that R2 filter out the 172.16.0.0/16 network prefixes, and R2 is requesting that XR1 filter the 172.20.0.0/16 network prefixes. Notice that the route policy structure uses only an *if-then* programmatic structure.

### **Example 14-36** BGP ORF Configuration

[Click here to view code image](#)

**XR1**

```
router bgp 100
  address-family ipv4 unicast
    network 172.20.1.0/24
    network 192.168.1.1/32
  !
  neighbor 10.12.1.2
    remote-as 200
    address-family ipv4 unicast
      route-policy PASSALL in
      route-policy PASSALL out
      orf route-policy ORF
      capability orf prefix both
  !
route-policy ORF
  if orf prefix in (172.16.0.0/12 le 32) then
    drop
  endif
  if orf prefix in (0.0.0.0/0 le 32) then
    pass
  endif
end-policy
```

**R2**

```
router bgp 200
  no bgp default ipv4-unicast
  neighbor 10.12.1.1 remote-as 100
  !
  address-family ipv4
    network 172.16.1.0 mask 255.255.255.0
    network 192.168.2.2 mask 255.255.255.255
    neighbor 10.12.1.1 activate
    neighbor 10.12.1.1 capability orf prefix-list both
    neighbor 10.12.1.1 prefix-list ORF-PREFIXLIST in
  exit-address-family

ip prefix-list ORF-PREFIXLIST seq 5 deny 172.20.0.0/13 ge 14
ip prefix-list ORF-PREFIXLIST seq 10 permit 0.0.0.0/0 le 32
```

**Example 14-37** verifies that the routers were able to agree on and establish ORF capabilities. XR1 sends R2 the list of routes in the ORF route policy that R2 should advertise to it, and R2 sends XR1 the list of routes in the ORF-PREFIXLIST prefix list that XR1 should advertise to it. Notice that the IOS output provides a count for the number of prefixes filtered by ORF.

**Example 14-37 BGP ORF Verification**

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast neighbors 10.12.1.2
! Output omitted for brevity

BGP neighbor is 10.12.1.2
For Address Family: IPv4 Unicast
  AF-dependent capabilities:
    Outbound Route Filter (ORF) type (128) Prefix:
      Send-mode: advertised, received
      Receive-mode: advertised, received
    Outbound Route Filter (ORF): sent; received (2 entries)
  Incoming update prefix filter is ORF
```

```
R2#show bgp ipv4 unicast neighbors 10.12.1.1
! Output omitted for brevity

BGP neighbor is 10.12.1.1, remote AS 100, external link
For address family: IPv4 Unicast
  AF-dependant capabilities:
    Outbound Route Filter (ORF) type (128) Prefix-list:
      Send-mode: advertised, received
      Receive-mode: advertised, received
    Outbound Route Filter (ORF): sent; received (2 entries)
  Incoming update prefix filter list is ORF-PREFIXLIST

Local Policy Denied Prefixes:
Outbound      Inbound
-----      -
Bestpath from this peer:      5      n/a
ORF prefix-list:              4      n/a
Total:                        9      0
```

## BACKDOOR NETWORKS

BGP uses an administrative distance (AD) of 20 for eBGP routes and 200 for iBGP routes and works well in almost all network designs. Occasionally, a route learned via an interior gateway protocol (IGP) needs to take preference over a route learned via eBGP. This can be accomplished using the BGP backdoor network feature. A BGP backdoor network is treated as a local network and raises the AD for the eBGP-learned route from 20 to 200, and the router prohibits the advertisement of the backdoor network to eBGP peers.

BGP backdoor networks use the BGP address family configuration command **network network mask subnet-mask backdoor** on IOS routers, and the command **network network/prefix-length backdoor** on IOS XR routers. The following scenario provides a use case for the BGP backdoor network feature.

**Figure 14-15** illustrates a topology where AS200 and AS 400 connect via AS300 and are mutually redistributing OSPF into BGP. AS200 and AS300 plan to connect to each other through AS300 so that servers in the 10.1.1.0/24 network can communicate with servers in the 10.5.5.0/24 network.

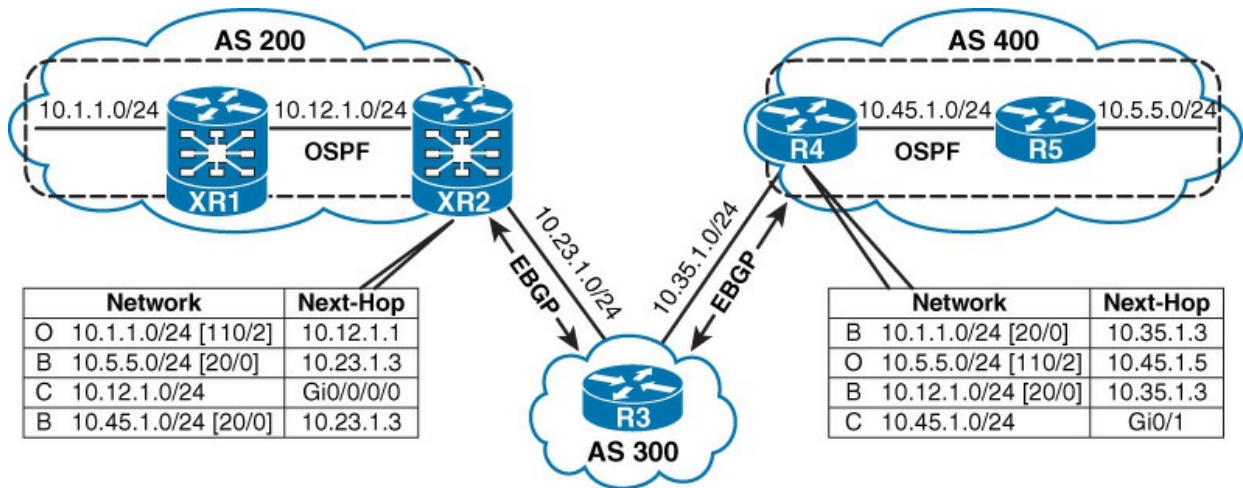


Figure 14-15 AS200 and AS 400 Connectivity

The throughput between the AS200 and AS 400 is insufficient, so a dedicated link (10.24.1.0/24) between XR2 and R4 is established, as shown in Figure 14-16. Packets between AS200 and AS 400 still route through AS300 because AS200 routes are learned via eBGP with an AD of 20, which is lower than the AD advertised by the OSPF direct path. The AD for routes is shown in brackets.

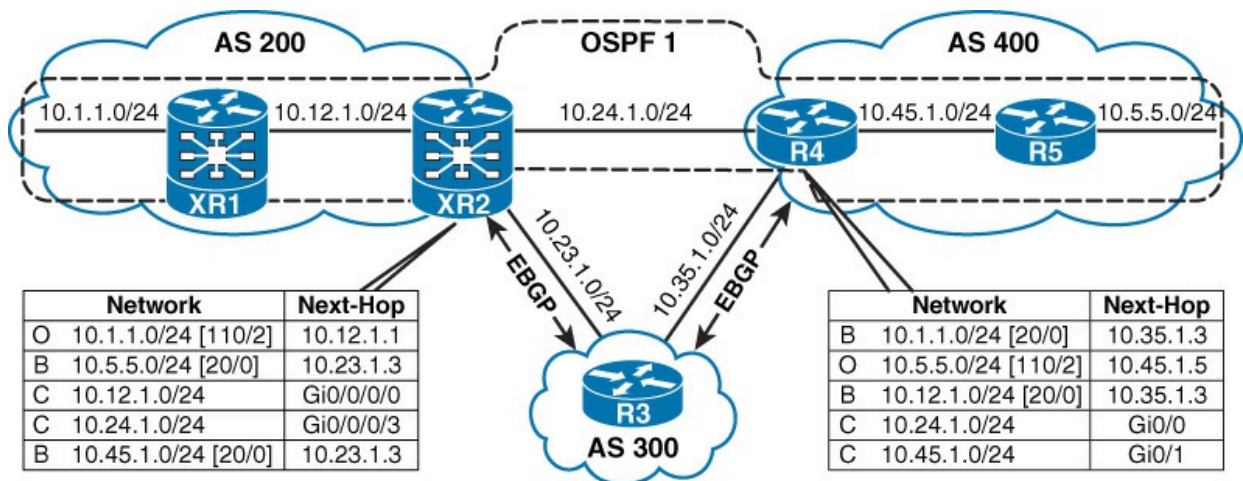


Figure 14-16 Suboptimal Routing Between AS200 and AS 400

XR2 and R4 configure the remote networks as BGP backdoor networks. This raises the AD of the backdoor networks to 200, which is higher than the AD of OSPF. XR2 and R4 then install the remote networks into the RIB with the OSPF path. Figure 14-17 provides the topology and routing table for XR2 and R4. Notice that the next hop for the remote networks uses the 10.24.1.0/24 network instead of routing traffic through AS300.



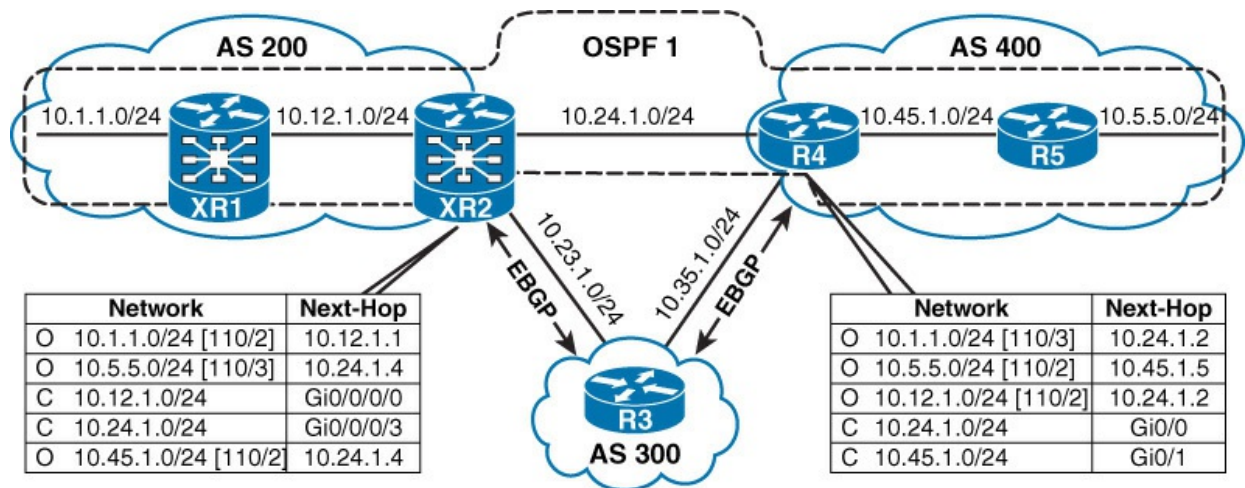


Figure 14-17 BGP Backdoor Network Topology

Example 14-38 demonstrates the BGP backdoor network configuration. XR2 configures the AS 400 networks (10.45.1.0/24 and 10.5.5.0/24) and R4 configures AS200 networks (10.1.1.0/24 and 10.12.1.0/24) as BGP backdoor networks.

### Example 14-38 BGP Backdoor Network Configuration

[Click here to view code image](#)

#### XR2

```
router bgp 200
  bgp router-id 192.168.2.2
  address-family ipv4 unicast
    network 10.1.1.0/24
    network 10.5.5.0/24 backdoor
    network 10.12.1.0/24
    network 10.45.1.0/24 backdoor
  redistribute ospf 1
  !
  neighbor 10.23.1.3
    remote-as 300
    address-family ipv4 unicast
      route-policy PASSALL in
      route-policy PASSALL out
```

#### R4

```
router bgp 400
  bgp log-neighbor-changes
  neighbor 10.34.1.3 remote-as 300
  !
  address-family ipv4
    network 10.1.1.0 mask 255.255.255.0 backdoor
    network 10.5.5.0 mask 255.255.255.0
    network 10.12.1.0 mask 255.255.255.0 backdoor
    network 10.45.1.0 mask 255.255.255.0
  redistribute ospf 1
  neighbor 10.34.1.3 activate
  exit-address-family
```



## MAXIMUM AUTONOMOUS SYSTEM

The AS\_Path maintains a complete list of the autonomous systems that a route has traversed to reach a router. BGP uses the AS\_Path length as a step to determine the BGP best path. The AS\_Path is often prepended to influence traffic patterns. In rare instances, the prepending policies have led to problems with routers that could not handle routes with more than 100 ASNs in the AS\_Path. The original solution required filtering prefixes with an excessive number of ASNs in the AS\_Path.

IOS routers can filter out routes with an excessive number of ASNs in their AS\_Path with the BGP configuration command **bgp maxas-limit 1-254**. Any routes with an AS\_Path count higher than the configured value are discarded. IOS XR routers can accomplish the same functionality by conditionally matching the routes in a route policy with the **as-path length** feature (described in [Chapter 11](#)) in a route policy.

### Note

Only the ASNs shown in the AS\_Path are counted. The ASNs for the router receiving the route are not part of the count.

[Example 14-39](#) displays XR1's and R2's BGP table as a reference while demonstrating the maximum autonomous system feature. Two routes have an AS\_Path length of 5, and two routes have an AS\_Path length of 1.

### Example 14-39 Reference BGP Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 192.168.1.1/32	10.12.1.1	0		0	100 110 120 130 140 i
*>i192.168.2.2/32	10.12.1.1	0	100	0	100 110 120 130 140 i
*> 192.168.3.3/32	10.12.1.1	0		0	100 i
*>i192.168.4.4/32	10.12.1.1	0	100	0	100 i

Processed 4 prefixes, 4 paths

```
R2#show ip bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i192.168.1.1/32	10.23.1.2	0	100	0	100 110 120 130 140 i
*> 192.168.2.2/32	10.23.1.2	0		0	100 110 120 130 140 i
*>i192.168.3.3/32	10.23.1.2	0	100	0	100 i
*> 192.168.4.4/32	10.23.1.2	0		0	100 i

[Example 14-40](#) demonstrates the BGP configuration for XR1 and R2 that will accept only routes with an AS\_Path length of 3 or lower.

### Example 14-40 BGP Configuration for Restricting AS\_Path Length

[Click here to view code image](#)

**XR1**

```

router bgp 200
address-family ipv4 unicast
!
neighbor 10.12.1.1
remote-as 100
address-family ipv4 unicast
route-policy MAXAS-LIMIT in
!
route-policy MAXAS-LIMIT
if as-path length ge 3 then
drop
endif
pass
end-policy

```

**R2**

```

router bgp 200
bgp maxas-limit 3
no bgp default ipv4-unicast
neighbor 10.23.1.2 remote-as 200
!
address-family ipv4
neighbor 10.23.1.2 activate
exit-address-family

```

**Example 14-41** provides XR1's and R2's BGP table after the AS\_Path length restriction is applied to the BGP configuration. Notice that the 192.168.1.1/32 and 192.168.2.2/32 networks are filtered because their AS\_Path length exceeded the maximum autonomous system count.

**Example 14-41 XR1's and R2's BGP Table**

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 192.168.3.3/32	10.12.1.1	0		0	100 i
*>i192.168.4.4/32	10.12.1.1	0	100	0	100 i

Processed 2 prefixes, 2 paths

```
R2#show ip bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i192.168.3.3/32	10.23.1.2	0	100	0	100 i
*> 192.168.4.4/32	10.23.1.2	0		0	100 i

Note

IOS will log network prefix's filtered out with the **bgp maxas-limit** command. The following log message was captured in the example:

```
04:20:07.385: %BGP-6-ASPATH: Long AS path 100 110 120 130 140 received from 10.23.1.2: Prefixes: 192.168.1.1/32
```

## MAXIMUM PREFIX

Multiple Internet outages have occurred because a router received more routes than it could handle. The *BGP* maximum prefix feature restricts the amount of routes that are received from a BGP peer. This feature ensures that the BGP table does not overwhelm the router by exceeding its memory or processing capability. Prefix limits are typically set for BGP peers on low-end routers as a safety mechanism to ensure they do not become overloaded.

IOS routers can place prefix restrictions on a BGP neighbor with the BGP address family configuration command **neighbor ip-address maximum-prefix prefix-count [warning-percentage] [restart time] [warning-only]**. IOS XR routers use the command **maximum-prefix prefix-count [warning-percentage] [restart time] [warning-only]** to apply the prefix restriction.

When a peer advertises more routes than the maximum prefix count, the peer moves the neighbor to the *idle (PfxCt)* state in the Finite State Machine (FSM), closes the BGP session, and sends out the appropriate syslog message. The BGP session will not reestablish automatically by default. This behavior prevents a continuous cycle of loading routes, resetting the session, and reloading the routes again. If you want to restart the BGP session after a certain number of minutes, you may use the optional keyword **restart time**.

A warning is not generated before the prefix limit is reached. By adding a *warning-percentage* (1–100) after the maximum prefix count, a warning message will be sent when the percentage is exceeded. The syntax for a maximum of 100 prefixes with a warning threshold of 75 is **maximum-prefix 100 75**. When the threshold is reached, the router reports the following warning message:

```
%ROUTING-BGP-5-MAXPFX : No. of IPv4 Unicast prefixes received from 192.168.1.1 has reached 75, max 100
```

The maximum prefix behavior can be changed with the optional keyword **warning-only** so that a warning message is generated instead of closing the BGP session. Once the threshold has been reached, additional prefixes are discarded.

[Example 14-42](#) demonstrates maximum prefix configuration for IOS and IOS XR so that each peer can only advertise two prefixes.

### **Example 14-42** Maximum Prefix Configuration

[Click here to view code image](#)

**XR**

```
router bgp 100
  address-family ipv4 unicast
  !
  neighbor 10.12.1.2
    remote-as 200
    address-family ipv4 unicast
    maximum-prefix 2
  route-policy PASSALL in
  route-policy PASSALL out
```

**IOS**

```
router bgp 100
  neighbor 10.12.1.2 remote-as 200
  !
  address-family ipv4
    neighbor 10.12.1.2 activate
    neighbor 10.12.1.2 maximum-prefix 2
  exit-address-family
```

[Example 14-43](#) displays that the 10.12.1.2 neighbor has exceeded the maximum prefix threshold and shutdown the BGP session.

**Example 14-43** *Maximum Prefix Violation*

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast summary
```

Neighbor	Spk	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	St/PfxRcd
10.12.1.2	0	100	36	29	0	0	0	00:00:15	Idle (PfxCt)

```
RP/0/0/CPU0:XR1#show log | i BGP
```

```
RP/0/0/CPU0:Apr 16 00:55:45.368 : bgp[1046]: %ROUTING-BGP-5-MAXPFX : No. of IPv4
Unicast prefixes received from 10.12.1.2 has reached 2, max 2
RP/0/0/CPU0:Apr 16 00:55:45.369 : bgp[1046]: %ROUTING-BGP-4-MAXPFXEXCEED : No. of
IPv4 Unicast prefixes received from 10.12.1.2: 3 exceed limit 2
RP/0/0/CPU0:Apr 16 00:55:45.370 : bgp[1046]: %ROUTING-BGP-5-ADJCHANGE : neighbor
10.12.1.2 Down - Peer exceeding maximum prefix limit (CEASE notification sent -
maximum number of prefixes reached) (VRF: default)
```

[Example 14-44](#) demonstrates that IOS XR reports the number of maximum prefixes that can be received when displaying the status of BGP neighbors.

**Example 14-44** *Maximum Prefix Violation*

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast neighbors 10.12.1.2 | i prefixes
! Output omitted for brevity
Maximum prefixes allowed 2
Threshold for warning message 75%, restart interval 0 min
```

## REMOVE PRIVATE AUTONOMOUS SYSTEM

Some organizations may not be able to meet the qualifications for obtaining their own ASN but still want to receive Internet routing tables from their service provider. In these situations, the service provider may assign the organization a private ASN for peering. Private ASNs should not be advertised by the service provider to other ISPs on the Internet.

The feature remove private autonomous system removes the private autonomous system of routes that are advertised to the configured peer. The router performs the following path analysis with the remove private autonomous system feature:

- Only removes private ASNs on routes advertised to eBGP peers.
- If the AS\_Path for the route only has private ASNs, the private ASNs are removed.
- If the AS\_Path for the route has a private ASN between public ASNs, it is assumed that this is a design choice, and the private ASN is not removed.
- If the AS\_Path contains confederations (AS\_CONFED\_SEQ), BGP removes the private ASNs only if they are included after the AS\_CONFED\_SEQ (confederation AS\_Path) of the path.

The remove private autonomous system feature is configured on IOS routers with the BGP address family configuration command **neighbor ip-address remove-private-as**, and IOS XR routers use the BGP neighbor address family configuration command **remove-private-as**.

### Note

Starting in Cisco IOS 15.1(2), functionality was added to the IOS trains that allows for the private ASN removed regardless of its location in the AS\_Path. The new configuration command is **neighbor ip-address remove-private-as [all [replace-as]]**. The **all** keyword removes the private ASNs regardless of the location in the AS\_Path, and the keyword **replace-as** replaces private ASNs with the router's local ASN.

Figure 14-18 demonstrates the removal of private ASNs between eBGP sessions:

- XR1 and R4 use private ASNs.
- XR1 advertises the 192.168.1.1/32 network and R4 advertises the 192.168.4.4/32 network.
- XR2 and R3 must remove the private ASNs before the routes can be advertised between public ASNs.

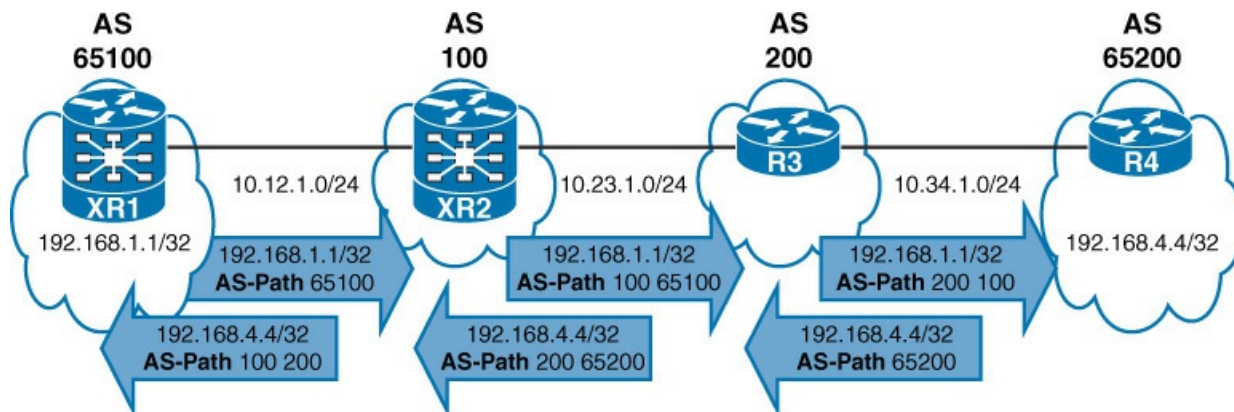


Figure 14-18 Remove Private Autonomous System Topology

Example 14-45 demonstrates the BGP configuration for removing the private ASNs on XR2 and R3. The remove private autonomous system feature is configured for a specific neighbor for each address family.

### Example 14-45 Remove Private Autonomous System BGP Configuration

[Click here to view code image](#)

#### XR2

```
router bgp 100
address-family ipv4 unicast
!
neighbor 10.12.1.1
remote-as 65100
address-family ipv4 unicast
route-policy PASSALL in
route-policy PASSALL out
!
neighbor 10.23.1.3
remote-as 200
address-family ipv4 unicast
route-policy PASSALL in
route-policy PASSALL out
remove-private-AS
```

#### R3

```
router bgp 200
no bgp default ipv4-unicast
neighbor 10.23.1.2 remote-as 100
neighbor 10.34.1.4 remote-as 65200
!
address-family ipv4
neighbor 10.23.1.2 activate
neighbor 10.23.1.2 remove-private-as
neighbor 10.34.1.4 activate
exit-address-family
```

Example 14-46 displays XR2 and R3's BGP table. The 192.168.1.1/32 network prefix contains an AS\_Path of 65100 on XR2, which was successfully removed during XR2's eBGP advertisement to

R3. The 192.168.4.4./32 network prefix contains an AS\_Path of 65200 on R3, which was removed during R3's eBGP advertisement to XR2.

### Example 14-46 XR2's and R3's BGP Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show bgp ipv4 unicast
! Output omitted for brevity
  Network          Next Hop          Metric LocPrf Weight Path
*> 192.168.1.1/32   10.12.1.1         0             0 65100 i
*> 192.168.4.4/32   10.23.1.3         0             0 200 i

Processed 4 prefixes, 4 paths
```

```
R3#show bgp ipv4 unicast
! Output omitted for brevity
  Network          Next Hop          Metric LocPrf Weight Path
*> 192.168.1.1/32   10.23.1.2         0             0 100 i
*> 192.168.4.4/32   10.34.1.4         0             0 65200 i
```

## ALLOW AUTONOMOUS SYSTEM

The allow autonomous system feature enables routes to be received and processed even if the router detects its own ASN in the AS\_Path. As a loop-prevention mechanism, a router will discard BGP network prefixes if it sees its ASN in the AS\_Path. Some network designs use a transit autonomous system to provide connectivity to two different locations. BGP will detect the network advertisements from the remote site as a loop and discards the route. The AS\_Path loop check feature needs to be disabled to maintain connectivity in scenarios such as these.

On IOS routers, the command **neighbor ip-address allow-as-in** is placed under the address family, and IOS XR routers use the BGP neighbor address family configuration command **allowas-in**.

Figure 14-19 provides a topology example to demonstrate the usage of the allow autonomous system feature.

- XR1 and R4 use ASN 100, and R2 and R3 use ASN 200.
- XR1 is advertising the 192.168.1.1/32 network and R4 is advertising the 192.168.4.4/32 network.
- Default BGP behavior prevents XR1 from processing R4's routes because the AS\_Path will contain the ASN100, which is detected as a loop. R4 will encounter identical behavior and discard XR1's routes.
- XR1 and R4 may use the allow autonomous system feature so that they can process each other's routes.

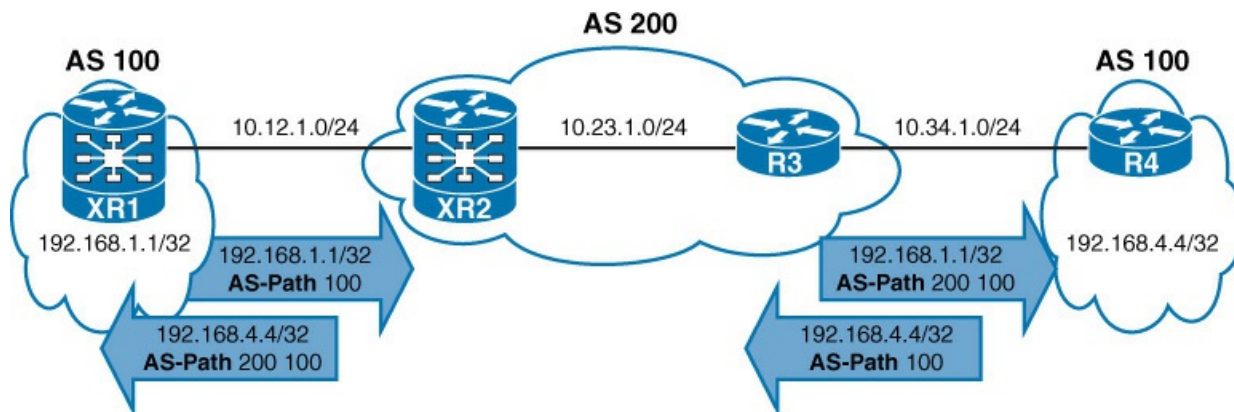


Figure 14-19 Topology with Duplicate Edge ASNs

Example 14-47 demonstrates the BGP configuration of the allow autonomous system feature for XR1 and R4. Notice that the allow autonomous system feature is configured for the neighbor under the appropriate address family.

### Example 14-47 Allow Autonomous System Configuration

[Click here to view code image](#)

```
XR1
router bgp 100
 address-family ipv4 unicast
  network 192.168.1.1/32
  !
 neighbor 10.12.1.2
  remote-as 200
 address-family ipv4 unicast
  route-policy PASSALL in
  allowas-in
  route-policy PASSALL out
```

```
R4
router bgp 100
 no bgp default ipv4-unicast
 neighbor 10.34.1.3 remote-as 200
 !
 address-family ipv4
  network 192.168.4.4 mask 255.255.255.255
  neighbor 10.34.1.3 activate
  neighbor 10.34.1.3 allowas-in
 exit-address-family
```

Example 14-48 displays XR1's and R4's BGP table. XR1 was able to process the 192.168.4.4/32 network prefix even though AS100 was detected in the AS\_Path, and R4 was able to process the 192.168.1.1/32 network prefix with the AS100 in the AS\_Path.

### Example 14-48 Verification of the Allow Autonomous System Feature

[Click here to view code image](#)



```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

```
*> 192.168.1.1/32      0.0.0.0          0          32768 i
*> 192.168.4.4/32      10.12.1.2        0 200 100 i
```

```
Processed 2 prefixes, 2 paths
```

```
R4#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 192.168.1.1/32	10.34.1.3			0 200	100 i
*> 192.168.4.4/32	0.0.0.0	0		32768	i

#### Note

If XR1 or R4 did not enable the `allow autonomous system` feature, the bandwidth and processing by AS200 routers would have been wasted. IOS XR routers perform a loop check as part of the advertisement process to avoid sending updates to peers that would fail the loop check. To disable this feature the BGP address family configuration command `as-path-loopcheck out disable` is configured on XR2.

## LOCAL AUTONOMOUS SYSTEM

When two companies merge, one of the ASNs is usually returned to the regional Internet Registry (RIR). During the migration, each company needs to maintain their own ASN while changes are made with their peering neighbors to update their configuration.

The local autonomous system feature is configured on a per-peer basis, and allows for BGP sessions to establish using an alternate ASN from the ASN that the BGP process is running on. The local autonomous system feature works only with eBGP peerings

IOS routers use the BGP address family neighbor configuration command **neighbor ip-address local-as alternate-as-number [no-prepend [replace-as [dual-as]]]**, and IOS XR routers use the equivalent command **local-as alternate-as-number [no-prepend [replace-as [dual-as]]]**. By default, the alternate ASN is added to the AS\_Path for routes that are sent and received between these two peers.

Figure 14-20 demonstrates the use of the local autonomous system feature. The following logic is applied to this scenario:

- AS100 (XR1) has acquired AS200 (XR2) and set up a direct link 10.12.1.0/24.
- AS 400 (R4) has acquired AS 500 (R5) and setup a direct link to 10.45.1.0/24.
- AS300 (R3) cannot make the appropriate BGP remote autonomous system changes for XR2 and R5's connection for another 2 months.
- XR2 and R5 may use the local autonomous system feature to overcome AS300's configuration delay while allowing them to reconfigure their routers using the ASN of the acquiring company.

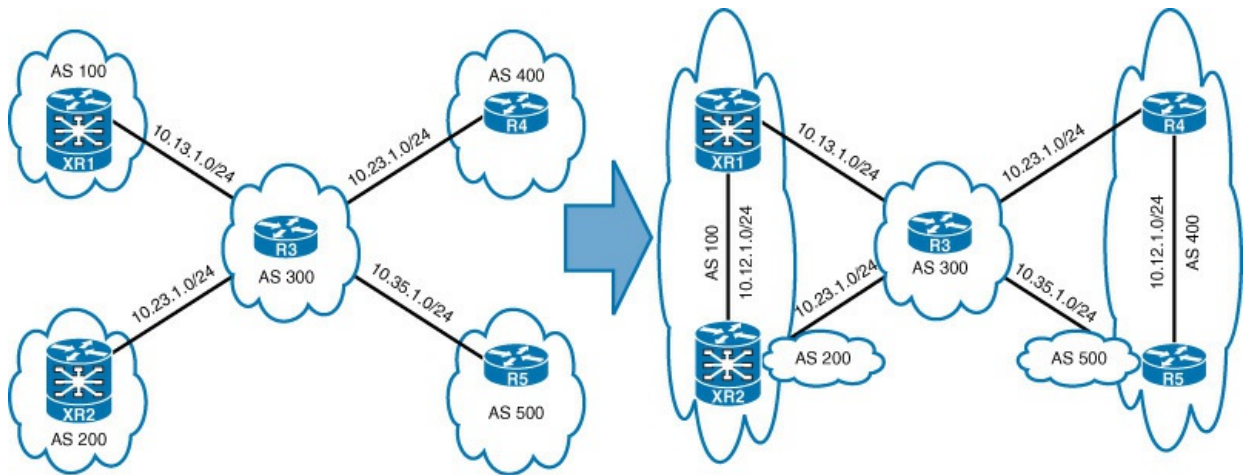


Figure 14-20 Local Autonomous System Topology

Example 14-49 displays the BGP configuration for XR1 and R4 to demonstrate the BGP peering with the acquired router. The only change on these routers involves the additional BGP peering of the acquired router. No configuration changes are necessary on R3.

### Example 14-49 XR1 and R4 Configuration

[Click here to view code image](#)

```

XR1
router bgp 100
 address-family ipv4 unicast
   network 192.168.1.1/32
   !
 neighbor 10.12.1.2
   remote-as 100
   address-family ipv4 unicast
     next-hop-self
   !
 !
 neighbor 10.13.1.3
   remote-as 300
   address-family ipv4 unicast
     route-policy PASSALL in
     route-policy PASSALL out

 neighbor 10.13.1.3
   remote-as 300
   address-family ipv4 unicast
     route-policy PASSALL in
     route-policy PASSALL out

```

**R4**

```
router bgp 400
  no bgp default ipv4-unicast
  neighbor 10.34.1.3 remote-as 300
  neighbor 10.45.1.5 remote-as 400
  !
  address-family ipv4
    network 192.168.4.4 mask 255.255.255.255
    neighbor 10.34.1.3 activate
    neighbor 10.45.1.5 activate
    neighbor 10.45.1.5 next-hop-self
  exit-address-family
```

**Example 14-50** demonstrates the local autonomous system feature in the BGP configuration of XR2 and R5. XR2 now uses the ASN of 100, and R5 uses the ASN of 400. XR2 uses the local autonomous system function with the original AS of 200 for R3's BGP session, and R5 uses the local autonomous system function with the original autonomous system of 500 for R3's BGP session.

**Example 14-50 XR1's and R4's Configuration**

[Click here to view code image](#)

**XR2**

```
router bgp 100
  bgp router-id 192.168.2.2
  address-family ipv4 unicast
    network 192.168.2.2/32
  !
  neighbor 10.12.1.1
    remote-as 100
    address-family ipv4 unicast
      next-hop-self
  !
  !
  neighbor 10.23.1.3
    remote-as 300
    local-as 200
  address-family ipv4 unicast
    route-policy PASSALL in
    route-policy PASSALL out
```

## R5

```
router bgp 400
no bgp default ipv4-unicast
neighbor 10.35.1.3 remote-as 300
neighbor 10.35.1.3 local-as 500
neighbor 10.45.1.4 remote-as 400
!
address-family ipv4
network 192.168.5.5 mask 255.255.255.255
neighbor 10.35.1.3 activate
neighbor 10.45.1.4 activate
neighbor 10.45.1.4 next-hop-self
exit-address-family
```

**Example 14-51** displays R3's BGP table. Notice that the XR2's advertisements for the 192.168.1.1/32 and 192.168.2.2/32 network contain the AS\_Path of 200 100 even though R3 still uses AS200 for establishing a connection. R5's advertisements for the 192.168.4.4/32 and 192.168.5.5/32 network contain the AS\_Path of 500 400 even though R3 still uses AS500 for the BGP session.

### Example 14-51 R3's BGP Table

[Click here to view code image](#)

```
R3#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

	Network	Next Hop	Metric	LocPrf	Weight	Path
*	192.168.1.1/32	10.23.1.2			0	200 100 i
*>		10.13.1.1	0		0	100 i
*>	192.168.2.2/32	10.13.1.1			0	100 i
*		10.23.1.2	0		0	200 100 i
*>	192.168.3.3/32	0.0.0.0	0		32768	i
*	192.168.4.4/32	10.35.1.5			0	500 400 i
*>		10.34.1.4	0		0	400 i
*	192.168.5.5/32	10.35.1.5	0		0	500 400 i
*>		10.34.1.4			0	400 i

One problem with the alternate ASN being prepended when receiving the routes is that other iBGP peers will drop the network prefixes as part of a routing loop detection:

- To stop the alternate ASN from being prepended when *receiving routes*, the optional keyword **no-prepend** is used.
- To stop the alternate ASN from being prepended when *sending routes*, the optional keywords **no-prepend replace-as** is used.
- If both **no-prepend replace-as** keywords are used, all routers see the BGP advertisements as if they were running the original autonomous system in the BGP process.

After the remote peer changes the remote autonomous system setting on the BGP configuration,

the **local-as** commands should be removed. If the coordination of maintenance windows cannot occur during the same time, then the **no-prepend replace-as dual-as** optional keywords allows the remote peer to use either ASN for the BGP session. The remote BGP router will peer with the ASN in the router process statement or the alternate ASN in the **local-as** configuration.

## CONFIGURATION SCALABILITY

BGP configurations can become fairly large as features are configured or BGP sessions increase. IOS and IOS XR operating systems provide methods to apply similar configuration to multiple neighbors. This simplifies the configuration from a deployment standpoint and makes the configuration easier to read. These features are commonly used on out-of-band route reflectors because route reflectors (RRs) should not modify routes upon route reflection. This implies that an RR uses the same outbound routing policy.

### IOS Peer Groups

IOS peer groups simplify BGP configuration and reduce system resources (CPU and memory) by grouping BGP peers together into BGP update groups. BGP update groups allow for a router to perform the outbound route policy processing one time and then replicate the update to all the members, versus performing the outbound route policy processing for every router. Because all members in the BGP update group share the same outbound policy, router resources are conserved during outbound route processing.

BGP peer groups were initially required to identify BGP routers to the same outbound policy, but current software releases dynamically generate the BGP update groups. In addition to enhancing router performance, BGP peer groups still simplify the BGP configuration. All peer group settings used the *peer-group-name* in lieu of the *ip-address* field in the **neighbor ip-address** commands. All routers in the peer group are in the same update group and therefore must be of the same session type (iBGP or eBGP).

#### Note

Members of a peer-group can have a unique inbound routing policy.

The peer group is defined with the command **neighbor group-name peer-group** within the global BGP configuration. All BGP parameters are configured using the peer group *group-name* in lieu of the neighbor's *ip-address*. BGP peer IP addresses are linked to the peer group with the command **neighbor ip-address peer-group group-name**. BGP neighbors cannot be activated by peer group name, and must be activated for each address family by the IP address.

**Example 14-52** demonstrates the BGP configuration for an IOS router using a peer group. The configuration establishes a BGP session with the 192.168.2.2, 192.168.3.3, and 192.168.4.4 routers that are in AS100 that will connect to the router's Loopback 0 IP address. In addition, the router will modify the next-hop address to the IP address used to establish the BGP session.

### **Example 14-52** Example Peer Group Configuration

[Click here to view code image](#)

## IOS

```
router bgp 100
  no bgp default ipv4-unicast
  neighbor AS100 peer-group
  neighbor AS100 remote-as 100
  neighbor AS100 update-source Loopback0
  neighbor 192.168.2.2 peer-group AS100
  neighbor 192.168.3.3 peer-group AS100
  neighbor 192.168.4.4 peer-group AS100
  !
  address-family ipv4
    neighbor AS100 next-hop-self
    neighbor 192.168.2.2 activate
    neighbor 192.168.3.3 activate
    neighbor 192.168.4.4 activate
  exit-address-family
```

### IOS Peer Templates

A restriction for BGP peer groups is that it requires all neighbors to have the same outbound routing policy. IOS BGP peer templates allow for a reusable pattern of settings that can be applied as needed in a hierarchical format through inheritance and nesting of templates. If a conflict exists between an inherited configuration and the invoking peer template, the invoking template preempts the inherited value. There are two types of BGP peer templates:

#### ■ **Peer session:** Configuration settings specifically for the BGP session

Peer session template settings are defined with the BGP configuration command **template peer-session *template-name***, and then any BGP session-related configuration commands are entered.

#### ■ **Peer policy:** Configuration settings specifically for the address family policy

Peer policy template **settings** are defined with the BGP configuration command **template peer-policy *template-name***, and then any BGP address family-related configuration commands can be entered.

The nesting of session templates uses the command **inherit peer-session *template-name sequence***, and the nesting of policy templates uses the command **inherit peer-policy *template-name sequence***.

**Example 14-53** demonstrates the configuration of BGP peer templates. The BGP neighbor 10.12.1.2 invokes the TEMPLATE-PARENT-POLICY for address family policy settings. The TEMPLATE-PARENT-POLICY sets the maximum prefix limit to 10 and invokes the TEMPLATE-CHILD-POLICY that applies the inbound route map to FILTERROUTES.

### **Example 14-53** Peer Template Sample Configuration

[Click here to view code image](#)

## IOS

```
router bgp 100
  template peer-policy TEMPLATE-PARENT-POLICY
    route-map FILTERROUTES in
    inherit peer-policy TEMPLATE-CHILD-POLICY 20
  exit-peer-policy
  !
  template peer-policy TEMPLATE-CHILD-POLICY
    maximum-prefix 10
  exit-peer-policy
  !
  bgp log-neighbor-changes
  neighbor 10.12.1.2 remote-as 200
  !
  address-family ipv4
    neighbor 10.12.1.2 activate
    neighbor 10.12.1.2 inherit peer-policy TEMPLATE-PARENT-POLICY
  exit-address-family
```

### Note

A BGP peer can be associated with either a peer group or template, but not both.

## IOS XR Configuration Templates

IOS XR uses three different types of templates to support BGP peer configuration:

- **Session groups:** Configures identical BGP sessions (update source, security)
- **Address family groups:** Configures identical address family settings (route policy, next-hop-self, maximum prefix, and so on)
- **Neighbor groups:** Configures identical BGP session settings and address family settings

A more explicit setting will preempt the setting within the configuration template. The templates are defined with the BGP configuration command **neighbor-group** *neighbor-group-name*, **session-group** *session-group-name*, or **af-group** *af-group-name*. Then the appropriate settings are applied underneath the group itself.

Session groups are linked with the BGP neighbor configuration command **use session-group** *session-group-name*, address family groups use the BGP neighbor address family configuration command **use af-group** *af-group-name*, and neighbor groups use the BGP neighbor configuration command **use neighbor-group** *neighbor-group-name*.

[Example 14-54](#) demonstrates the use of a neighbor group on the left, and the use of an address family group and session group on the right. In both configurations, the router is forming three eBGP sessions with AS200 that authenticate the BGP session, and use the PASSALL route policy for inbound and outbound route processing.

### Example 14-54 IOS XR BGP Group Configuration Example

[Click here to view code image](#)

#### **IOS XR (Neighbor-Groups)**

```
router bgp 200
  address-family ipv4 unicast
  !
  neighbor-group AS100
    remote-as 100
    password CISCO
    address-family ipv4 unicast
      route-policy PASSALL in
      route-policy PASSALL out
    !
  !
  neighbor 192.168.1.1
    use neighbor-group AS100
  !
  neighbor 192.168.2.2
    use neighbor-group AS100
  !
  neighbor 192.168.4.4
    use neighbor-group AS100
```

#### **IOS XR (AF-Groups and Session-Groups)**

```
router bgp 200
  address-family ipv4 unicast
  !
  af-group IPv4-PASSALL
    address-family ipv4 unicast
      route-policy PASSALL in
      route-policy PASSALL out
    !
  session-group AS100
    remote-as 100
    password CISCO
  !
  neighbor 192.168.1.1
    use session-group AS100
    address-family ipv4 unicast
      use af-group IPv4-PASSALL
  !
  neighbor 192.168.2.2
    use session-group AS100
    address-family ipv4 unicast
      use af-group IPv4-PASSALL
  !
  neighbor 192.168.4.4
    use session-group AS100
    address-family ipv4 unicast
      use af-group IPv4-PASSALL
```

## **SUMMARY**

This chapter demonstrated multiple BGP features that allow for the influence and manipulation of routes on a BGP router:

- BGP communities provide additional traffic engineering capabilities by tagging routes that can



be used by downstream routers to influence routing policies. There are two types of BGP communities: private and well-known. Well-known communities directly affect the advertisement of routes to other autonomous systems. Private communities tag routes based on the logic of the autonomous system adding the community.

- BGP route summarization conserves router resources by reducing the BGP table memory consumption and shrinks BGP path computation time because of a smaller BGP table. If all path attributes from the summarized routes are not included in the summary route, the atomic aggregate is set to indicate that a loss of path information has occurred.

- A default route can be installed into the BGP Loc-RIB, or advertised to specific neighbors. Advertising only default routes from a PE to a CE provides the CE connectivity, while the PE maintains a complete routing table.

- Backdoor networks are eBGP-learned networks that are also available via an IGP. Backdoor networks should prefer to use the IGP route but do not because eBGP has a lower AD (20) than an IGP. Configuring a remote network as a backdoor network changes the BGP AD 200 so that IGP can install the route into the RIB.

- ORF allows for a CE router to specify the prefixes that a PE router advertises to it without adding the administrative burden to the service provider.

- Maximum autonomous system and maximum prefix are used to conserve router resources on smaller platforms that do not have that much memory or CPU resources. The maximum autonomous system feature restricts the AS\_Path length of routes. The maximum prefix feature restricts a BGP peer from advertising more prefixes than was agreed upon between the BGP peers.

- Allow autonomous system disables the BGP loop check and enables a router to receive routes with its ASN in the AS\_Path

- Peer groups, peer templates, neighbor groups, address family groups, and session groups provide scalability for BGP configuration.

## REFERENCES IN THIS CHAPTER

Traina, Paul and Ravi Chandra. RFC 1997, *BGP Communities Attribute*. IETF, <http://www.ietf.org/rfc/rfc1997.txt>, August 1996

Dangli, Srihari, Dan Tappan, and Yakov Rekhter. RFC 4360, *BGP Extended Communities Attribute*. IETF, <http://www.ietf.org/rfc/rfc4360.txt>, February 2006

Doyle, Jeff and Jennifer Carrol. *Routing TCP/IP Volume I, 2nd Edition*. Indianapolis: Cisco Press, 2006. Print.

Cisco. Cisco IOS Software Configuration Guides. <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides. <http://www.cisco.com>



## Chapter 15. BGP Best Path Selection

This chapter covers the following topics:

- BGP best path
- BGP multipath
- Route reflector considerations

In Border Gateway Protocol (BGP), route advertisements consist of the network layer reachability information (NLRI) and the path attributes (PAs). The NLRI consists of the network prefix and prefix length, and the BGP attributes such as AS\_Path, origin, and so on are stored in the PAs. A BGP route may contain multiple paths to the same destination network. Every path's attributes impact the desirability of the route when a router selects the best path. A BGP router will advertise only the best path to the neighboring routers.

Inside of the BGP Loc-RIB table, all the routes and their path attributes are maintained with the best path calculated. The best path is then installed in the RIB of the router. If the best path is no longer available, the router can use the existing paths to quickly identify a new best path. BGP recalculates the best path for a prefix upon four possible events:

- BGP next-hop reachability change
- Failure of an interface connected to an eBGP peer
- Redistribution change
- Reception of new or removed paths for a route

This chapter focuses on the BGP best path algorithm and the impact of that decision on the traffic flows of downstream routers.

### BGP BEST PATH OVERVIEW

The BGP best path selection algorithm influences how traffic enters or leaves an autonomous system. Some router configurations modify the BGP attributes to influence inbound traffic, outbound traffic, or inbound and outbound traffic depending on the network design requirements.

BGP installs the first received path as the best path automatically. When additional paths are received, the newer paths are compared against the current best path. If there is a tie, processing continues onto the next step, until a best path winner is identified.

The following list provides the attributes that the BGP best path algorithm uses for the best route selection process. These attributes are processed in the order listed:

1. Weight

2. Local preference
3. Local originated (**network** statement, redistribution, aggregation)
4. Accumulated interior gateway protocol (AIGP)
5. Shortest AS\_Path
6. Origin type
7. Lowest MED
8. eBGP over iBGP
9. Lowest IGP next hop
10. If both paths are external (eBGP), prefer the first (oldest)
11. Prefer the route that comes from the BGP peer with the lower RID
12. Prefer the route with the minimum cluster list length
13. Prefer the path that comes from the lowest neighbor address

Note

All BGP prefixes must pass the route validity check and the next-hop IP address must be resolvable for the route to be eligible as a best path. Some vendors and publications consider this the first step.

The best path algorithm is explained in the following sections by modifying various path attributes on edge routers to demonstrate the techniques used to manipulate traffic flow into, out of, and around an autonomous system. The route maps and route policies conditionally match against destination network's in this chapter's examples but commonly match on any combination of BGP attributes, such as AS\_Path length, a specific autonomous system number (ASN) via regex patterns, or BGP communities in real-world production networks.

BGP path attribute classifications were explained earlier in Chapter 10, “Border Gateway Protocol (BGP).” Table 15-1 displays which BGP attributes must be supported by all BGP implementations and which BGP attributes are advertised between autonomous systems.

<b>Name</b>	<b>Supported by All BGP Implementations</b>	<b>Advertised Between Autonomous Systems</b>
Well-known mandatory	Yes	Yes
Well-known discretionary	Yes	No
Optional transitive	No	Yes
Optional nontransitive	No	No

**Table 15-1** Path Attribute Classifications

### Weight

BGP weight is a Cisco-defined attribute and the first step for selecting the BGP best path. Weight is a 16-bit value (0–65,535) assigned locally on the router and is not advertised to other routers. The path with the higher weight is preferred. Weight is not advertised to peers and influences only outbound traffic from a router or autonomous system. Because it is the first step in the best path algorithm, it should be used when other attributes should not influence the best path for a specific network.

The command **set weight weight** in a route map or route policy sets the weight value for a prefix. The weight can be set for all prefixes received by a neighbor using the command **neighbor ip-address weight weight** on IOS routers, and the command **weight weight** on IOS XR routers.

Figure 15-1 demonstrates the weight attribute and its influence on the BGP best path algorithm:

- XR4, XR5 and R6 are in AS400, with AS200 and AS300 providing transit connectivity to AS100.
- XR4 is an edge router for AS400 and sets the weight to 222 for the 172.24.0.0/24 prefix received from R2. This ensures that XR4 uses R2 for outbound traffic to this prefix.
- R6 is an edge router for AS400 and sets the weight to 333 for the 172.16.0.0/24 prefix received from R3. This ensures that R6 uses R3 for outbound traffic to this prefix.

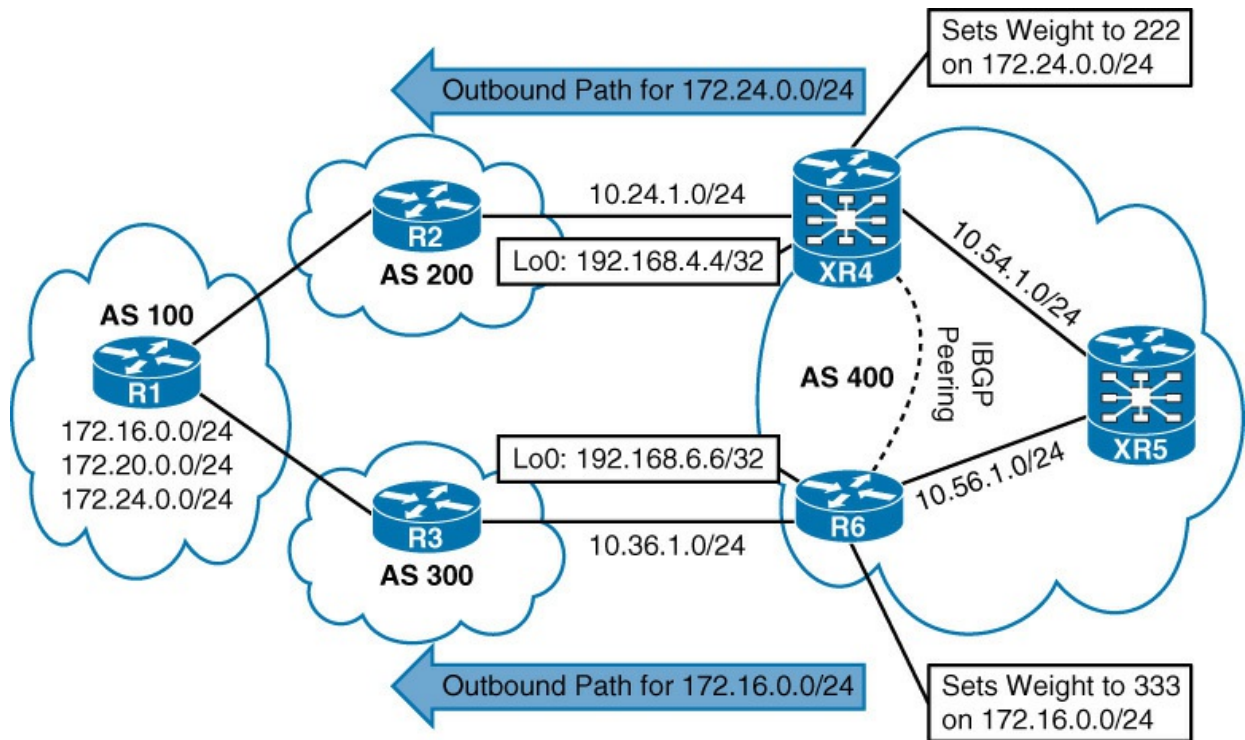


Figure 15-1 BGP Weight Topology

Example 15-1 demonstrates the BGP configuration for manipulating weight on XR4 and R6. Notice that XR4 uses neighbor groups and R6 uses peer groups to minimize the configuration for the other AS400 routers.

**Example 15-1 BGP Weight Manipulation Configuration**

[Click here to view code image](#)

**XR4**

```
router bgp 400
  address-family ipv4 unicast
  !
  neighbor-group AS400
    remote-as 400
    update-source Loopback0
    address-family ipv4 unicast
      next-hop-self
    !
  !
  neighbor 10.24.1.2
    remote-as 200
    address-family ipv4 unicast
      route-policy AS200 in
  !
  neighbor 192.168.5.5
    use neighbor-group AS400
  !
  neighbor 192.168.6.6
    use neighbor-group AS400
  !
route-policy AS200
  if destination in (172.24.0.0/24) then
    set weight 222
  endif
  pass
end-policy
```

**R6**

```
router bgp 400
  no bgp default ipv4-unicast
  neighbor AS400 peer-group
  neighbor AS400 remote-as 400
  neighbor AS400 update-source Loopback0
  neighbor 10.36.1.3 remote-as 300
  neighbor 192.168.4.4 peer-group AS400
  neighbor 192.168.5.5 peer-group AS400
  !
  address-family ipv4
    neighbor AS400 next-hop-self
    neighbor 10.36.1.3 activate
    neighbor 10.36.1.3 route-map AS300 in
    neighbor 192.168.4.4 activate
    neighbor 192.168.5.5 activate
  exit-address-family
  !
  ip prefix-list PRE172 permit 172.16.0.0/24
  !
  route-map AS300 permit 10
    match ip address prefix-list PRE172
    set weight 333
  route-map AS300 permit 20
```

**Example 15-2** displays the BGP table for XR4, XR5, and R6. Notice that the weight is set only

locally on XR4 and R6. The weight was not advertised to any of the AS400 routers, and is set to 0 for all other prefixes. The > indicates the best path.

### Example 15-2 XR4, XR5, and R6 BGP Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast
! Output omitted for brevity

Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.0.0/24  10.24.1.2         0      100     0 200 100 i
* i              192.168.6.6      0      100     0 300 100 i
*> 172.20.0.0/24  10.24.1.2         0      100     0 200 100 i
* i              192.168.6.6      0      100     0 300 100 i
*> 172.24.0.0/24  10.24.1.2         222    100     0 200 100 i
* i              192.168.6.6      0      100     0 300 100 i

Processed 3 prefixes, 6 paths
```

```
RP/0/0/CPU0:XR5#show bgp ipv4 unicast
! Output omitted for brevity

Network          Next Hop          Metric LocPrf Weight Path
* i172.16.0.0/24  192.168.4.4      100     100     0 200 100 i
*>i              192.168.6.6      0      100     0 300 100 i
* i172.20.0.0/24  192.168.4.4      100     100     0 200 100 i
*>i              192.168.6.6      0      100     0 300 100 i
* i172.24.0.0/24  192.168.4.4      100     100     0 200 100 i
*>i              192.168.6.6      0      100     0 300 100 i

Processed 3 prefixes, 6 paths
```

```
R6#show bgp ipv4 unicast
! Output omitted for brevity

Network          Next Hop          Metric LocPrf Weight Path
* i172.16.0.0/24  192.168.4.4      100     100     0 200 100 i
*>                10.36.1.3        333    100     0 300 100 i
* i172.20.0.0/24  192.168.4.4      100     100     0 200 100 i
*>                10.36.1.3        0      100     0 300 100 i
* i172.24.0.0/24  192.168.4.4      100     100     0 200 100 i
*>                10.36.1.3        0      100     0 300 100 i
```

Table 15-2 explains the processing logic for the BGP best path for XR4, XR5, and R6 as the routers process the route updates.



Phase	Device	Processes
Phase 1	XR4	<ul style="list-style-type: none"> <li>■ XR4 receives the path for 172.24.0.0/24 from R2 and modifies the weight to 222.</li> <li>■ XR4 receives the 172.16.0.0/24 and 172.20.0.0/24 prefixes from R2 in unmodified state.</li> <li>■ Only the paths from R2 exist for each prefix, so those paths are marked as the best path.</li> <li>■ XR4 advertises these paths to XR5 and R6. (Weight is not advertised.)</li> </ul>
	R6	<ul style="list-style-type: none"> <li>■ R6 receives the path for 172.16.0.0/24 from R3 and modifies the weight to 333.</li> <li>■ R6 receives the 172.24.0.0/24 and 172.20.0.0/24 prefixes from R3 in unmodified state.</li> <li>■ Only the paths from R3 exist for each prefix, so those paths are marked as the best path.</li> <li>■ R6 advertises these paths to XR4 and XR5. (Weight is not advertised.)</li> </ul>
Phase 2	XR4	<ul style="list-style-type: none"> <li>■ XR4 receives R6's paths for all three prefixes.</li> <li>■ XR4 detects that the 172.16.0.0/24 and 172.20.0.0/24 paths from R6 have the same weight (0) as the paths via R2. Because of the tie, the best path is selected using steps after weight in the best path algorithm.</li> <li>■ XR4 selects R2's path as the best path for 172.24.0.0/24 prefix because the weight of 222 is higher than R6's path that has a weight of 0.</li> </ul>
	XR5	<ul style="list-style-type: none"> <li>■ XR5 receives paths for all network prefixes from XR4 and R6.</li> <li>■ Weight is not passed to XR5 and therefore selects the best path for the prefixes using steps after weight in the best path algorithm.</li> </ul>
	R6	<ul style="list-style-type: none"> <li>■ R6 receives XR4's path advertisements for all three prefixes.</li> <li>■ R6 selects R3's path as the best path for the 172.16.0.0/24 prefix because the weight of 333 is higher than XR4's path that has a weight of 0.</li> <li>■ R6 detects that the 172.24.0.0/24 and 172.20.0.0/24 paths from XR4 have the same weight (0) as the paths via R3. Because of the tie, the best path is selected using steps after weight in the best path algorithm.</li> </ul>

**Table 15-2** BGP Weight Processing Logic for XR4, XR5, and R6

Example 15-3 displays R6's path information for the 172.16.0.0/24 network prefix. Notice that there are multiple paths and that the best path is via R3. The **show bgp ipv4 unicast network** command is extremely helpful for viewing and comparing BGP path attributes.

### **Example 15-3** BGP Table Output for 172.16.0.0/24

[Click here to view code image](#)

```

R6#show bgp ipv4 unicast 172.16.0.0/24
BGP routing table entry for 172.16.0.0/24, version 5
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
    2
! Path #1
  Refresh Epoch 1
  200 100
    192.168.4.4 (metric 12) from 192.168.4.4 (192.168.4.4)
      Origin IGP, localpref 100, valid, internal
! Path #2
  Refresh Epoch 2
  300 100
    10.36.1.3 from 10.36.1.3 (192.168.3.3)
      Origin IGP, localpref 100, weight 333, valid, external, best

```

If an IOS router has multiple paths for a specific BGP route, the command **show bgp ipv4 unicast network bestpath** displays only the path information for the best path. [Example 15-4](#) displays the BGP best path for the 172.16.0.0/24 network.

#### Example 15-4 IOS Best Path

[Click here to view code image](#)

```

R6#show bgp ipv4 unicast 172.16.0.0/24 bestpath
BGP routing table entry for 172.16.0.0/24, version 5
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
    2
  Refresh Epoch 2
  300 100
    10.36.1.3 from 10.36.1.3 (100.2.2.254)
      Origin IGP, localpref 100, weight 333, valid, external, best

```

IOS XR provides similar functionality with the command **show bgp ipv4 unicast network bestpath-compare**. In addition, the output will explain why a path was not selected as the best. In [Example 15-5](#), path 2 is not the best path because it has a lower weight than path 1.

#### Example 15-5 IOS XR Best Path Compare

[Click here to view code image](#)

```

RP/0/0/CPU0:XR4#show bgp ipv4 unicast 172.24.0.0/24 bestpath-compare
BGP routing table entry for 172.24.0.0/24
Paths: (2 available, best #1)
  Path #1: Received by speaker 0
    200 100
    10.24.1.2 from 10.24.1.2 (100.1.1.254)
      Origin IGP, localpref 100, weight 444, valid, external, best, group-best
      Origin-AS validity: not-found
      best of AS 200, Overall best
  Path #2: Received by speaker 0
    300 100
    192.168.6.6 (metric 3) from 192.168.6.6 (192.168.6.6)
      Origin IGP, metric 0, localpref 100, valid, internal, group-best
      best of AS 300
      Lower weight than best path (path #1)

```

## Local Preference

Local preference (LOCAL\_PREF) is a well-known discretionary path attribute and is included with path advertisements throughout an autonomous system. The local preference attribute is a 32-bit value (0–4,294,967,295) that indicates the preference for exiting the autonomous system to the destination network. A higher value is preferred over a lower value. The local preference is not advertised between external BGP (eBGP) peers and is typically used to influence the next-hop address for outbound traffic (that is, leaving an autonomous system).

Setting the local preference for specific routes can be accomplished via a route map or route policy with the action **set local-preference preference**. Local preference for all routes received by a neighbor can be set with the BGP neighbor address family configuration command **neighbor ip-address local-preference preference** on IOS routers or the IOS XR equivalent command **local-preference preference**.

### Note

If an edge BGP router does not define the local preference upon receipt of a prefix, the default local preference value of 100 is used during best path calculation and is included in advertisements to other iBGP peers.

**Figure 15-2** demonstrates modifying the local preference to influence the traffic flow for prefix 172.24.0.0/24 and 172.16.0.0/24.

- XR4, XR5, and R6 are in AS400, with AS200 and AS300 providing transient connectivity to AS100.
- XR4 is an edge router for AS400 and sets the local preference to 222 for the 172.24.0.0/24 prefix received from R2, making it the preferred path for AS400.
- R6 is an edge router for AS400 and sets the local preference to 333 for the 172.16.0.0/24 prefix received from R3, making it the preferred path for AS400.

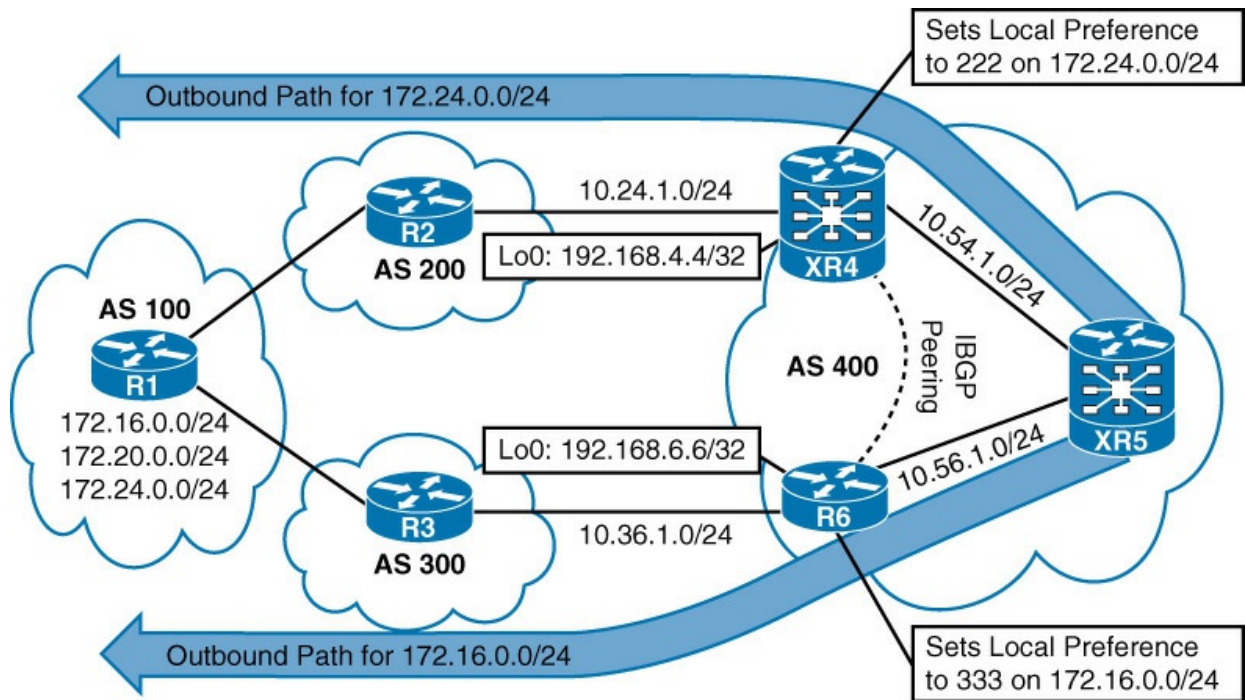


Figure 15-2 BGP Local Preference Topology

Example 15-6 demonstrates the BGP configuration for modifying the local preference on XR4 and R6.

#### Example 15-6 BGP Local Preference Configuration

[Click here to view code image](#)

**XR4**

```
router bgp 400
  address-family ipv4 unicast
  !
  neighbor-group AS400
    remote-as 400
    update-source Loopback0
    address-family ipv4 unicast
    next-hop-self
  !
  neighbor 10.24.1.2
    remote-as 200
    address-family ipv4 unicast
    route-policy AS200 in
  !
  neighbor 192.168.5.5
    use neighbor-group AS400
  !
  neighbor 192.168.6.6
    use neighbor-group AS400
  !
  route-policy AS200
    if destination in (172.24.0.0/24) then
      set local-preference 222
    endif
    pass
  end-policy
```

**R6**

```
router bgp 400
  no bgp default ipv4-unicast
  neighbor AS400 peer-group
  neighbor AS400 remote-as 400
  neighbor AS400 update-source Loopback0
  neighbor 10.36.1.3 remote-as 300
  neighbor 192.168.4.4 peer-group AS400
  neighbor 192.168.5.5 peer-group AS400
  !
  address-family ipv4
    neighbor AS400 next-hop-self
    neighbor 10.36.1.3 activate
    neighbor 10.36.1.3 route-map AS300 in
    neighbor 192.168.4.4 activate
    neighbor 192.168.5.5 activate
  exit-address-family
  !
  ip prefix-list PRE172 permit 172.16.0.0/24
  !
  route-map AS300 permit 10
    match ip address prefix-list PRE172
    set local-preference 333
  route-map AS300 permit 20
```

**Example 15-7** displays the BGP table for XR4, XR5, and R6. All three AS400 routers will send

traffic toward the 172.16.0.0/24 network through R6's link to R3, and all three AS400 routers will send traffic toward the 172.24.0.0/24 network through XR4's link to R2. The 172.20.0.0/24 uses logic in a later step of the BGP best path algorithm. Notice that the default local preference of 100 is advertised to the internal BGP (iBGP) routers for the other prefixes.

XR4 has only one path for the 172.24.0.0/24 network, R6 has only one path for the 172.16.0.0/24 network, and XR5 has only one path for the 172.16.0.0/24 and 172.24.0.0/24 network.

### Example 15-7 XR4, XR5, and R6 BGP Table After Local Preference Modification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
* 172.16.0.0/24	10.24.1.2			0 200	100 i
*>i	192.168.6.6	0	333	0 300	100 i
*> 172.20.0.0/24	10.24.1.2			0 200	100 i
* i	192.168.6.6	0	100	0 300	100 i
*> 172.24.0.0/24	10.24.1.2		222	0 200	100 i

```
Processed 3 prefixes, 5 paths
```

```
RP/0/0/CPU0:XR5#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i172.16.0.0/24	192.168.6.6	0	333	0 300	100 i
*>i172.20.0.0/24	192.168.4.4		100	0 200	100 i
* i	192.168.6.6	0	100	0 300	100 i
*>i172.24.0.0/24	192.168.4.4		222	0 200	100 i

```
Processed 3 prefixes, 4 paths
```

```
R6#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/24	10.36.1.3		333	0 300	100 i
* i172.20.0.0/24	192.168.4.4		100	0 200	100 i
*>	10.36.1.3			0 300	100 i
*>i172.24.0.0/24	192.168.4.4		222	0 200	100 i
*	10.36.1.3			0 300	100 i

A router will not discard a path that is not chosen as the best path. The path is always maintained in the BGP Loc-RIB database and could be used later in the event the best path is no longer available. If a router identifies a different best path from the one advertised, it will withdraw its previous best path advertisement and advertise the new best path.

Note

BGP routers do not advertise iBGP-learned routes to other iBGP peers unless the router is a route reflector.

Table 15-3 describes the processing logic for the BGP best path algorithm for XR4, XR5, and R6.

---

**Phase Device Processes**

---

Phase 1	XR4	<ul style="list-style-type: none"><li>■ XR4 receives the prefix for 172.24.0.0/24 from R2 and sets the local preference to 222.</li><li>■ XR4 receives the 172.16.0.0/24 and 172.20.0.0/24 prefixes from R2.</li><li>■ No other paths exist for these prefixes, so all paths are marked as best paths.</li><li>■ XR4 advertises these paths to XR5 and R6.</li><li>■ (Routes without local preference set, are advertised with the local preference of 100.)</li></ul>
	R6	<ul style="list-style-type: none"><li>■ R6 receives the prefix for 172.16.0.0/24 from R3 and sets the local preference to 333.</li><li>■ R6 receives the 172.24.0.0/24 and 172.20.0.0/24 prefixes from R3.</li><li>■ No other paths exist for these prefixes, so all paths are marked as best paths.</li><li>■ R6 advertises these paths to XR4 and XR5.</li><li>■ (Routes without local preference set are advertised with the local preference of 100.)</li></ul>
Phase 2	XR4	<ul style="list-style-type: none"><li>■ XR4 receives R6's paths for all the prefixes from R3.</li><li>■ XR4 detects that the 172.16.0.0/24 path from R6 has a higher local preference than the path from R2. XR4 marks the path from R6 as the best path for the prefix and sends route withdrawals to XR5 and R6 for the path from R2.</li><li>■ XR4 detects that the 172.20.0.0/24 path from R3 has the same local preference as the path from R2. (Routes without local preference use the default value of 100.) Because of the tie, the best path is selected using steps after local preference in the best path algorithm.</li><li>■ XR4 detects that the 172.24.0.0/24 path from R2 has a higher local preference than the path from R6. XR4 keeps the path from R2 as the best path for the prefix.</li></ul>
	XR5	<ul style="list-style-type: none"><li>■ XR5 receives paths for all network prefixes from XR4 and R6.</li><li>■ XR5 detects that the 172.16.0.0/24 path from R6 has a higher local preference than the path from R2. XR5 marks the path from R3 as the best path for the prefix.</li><li>■ XR5 detects that the 172.20.0.0/24 paths from XR4 and R6 has identical local preference values and uses steps after local preference in the best path algorithm.</li><li>■ XR5 detects that the 172.24.0.0/24 path from R2, has a higher local preference than the path from R6. XR5 selects the path from R2 as the best path for the prefix.</li></ul>
	R6	<ul style="list-style-type: none"><li>■ R6 receives XR4's route advertisement for all the prefixes from R2.</li><li>■ R6 detects that the 172.16.0.0/24 path from R3 has a higher local preference than the path from XR4. R6 keeps the path from R3 as the best path for the prefix.</li><li>■ R6 detects that the 172.20.0.0/24 path from R3 has the same local preference as the path from XR4. (Routes without local preference use the default value of 100.) Because of the tie, the best path is selected using steps after local preference in the best path algorithm.</li><li>■ R6 detects that the 172.24.0.0/24 path from XR4 has a higher local preference than the path from R3. R6 selects the path from XR4 as the best path for the prefix and sends route withdrawals to XR5 and R6 for the paths from R3.</li></ul>
Phase 3	XR4	<ul style="list-style-type: none"><li>■ XR4 receives R6's withdraw for 172.24.0.0/24 path and removes it from the BGP table.</li></ul>
	XR5	<ul style="list-style-type: none"><li>■ XR5 receives XR4's withdraw for 172.16.0.0/24 path and removes it from the BGP table.</li><li>■ XR5 receives R6's withdraw for 172.24.0.0/24 path and removes it from the BGP table.</li></ul>
	R6	<ul style="list-style-type: none"><li>■ R6 receives XR4's withdraw for 172.16.0.0/24 path and removes it from the BGP table.</li></ul>

---

**Table 15-3** BGP Local Preference Processing Logic for XR4, XR5, and R6



### Locally Originated via Network or Aggregate Advertisement

The third decision point in the best path algorithm is to determine whether the route originated locally or was received from a BGP peer. Preference is given in the following order:

1. Routes that were advertised locally
2. Routes that have been aggregated locally
3. Routes received from a BGP peers

Figure 15-3 demonstrates the concept of locally originated route preference. The topology focuses on AS400 with all three routers (XR4, XR5, and R6) peering via loopback addresses in an iBGP full mesh. XR4 and R6 advertise the 192.168.4.4/32 network. XR4 uses the directly connected entry in the Routing Information Base (RIB), and R6 uses the Open Shortest Path First (OSPF) Protocol entry in the RIB for advertisement.

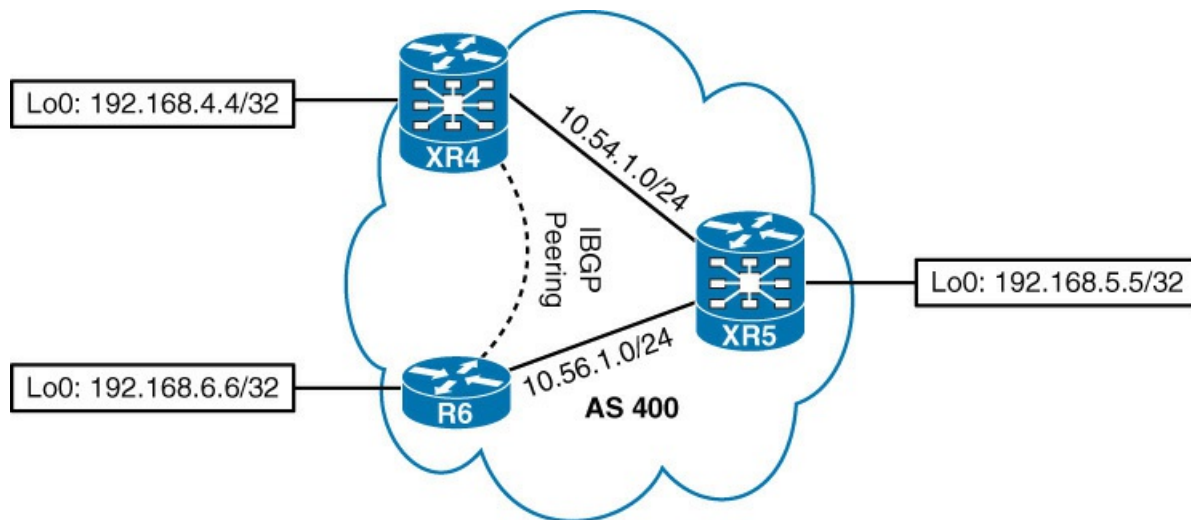


Figure 15-3 AS400 Topology

Example 15-8 demonstrates a basic BGP configuration for XR4 and R6 with the advertisement of the 192.168.4.4/32 network.

### Example 15-8 XR4 and R6 BGP Configuration

[Click here to view code image](#)

**XR4**

```
router bgp 400
  address-family ipv4 unicast
    network 192.168.4.4/32
  !
  neighbor-group AS400
    remote-as 400
    update-source Loopback0
    address-family ipv4 unicast
      next-hop-self
  !
  neighbor 192.168.5.5
    use neighbor-group AS400
  !
  neighbor 192.168.6.6
    use neighbor-group AS400
```

**R6**

```
router bgp 400
  no bgp default ipv4-unicast
  neighbor AS400 peer-group
  neighbor AS400 remote-as 400
  neighbor AS400 update-source Loopback0
  neighbor 192.168.4.4 peer-group AS400
  neighbor 192.168.5.5 peer-group AS400
  !
  address-family ipv4
    network 192.168.4.4 mask 255.255.255.255
    neighbor AS400 next-hop-self

  neighbor 192.168.4.4 activate
  neighbor 192.168.5.5 activate
  exit-address-family
```

**Example 15-9** displays the BGP path information for the 192.168.4.4/32 route. XR4 identifies itself as the best path, and R6 identifies itself as the best path. Notice that the weight is set to 32,768 on the locally originated route of both routers.

**Example 15-9** *192.168.4.4/32 Path Information*

[Click here to view code image](#)

```

RP/0/0/CPU0:XR4#show bgp ipv4 unicast 192.168.4.4
BGP routing table entry for 192.168.4.4/32
Paths: (2 available, best #1)
  Advertised to update-groups (with more than one peer):
    0.2
  Path #1: Received by speaker 0
  Advertised to update-groups (with more than one peer):
    0.2
  Local
    0.0.0.0 from 0.0.0.0 (192.168.4.4)
      Origin IGP, metric 0, localpref 100, weight 32768, valid, local, best,
group-best
      Received Path ID 0, Local Path ID 1, version 14
  Path #2: Received by speaker 0
  Not advertised to any peer
  Local
    192.168.6.6 (metric 3) from 192.168.6.6 (192.168.6.6)
      Origin IGP, metric 12, localpref 100, valid, internal
      Received Path ID 0, Local Path ID 0, version 0

```

```

R6#show bgp ipv4 unicast 192.168.4.4
BGP routing table entry for 192.168.4.4/32, version 8
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
    2
  Refresh Epoch 1
  Local
    192.168.4.4 (metric 12) from 192.168.4.4 (192.168.4.4)
      Origin IGP, metric 0, localpref 100, valid, internal
  Refresh Epoch 1
  Local
    10.56.1.5 from 0.0.0.0 (192.168.6.6)
      Origin IGP, metric 12, localpref 100, weight 32768, valid, sourced, local,
best

```

### Accumulated Interior Gateway Protocol

The accumulated interior gateway protocol (AIGP) is an optional nontransitive path attribute that is included with advertisements throughout an autonomous system. IGP's typically use the lowest path metric to identify the shortest path to a destination but cannot provide the scalability of BGP. BGP uses an autonomous system to identify a single domain of control for a routing policy. BGP does not use path metric due to scalability issues combined with the notion that each autonomous system may use a different routing policy to calculate metrics.

AIGP provides the ability for BGP to maintain and calculate a conceptual path metric in environments that use multiple autonomous systems with unique IGP routing domains in each autonomous system. The capability for BGP to make routing decisions based on a path metric is a viable option because all the autonomous systems are under the control of a single domain with consistent routing policies for BGP and IGP protocols.

#### Note

The IGP routing protocol should remain consistent between the routing domains. Every IGP routing protocol path metric has a different default value. For example, the Enhanced Interior Gateway Routing Protocol (EIGRP) path metric is much higher than OSPF's.

In Figure 15-4, AS100, AS200, and AS300 are all under the control of the same service provider. AIGP has been enabled on the BGP sessions between all the routers, and the IGP protocols are redistributed into BGP. The AIGP metric is advertised between AS100, AS200, and AS300, allowing BGP to use the AIGP metric for best path calculations between the autonomous systems.

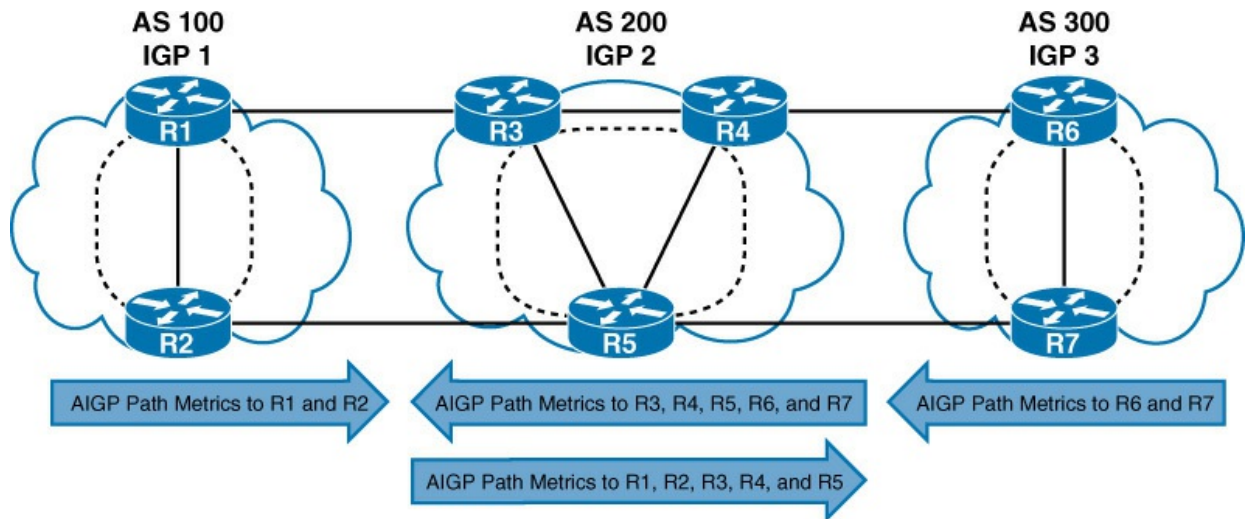


Figure 15-4 AIGP Path Attribute Exchange Between Autonomous Systems

Exchanging AIGP PAs must be agreed upon between the BGP peers and will be included only in prefix advertisements between AIGP-enabled peers. IOS routers enable AIGP metrics for BGP neighbors with the address family configuration command **neighbor ip-address aigp**. IOS XR enables AIGP for all iBGP sessions automatically but requires that configuration for eBGP sessions with the BGP neighbor address family configuration command **aigp**.

The AIGP metric is a 32-bit (0–4,294,967,295) value that can be set during redistribution or during receipt of a prefix with a route map or route policy. Route maps and route policies use the configuration command **set aigp-metric {igp-metric | metric}**. The **igp-metric** keyword sets the value to the IGP path metric on the redistributing router for the specific route. Static routes and network advertisements populate the AIGP metric with the path metric to the next-hop address of the route.

Note

Route maps or route policies must be used to populate the AIGP metric during redistribution.

The following guidelines apply to AIGP metrics:

- A path with an AIGP metric is preferred to a path without an AIGP metric.
- If the next-hop address requires a recursive lookup, the AIGP path needs to calculate a derived metric to include the distance to the next-hop address. This ensures that the cost to the BGP edge router is included. The formula is **Derived AIGP Metric = (Original AIGP Metric + Next-Hop AIGP Metric)**.

- If multiple AIGP paths exist and one next-hop address contains an AIGP metric and the other does not, the non-AIGP path is not used.
- The next-hop AIGP metric is recursively added if multiple lookups are performed.
- AIGP paths are compared based on the derived AIGP metric (with recursive next-hops) or the actual AIGP metric (nonrecursive next hop). The path with the lower AIGP metric is preferred.
- When a router R2 advertises an AIGP-enabled path that was learned from R1, if the next-hop address changes to an R2 address, R2 increments the AIGP metric to reflect the distance (the IGP path metric) between R1 and R2.

Note

It is recommended that BGP best external and additional path features are used in conjunction with AIGP metrics. These topics are covered in detail in Chapter 21, "High Availability."

Figure 15-5 demonstrates the manipulation of the AIGP metrics to influence AS400's outbound traffic:

- XR4, XR5 and R6 are in AS400, with AS200 and AS300 providing transient connectivity to AS100.
- XR4 sets the AIGP metric to 222 for the 172.24.0.0/24 path received from R2, making it the most preferred path for AS400.
- R6 sets the AIGP metric to 333 on for the 172.16.0.0/24 path received from R3, making it the most preferred path for AS400.

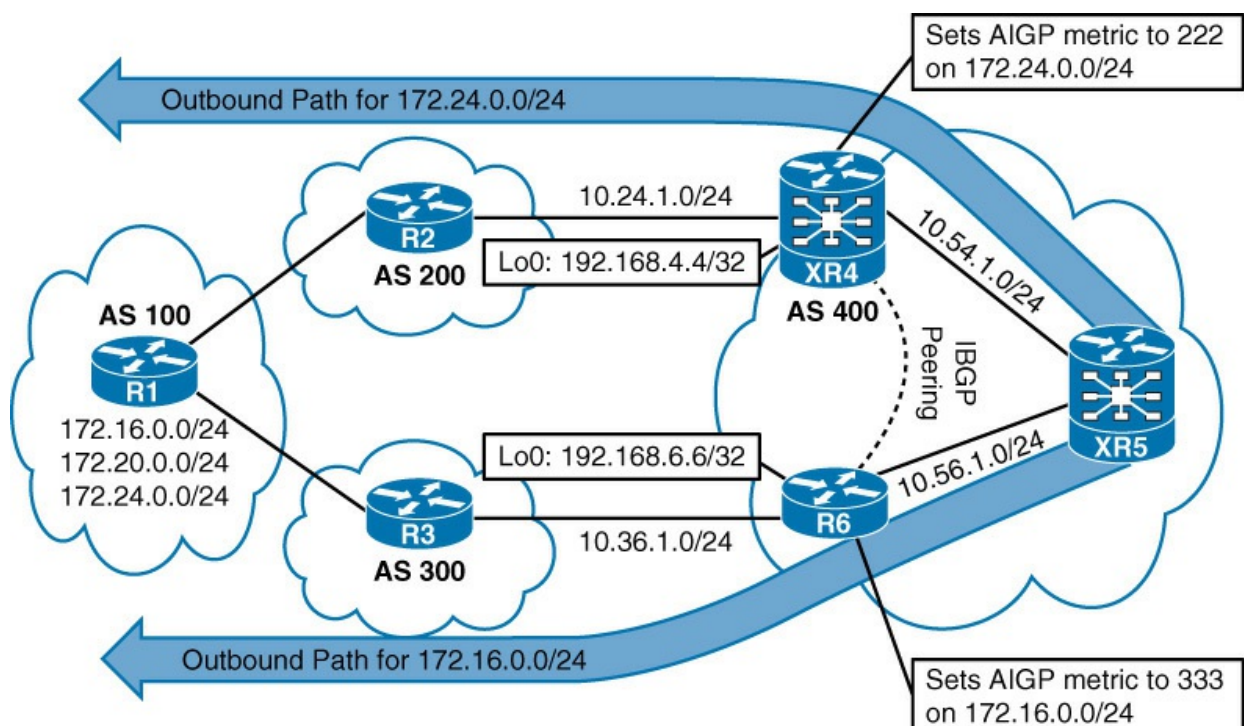


Figure 15-5 AIGP Metric Modification

Example 15-10 demonstrates the AIGP manipulation configuration for XR4 and R6.

### **Example 15-10** *BGP AIGP Metric Manipulation Configuration*

[Click here to view code image](#)

```
XR4
router bgp 400
  address-family ipv4 unicast
  !
  neighbor-group AS400
    remote-as 400
    update-source Loopback0
    address-family ipv4 unicast
      next-hop-self
  !
  neighbor 10.24.1.2
    remote-as 200
    address-family ipv4 unicast
      route-policy AS200 in
  !
  neighbor 192.168.5.5
    use neighbor-group AS400
  !
  neighbor 192.168.6.6
    use neighbor-group AS400
  !
  route-policy AS200
    if destination in (172.24.0.0/24) then
      set aigp-metric 222
    endif
  pass
end-policy
```

## R6

```
router bgp 400
  no bgp default ipv4-unicast
  neighbor AS400 peer-group
  neighbor AS400 remote-as 400
  neighbor AS400 update-source Loopback0
  neighbor 10.36.1.3 remote-as 300
  neighbor 192.168.4.4 peer-group AS400
  neighbor 192.168.5.5 peer-group AS400
  !
  address-family ipv4
    neighbor AS400 aigp
    neighbor AS400 next-hop-self
    neighbor 10.36.1.3 activate
    neighbor 10.36.1.3 route-map AS300 in
    neighbor 192.168.4.4 activate
    neighbor 192.168.5.5 activate
  exit-address-family

  !
ip prefix-list PRE172 permit 172.16.0.0/24
!
route-map AS300 permit 10
  match ip address prefix-list PRE172
  set as-path prepend 333
route-map AS300 permit 20
```

**Example 15-11** confirms that AIGP metrics are enabled between the AS400 routers by examining the neighbor sessions.

### **Example 15-11** Verification of BGP Neighbors Supporting AIGP Metrics

**Click here to view code image**

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast neighbors | i " neighbor is | AIGP"
BGP neighbor is 10.24.1.2
BGP neighbor is 192.168.5.5
  AIGP is enabled
BGP neighbor is 192.168.6.6
  AIGP is enabled
```

```
R6#show bgp ipv4 unicast neighbors | i AS | AIGP
BGP neighbor is 10.36.1.3, remote AS 300, external link
BGP neighbor is 192.168.4.4, remote AS 400, internal link
  AIGP is enabled
BGP neighbor is 192.168.5.5, remote AS 400, internal link
  AIGP is enabled
```

**Example 15-12** displays the BGP table for XR4, XR5, and R6. All three AS400 routers will send traffic toward the 172.16.0.0/24 network through R6's link to R3, and all three AS400 routers will send traffic toward the 172.24.0.0/24 network through XR4's link to R2. The 172.20.0.0/24 uses logic in a later step of the BGP best path algorithm. The AIGP metric is not displayed in the

current output.

### Example 15-12 XR4, XR5, and R6 BGP Table After AIGP Metric Modification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast
! Output omitted for brevity

Network          Next Hop          Metric LocPrf Weight Path
* 172.16.0.0/24  10.24.1.2        0      100      0 200 100 i
*>i              192.168.6.6      0      100      0 300 100 i
*> 172.20.0.0/24  10.24.1.2        0      100      0 200 100 i
* i              192.168.6.6      0      100      0 300 100 i
*> 172.24.0.0/24  10.24.1.2        0      100      0 200 100 i

Processed 3 prefixes, 5 paths
```

```
RP/0/0/CPU0:XR5#show bgp ipv4 unicast
! Output omitted for brevity

Network          Next Hop          Metric LocPrf Weight Path
*>i172.16.0.0/24  192.168.6.6      0      100      0 300 100 i
*>i172.20.0.0/24  192.168.4.4      100     0 200 100 i
* i              192.168.6.6      0      100      0 300 100 i
*>i172.24.0.0/24  192.168.4.4      100     0 200 100 i

Processed 3 prefixes, 4 paths
```

```
R6#show bgp ipv4 unicast
! Output omitted for brevity

Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.0.0/24  10.36.1.3        0      100      0 300 100 i
* i 172.20.0.0/24  192.168.4.4      100     0 200 100 i
*>              10.36.1.3        0      100      0 300 100 i
* 172.24.0.0/24  10.36.1.3        0      100      0 300 100 i
*>i              192.168.4.4      100     0 200 100 i
```

Example 15-13 displays the AIGP path attributes on XR4 and R6 for the 172.24.0.0/24 route. Notice that XR4 displays that the AIGP metric was set by inbound route policy and provides the total AIGP metric. The total AIGP metric would have incremented if the next hop had been recursive.

### Example 15-13 BGP Path Attributes with AIGP Metric

[Click here to view code image](#)



```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast 172.24.0.0
```

```
! Output omitted for brevity
```

```
BGP routing table entry for 172.24.0.0/24
```

```
Paths: (2 available, best #1)
```

```
Path #1: Received by speaker 0
```

```
200 100
```

```
10.24.1.2 from 10.24.1.2 (192.168.2.2)
```

```
Origin IGP, localpref 100, aigp metric 222, valid, external, best, group-best,  
import-candidate
```

```
AIGP set by inbound policy metric
```

```
Total AIGP metric 222
```

```
R6#show bgp ipv4 unicast 172.24.0.0
```

```
! Output omitted for brevity
```

```
BGP routing table entry for 172.24.0.0/24, version 14
```

```
Paths: (2 available, best #2, table default)
```

```
Refresh Epoch 1
```

```
200 100
```

```
192.168.4.4 (metric 3) from 192.168.4.4 (192.168.4.4)
```

```
Origin IGP, aigp-metric 222, localpref 100, valid, internal, best
```

**Table 15-4** explains the processing logic for the BGP best path for XR4, XR5, and R6 as the routers process the path updates.

Phase	Device	Processes
Phase 1	XR4	<ul style="list-style-type: none"> <li>■ XR4 receives the prefix for 172.24.0.0/24 from R2 and sets the AIGP metric to 222.</li> <li>■ XR4 receives the 172.16.0.0/24 and 172.20.0.0/24 prefixes from R2.</li> <li>■ No other paths exist for these prefixes, so all paths are marked as best paths.</li> <li>■ XR4 advertises these paths to XR5 and R6.</li> </ul>
	R6	<ul style="list-style-type: none"> <li>■ R6 receives the prefix for 172.16.0.0/24 from R3 and sets the AIGP metric to 333.</li> <li>■ R6 receives the 172.24.0.0/24 and 172.20.0.0/24 prefixes from R3.</li> <li>■ No other paths exist for these prefixes, so all paths are marked as best paths.</li> <li>■ R6 advertises these paths to XR4 and XR5.</li> </ul>
Phase 2	XR4	<ul style="list-style-type: none"> <li>■ XR4 receives R6's paths for all the prefixes from R3.</li> <li>■ XR4 detects that the 172.16.0.0/24 path from R6 has an AIGP metric and R2's path does not. XR4 marks the path from R3 as the best path for the prefix and sends route withdrawals for the path advertised from R2.</li> <li>■ XR4 detects that the 172.20.0.0/24 path from R2 and R6 does not have an AIGP metric, and the best path is selected using steps after AIGP metric in the best path algorithm.</li> <li>■ XR4 detects that the 172.24.0.0/24 path from R6 does not have an AIGP metric and R2's path does. XR4 marks R2's path as the best path for the prefix.</li> </ul>
	XR5	<ul style="list-style-type: none"> <li>■ XR5 receives paths for all network prefixes from XR4 and R6.</li> <li>■ XR5 detects that the 172.16.0.0/24 path from R3 has an AIGP metric and R3's path does not. XR5 marks R3's path as the best path for the prefix.</li> <li>■ XR5 detects that the 172.20.0.0/24 path from R2 and R3 does not have an AIGP metric, and the best path is selected using steps after AIGP metric in the best path algorithm.</li> <li>■ XR5 detects that the 172.24.0.0/24 path from R3 does not have an AIGP metric and R2's has a shorter AS_Path than the path from R3. XR5 marks the path from R3 as the best path for the prefix.</li> </ul>
	R6	<ul style="list-style-type: none"> <li>■ R6 receives XR4's path advertisement for all the prefixes from R2.</li> <li>■ R6 detects that the 172.16.0.0/24 path from XR4 does not have an AIGP metric, and the path from R3 does. R6 marks that path from XR4 as the best path for the prefix and sends route withdrawals for the path via R3.</li> <li>■ R6 detects that the 172.20.0.0/24 path from XR4 and R6 does not have an AIGP metric, and the best path is selected using steps after the AIGP metric in the best path algorithm.</li> <li>■ R6 detects that the 172.24.0.0/24 path from XR4 has an AIGP metric, and R3's does not. R6 marks the path from XR4 as the best path for the prefix and sends route withdrawals for the path via R3.</li> </ul>
Phase 3	XR4	<ul style="list-style-type: none"> <li>■ XR4 receives R6's withdraw for 172.16.0.0/24 and removes it from the BGP table.</li> </ul>
	XR5	<ul style="list-style-type: none"> <li>■ XR5 receives R6's withdraw for 172.16.0.0/24 and removes it from the BGP table.</li> <li>■ XR5 receives XR4's withdraw for 172.24.0.0/24 and removes it from the BGP table.</li> </ul>
	R6	<ul style="list-style-type: none"> <li>■ R6 receives XR4's withdraw for 172.24.0.0/24 and removes it from the BGP table.</li> </ul>

**Table 15-4** BGP AIGP Metric Processing Logic for XR4, XR5, and R6

The AIGP metric is fairly new; it was added in Cisco IOS XR Version 4.2.0, IOS 15.4(2)S, and IOS XE 3.12S. Cisco allows for the conversion of the AIGP metric into Multi-Exit Discriminator (MED) for traffic selection when the neighboring router does not support AIGP metrics. MED comparison occurs later in the BGP best path algorithm but is a well-known mandatory attribute that all routers support.

To convert the AIGP metric to MED on IOS XR routers, use the BGP neighbor address family configuration command **aigp send med [disable]**, and on IOS routers use the equivalent command **neighbor ip-address aigp send med**.

Example 15-14 demonstrates the configuration of the AIGP to MED feature. Notice that XR4 places the configuration into a BGP neighbor group.

### Example 15-14 AIGP via MED Configuration

[Click here to view code image](#)

```
XR4
router bgp 400
!
neighbor-group AS400
address-family ipv4 unicast
aigp send med
```

```
R6
router bgp 400
!
address-family ipv4
neighbor AS400 aigp send med
```

Example 15-15 provides sample output on XR5 with the **aigp send med** feature enabled on the AS400 routers. Notice that the MED is set to 0 for all routes that did not populate the AIGP metric for a route.

### Example 15-15 AIGP via MED Output

[Click here to view code image](#)

```
RP/0/0/CPU0:XR5#show bgp ipv4 unicast
! Output omitted for brevity

Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop           Metric LocPrf Weight Path
*>i172.16.0.0/24    192.168.4.4        0      100    0 200 100 i
* i                 192.168.6.6        333    100    0 300 100 i
*>i172.20.0.0/24   192.168.4.4        0      100    0 200 100 i
* i                 192.168.6.6        0      100    0 300 100 i
*>i172.24.0.0/24   192.168.4.4        222    100    0 200 100 i
* i                 192.168.6.6        0      100    0 300 100 i

Processed 3 prefixes, 6 paths
```

### Shortest AS\_Path

The next decision factor for the BGP best path algorithm is the AS\_Path length. The path length typically correlates to the autonomous system hop count. A shorter AS\_Path is preferred over a longer AS\_Path.

Note

When working with confederations, `AS_CONFED_SEQUENCE` (confederation AS\_Path) is not counted, and for aggregated addresses with multiple autonomous system numbers (ASNs) under the `AS_SET` portion of the AS\_Path, the `AS_SET` counts only for one AS\_Path entry.

Prepending ASNs to the AS\_Path makes it longer, thereby making that path less desirable when compared with other paths. IOS routers prepend a path with the command **set as-path prepend as-number** on a route map, or the command **prepend as-path as-number** on a route policy. In general, paths that have had the AS\_Path prepended will not be selected as the BGP best path because the AS\_Path is longer than the nonprepending path advertisement. Inbound traffic is influenced by prepending AS\_Path length in advertisement to other autonomous systems, and outbound traffic is influenced by prepending advertisements received from other autonomous systems.

Note

Typically, the AS\_Path is prepended by the network owner with their own ASN used for the prepending. An organization that uses two different service providers for Internet connectivity will commonly advertise smaller network blocks into BGP and alternates prepending of ASNs to balance inbound traffic. The general rule of thumb is when prepending inbound, use the ASN that advertises the route, and when prepending outbound, use your own ASN.

Figure 15-6 demonstrates how AS\_Path prepending influences outbound traffic pattern:

- XR4, XR5, and R6 are in AS400 with AS200 and AS300 providing transient connectivity to AS100.
- XR4 prepends AS200 for the 172.24.0.0/24 path received from R2, making it the least preferred path for AS400.
- R6 prepends AS300 for the 172.16.0.0/24 path received from R3, making it the least preferred path for AS400.

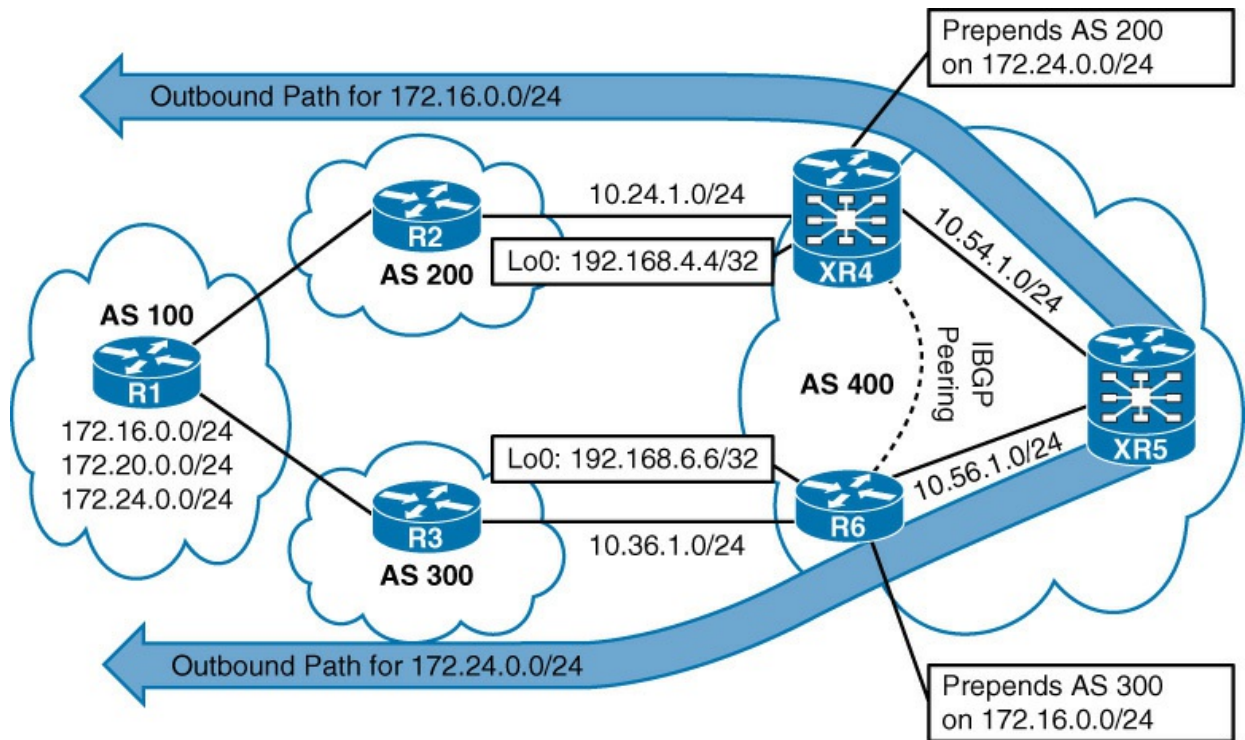


Figure 15-6 Modifying BGP AS\_Path

Example 15-16 demonstrates the configuration for prepending the AS\_Path on XR4 and R6.

**Example 15-16 BGP AS\_Path Prepending Configuration**

[Click here to view code image](#)

**XR4**

```
router bgp 400
  address-family ipv4 unicast
  !
  neighbor-group AS400
    remote-as 400
    update-source Loopback0
    address-family ipv4 unicast
      next-hop-self
    !
  !
  neighbor 10.24.1.2
    remote-as 200
    address-family ipv4 unicast
      route-policy AS200 in
    !
  neighbor 192.168.5.5
    use neighbor-group AS400
  !
  neighbor 192.168.6.6
    use neighbor-group AS400
  !
route-policy AS200
  if destination in (172.24.0.0/24) then
    prepend as-path 222
  endif
  pass
end-policy
```

**R6**

```
router bgp 400
  no bgp default ipv4-unicast
  neighbor AS400 peer-group
  neighbor AS400 remote-as 400
  neighbor AS400 update-source Loopback0
  neighbor 10.36.1.3 remote-as 300
  neighbor 192.168.4.4 peer-group AS400
  neighbor 192.168.5.5 peer-group AS400
  !
  address-family ipv4
    neighbor AS400 next-hop-self
    neighbor 10.36.1.3 activate
    neighbor 10.36.1.3 route-map AS300 in
    neighbor 192.168.4.4 activate
    neighbor 192.168.5.5 activate
  exit-address-family
  !
ip prefix-list PRE172 permit 172.16.0.0/24
!
route-map AS300 permit 10
  match ip address prefix-list PRE172
  set as-path prepend 333
route-map AS300 permit 20
```

**Example 15-17** displays the BGP table for XR4, XR5, and R6.

## Example 15-17 XR4, XR5, and R6 BGP Table After Autonomous System Prepending

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/24	10.24.1.2			0 200	100 i
*> 172.20.0.0/24	10.24.1.2			0 200	100 i
* i	192.168.6.6	0	100	0 300	100 i
* 172.24.0.0/24	10.24.1.2			0 200	200 100 i
*>i	192.168.6.6	0	100	0 300	100 i

```
Processed 3 prefixes, 5 paths
```

```
RP/0/0/CPU0:XR5#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i172.16.0.0/24	192.168.4.4			100	0 200 100 i
*>i172.20.0.0/24	192.168.4.4			100	0 200 100 i
* i	192.168.6.6	0	100	0 300	100 i
*>i172.24.0.0/24	192.168.6.6	0	100	0 300	100 i

```
Processed 3 prefixes, 4 paths
```

```
R6#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i172.16.0.0/24	192.168.4.4	100		0 200	100 i
*	10.36.1.3			0 300	300 100 i
* i172.20.0.0/24	192.168.4.4	100		0 200	100 i
*>	10.36.1.3			0 300	100 i
*> 172.24.0.0/24	10.36.1.3			0 300	100 i

Table 15-5 explains the processing logic for the BGP best path for XR4, XR5, and R6 as the routers process the updates.

Phase	Device	Processes
Phase 1	XR4	<ul style="list-style-type: none"> <li>■ XR4 receives the prefix for 172.24.0.0/24 from R2 and prepends 222 to the AS_Path.</li> <li>■ XR4 receives the 172.16.0.0/24 and 172.20.0.0/24 prefixes from R2.</li> <li>■ No other paths for the prefix exist, so all paths are marked as best path.</li> <li>■ XR4 advertises these paths to XR5 and R6.</li> </ul>
	R6	<ul style="list-style-type: none"> <li>■ R6 receives the prefix for 172.16.0.0/24 from R3 and prepends 333 to the AS_Path.</li> <li>■ R6 receives the 172.24.0.0/24 and 172.20.0.0/24 prefixes from R3.</li> <li>■ No other paths for the prefix exist, so all paths are marked as best path.</li> <li>■ R6 advertises these paths to XR4 and XR5.</li> </ul>
Phase 2	XR4	<ul style="list-style-type: none"> <li>■ XR4 receives R6's paths for all the prefixes from R3.</li> <li>■ XR4 detects that the 172.16.0.0/24 path from R6 has a longer AS_Path than the path from R2. XR4 marks the path from R2 as the best path for the prefix.</li> <li>■ XR4 detects that the 172.20.0.0/24 path from R6 has the same AS_Path length. Because of the tie, the best path is selected using steps after AS_Path length in the best path algorithm.</li> <li>■ XR4 detects that the 172.24.0.0/24 path from R6 has a shorter AS_Path than the path from R2. XR4 marks that path from R3 as the best path for the prefix and sends route withdrawals for the path via R2.</li> </ul>
	XR5	<ul style="list-style-type: none"> <li>■ XR5 receives paths for all network prefixes from XR4 and R6.</li> <li>■ XR5 detects that the 172.16.0.0/24 path from R2 has a shorter AS_Path than the path from R6. XR5 marks the path from XR4 as the best path for the prefix.</li> <li>■ XR5 detects that both the 172.20.0.0/24 paths have identical AS_Path length and proceeds to steps after AS_Path length in the best path algorithm.</li> <li>■ XR5 detects that the 172.24.0.0/24 path from R6 has a shorter AS_Path than the path from XR4. XR5 marks that path from R6 as the best path for the prefix.</li> </ul>



R6	<ul style="list-style-type: none"> <li>■ R6 receives XR4's path advertisement for all the prefixes from R2.</li> <li>■ R6 detects that the 172.16.0.0/24 path from XR4 has a shorter AS_Path than the path from R3. R6 marks that path from XR4 as the best path for the prefix and sends route withdrawals for the path via R3.</li> <li>■ R6 detects that the 172.20.0.0/24 path from R3 has the same AS_Path length as the path from XR4. Because of the tie, the best path is selected using steps after AS_Path length in the best path algorithm.</li> <li>■ R6 detects that the 172.24.0.0/24 path from XR4 has a longer AS_Path than the path from R3. R6 marks that path from R3 as the best path for the prefix.</li> </ul>	
Phase 3	XR4	<ul style="list-style-type: none"> <li>■ XR4 receives R6's withdraw for 172.16.0.0/24 and removes it from the BGP table.</li> </ul>
	XR5	<ul style="list-style-type: none"> <li>■ XR5 receives R6's withdraw for 172.16.0.0/24 and removes it from the BGP table.</li> <li>■ XR5 receives XR4's withdraw for 172.24.0.0/24 and removes it from the BGP table.</li> </ul>
	R6	<ul style="list-style-type: none"> <li>■ R6 receives XR4's withdraw for 172.24.0.0/24 and removes it from the BGP table.</li> </ul>

**Table 15-5** BGP Local Preference Processing Logic for XR4, XR5, and R6

Note

The shortest AS\_Path criteria can be skipped by the best path algorithm on IOS and IOS XR routers with the BGP configuration command **bgp bestpath as-path ignore**. Some versions of IOS do not have a context-sensitive help for this command and require the complete command typed in correctly to function. Many network engineers consider this a hidden feature.

### Origin Type

The next BGP best path decision factor is the well-known mandatory BGP attribute named *origin*. By default, networks that are advertised via the network statement are set with the *IGP* or *i* origin, and redistributed networks are assigned the *Incomplete* or *?* origin attribute. The *EGP* or *e* origin must be set explicitly by IOS XR routers. The origin preference order is as follows:

1. IGP origin (most)
2. EGP origin
3. Incomplete origin (least)

A prefix's origin attribute can be modified with the command **set origin {igp | incomplete}** on a route map, or with the command **set origin {egp | igp | incomplete}** on an IOS XR route policy.

Figure 15-7 demonstrates the modification of the origin attribute:

- XR4, XR5, and R6 are in AS400, with AS200 and AS300 providing transient connectivity to AS100.
- AS100 is advertising the 172.16.0.0/24, 172.20.0.0/24, and 172.24.0.0/24 with the IGP origin.

- XR4 sets the origin to incomplete for the 172.24.0.0/24 path received from R2, making it the least preferred path for AS400.
- R6 sets the origin to incomplete for the 172.16.0.0/24 path received from R3, making it the least preferred path for AS400.

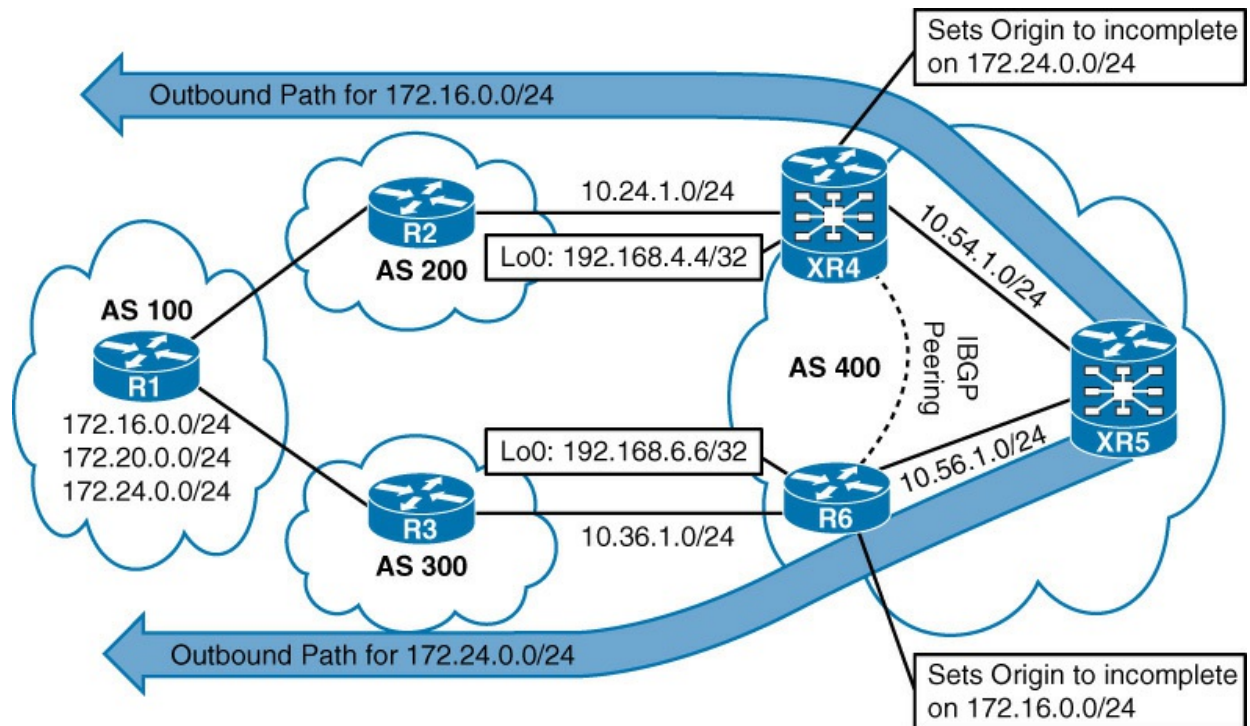


Figure 15-7 BGP Origin Topology

Example 15-18 demonstrates the configuration for modifying the BGP origin attribute on XR4 and R6.

### Example 15-18 BGP Origin Manipulation Configuration

[Click here to view code image](#)

**XR4**

```
router bgp 400
  address-family ipv4 unicast
  !
  neighbor-group AS400
    remote-as 400
    update-source Loopback0
    address-family ipv4 unicast
    next-hop-self
  !
  neighbor 10.24.1.2
    remote-as 200
    address-family ipv4 unicast
    route-policy AS200 in
  !
  neighbor 192.168.5.5
    use neighbor-group AS400
  !
  neighbor 192.168.6.6
    use neighbor-group AS400
  !
route-policy AS200
  if destination in (172.24.0.0/24) then
    set origin incomplete
  endif
  pass
end-policy
```

**R6**

```
router bgp 400
  no bgp default ipv4-unicast
  neighbor AS400 peer-group
  neighbor AS400 remote-as 400
  neighbor AS400 update-source Loopback0
  neighbor 10.36.1.3 remote-as 300
  neighbor 192.168.4.4 peer-group AS400
  neighbor 192.168.5.5 peer-group AS400
  !
  address-family ipv4
    neighbor AS400 next-hop-self
    neighbor 10.36.1.3 activate
    neighbor 10.36.1.3 route-map AS300 in
    neighbor 192.168.4.4 activate
    neighbor 192.168.5.5 activate
  exit-address-family
  !
ip prefix-list PRE172 permit 172.16.0.0/24
  !
route-map AS300 permit 10
  match ip address prefix-list PRE172
  set origin incomplete
route-map AS300 permit 20
```

**Example 15-19** displays the BGP table for XR4, XR5, and R6. Paths with an incomplete origin will

not be selected as the best path because the IGP origin is preferred over the incomplete origin. Notice the origin codes (*e*, *i*, or *?*) on the far right after the AS\_Path information.

### Example 15-19 XR4, XR5, and R6 BGP Table After Origin Manipulation

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast
! Output omitted for brevity

Origin codes: i - IGP, e - EGP, ? - incomplete
  Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.0.0/24   10.24.1.2                0 200 100 i
*> 172.20.0.0/24   10.24.1.2                0 200 100 i
* i                192.168.6.6            0 100    0 300 100 i
* 172.24.0.0/24   10.24.1.2                0 200 100 ?
*>i                192.168.6.6            0 100    0 300 100 i

Processed 3 prefixes, 5 paths
```

```
RP/0/0/CPU0:XR5#show bgp ipv4 unicast
! Output omitted for brevity

Origin codes: i - IGP, e - EGP, ? - incomplete
  Network          Next Hop          Metric LocPrf Weight Path
*>i172.16.0.0/24   192.168.4.4                100    0 200 100 i
*>i172.20.0.0/24   192.168.4.4                100    0 200 100 i
* i                192.168.6.6            0 100    0 300 100 i
*>i172.24.0.0/24   192.168.6.6            0 100    0 300 100 i

Processed 3 prefixes, 4 paths
```

```
R6#show bgp ipv4 unicast
! Output omitted for brevity

Origin codes: i - IGP, e - EGP, ? - incomplete
  Network          Next Hop          Metric LocPrf Weight Path
*>i172.16.0.0/24   192.168.4.4                100    0 200 100 i
*                  10.36.1.3                0 300 100 ?
* i172.20.0.0/24   192.168.4.4                100    0 200 100 i
*>                  10.36.1.3                0 300 100 i
*> 172.24.0.0/24   10.36.1.3                0 300 100 i
```

Table 15-6 explains the processing logic for the BGP best path for XR4, XR5, and R6 as the routers process the updates.

Phase	Device	Processes
Phase 1	XR4	<ul style="list-style-type: none"> <li>■ XR4 receives the prefix for 172.24.0.0/24 from R2 and sets the origin to incomplete (?).</li> <li>■ XR4 receives the 172.16.0.0/24 and 172.20.0.0/24 prefixes from R2.</li> <li>■ No other paths exist for the prefixes, so all paths are marked as best path.</li> <li>■ XR4 advertises these paths to XR5 and R6.</li> </ul>
	R6	<ul style="list-style-type: none"> <li>■ R6 receives the prefix for 172.16.0.0/24 from R3 and sets the origin to incomplete (?).</li> <li>■ R6 receives the 172.24.0.0/24 and 172.20.0.0/24 prefixes from R3.</li> <li>■ No other paths exist for the prefixes, so all paths are marked as best path.</li> <li>■ R6 advertises these paths to XR4 and XR5.</li> </ul>
Phase 2	XR4	<ul style="list-style-type: none"> <li>■ XR4 receives R6's paths for all the prefixes from R3.</li> <li>■ XR4 detects that the 172.16.0.0/24 path from R6 has the incomplete origin versus R2's path with an IGP origin. XR4 marks the path from R2 as the best path for the prefix.</li> <li>■ XR4 detects that the 172.20.0.0/24 path from R6 origin is the same as the path via R2. Because there is a tie, the best path processing uses steps after origin in the best path selection algorithm.</li> <li>■ XR4 detects that the 172.24.0.0/24 path from R6 has the IGP origin versus R2's path with an incomplete origin. XR4 marks the path from R6 as the best path for the prefix, and sends route withdrawals for the path via R2.</li> </ul>
	XR5	<ul style="list-style-type: none"> <li>■ XR5 receives paths for all network prefixes from XR4 and R6.</li> <li>■ XR5 detects that the 172.16.0.0/24 path from R6 has the incomplete origin versus XR4's path with an IGP origin. XR5 marks the path from XR4 as the best path for the prefix.</li> <li>■ XR5 detects that both the 172.20.0.0/24 paths have identical origins and uses steps after origin in the best path selection algorithm.</li> <li>■ XR5 detects that the 172.24.0.0/24 path from XR4 has an incomplete origin versus R6's path with the IGP origin. XR5 marks that path from R6 as the best path for the prefix.</li> </ul>
	R6	<ul style="list-style-type: none"> <li>■ R6 receives XR4's advertisement for all the prefixes from R2.</li> <li>■ R6 detects that the 172.16.0.0/24 path from R3 has the incomplete origin versus XR4's path with the IGP origin. R6 marks that path from XR4 as the best path for the prefix and sends route withdrawals for the path via R3.</li> <li>■ R6 detects that the 172.20.0.0/24 path from R3 has the same origin as the path from XR4. Because of the tie, the best path is selected using steps after origin in the best path algorithm.</li> <li>■ R6 detects that the 172.24.0.0/24 path from XR4 has the incomplete origin versus the path from R3 with an IGP origin. R6 marks that path from R3 as the best path for the prefix.</li> </ul>
Phase 3	XR4	<ul style="list-style-type: none"> <li>■ XR4 receives R6's withdraw for 172.16.0.0/24 and removes it from the BGP table.</li> </ul>
	XR5	<ul style="list-style-type: none"> <li>■ XR5 receives R6's withdraw for 172.16.0.0/24 and removes it from the BGP table.</li> <li>■ XR5 receives XR4's withdraw for 172.24.0.0/24 and removes it from the BGP table.</li> </ul>
	R6	<ul style="list-style-type: none"> <li>■ R6 receives XR4's withdraw for 172.24.0.0/24 and removes it from the BGP table.</li> </ul>

**Table 15-6** BGP Origin Processing Logic for XR4, XR5, and R6

### Multi-Exit Discriminator

The next BGP best path decision factor is the non-transitive BGP attribute named Multi-Exit Discriminator (MED). MED uses a 32-bit value (0–4,294,967,295) called a *metric*. If the MED is received from an eBGP session, it can be advertised to other iBGP peers but should not be sent outside of the autonomous system that received it. MED's purpose is to influence traffic flows inbound from a different autonomous system. A lower MED is preferred over a higher MED.

Note

BGP sets the MED automatically to the IGP path metric during network advertisement or redistribution.

RFC 4451 guidelines state that a prefix without a MED value should be given priority and in essence should be compared with a value of 0. Some organizations require that a MED be set to a specific value for all the prefixes and declare that paths without the MED be treated as the least preferred. By default, if the MED is missing from a prefix learned from an eBGP peer, IOS and IOS XR routers use a MED of 0 for the best path calculation. IOS routers will advertise a MED of 0 to iBGP peers, and IOS XR advertises the path to iBGP peers without a MED.

Figure 15-8 demonstrates the concept in a simple topology. AS100 advertises the 172.16.0.0/24 and 172.20.0.0/24 networks with different MED values at each edge router (R1 and R2). AS200 sends traffic out of R3 to the 172.16.10.0/24 network because R1's MED (40) is lower than R2's MED (60). AS200 sends traffic out of R4 to the 172.20.0.0/24 network because R2's MED (30) is lower than R1's MED (70).

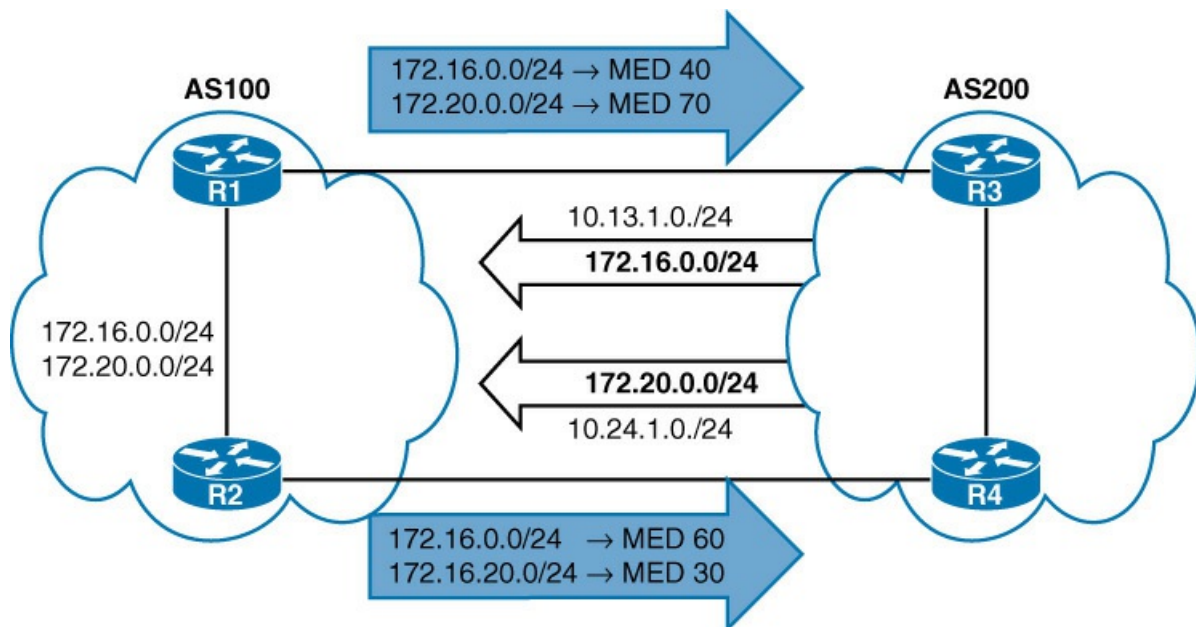


Figure 15-8 MED Influencing Outbound Traffic

Figure 15-9 revisits the previous topology and demonstrates the modification of MED with the following logic:

- XR4, XR5, and R6 are in AS400, with AS200 providing transient connectivity to AS100.
- XR4 sets the MED to 40 for the 172.24.0.0/24 path received from R2, making it the least preferred path for AS400.
- R6 sets the MED to 60 for the 172.16.0.0/24 path received from R3, making it the least preferred path for AS400.

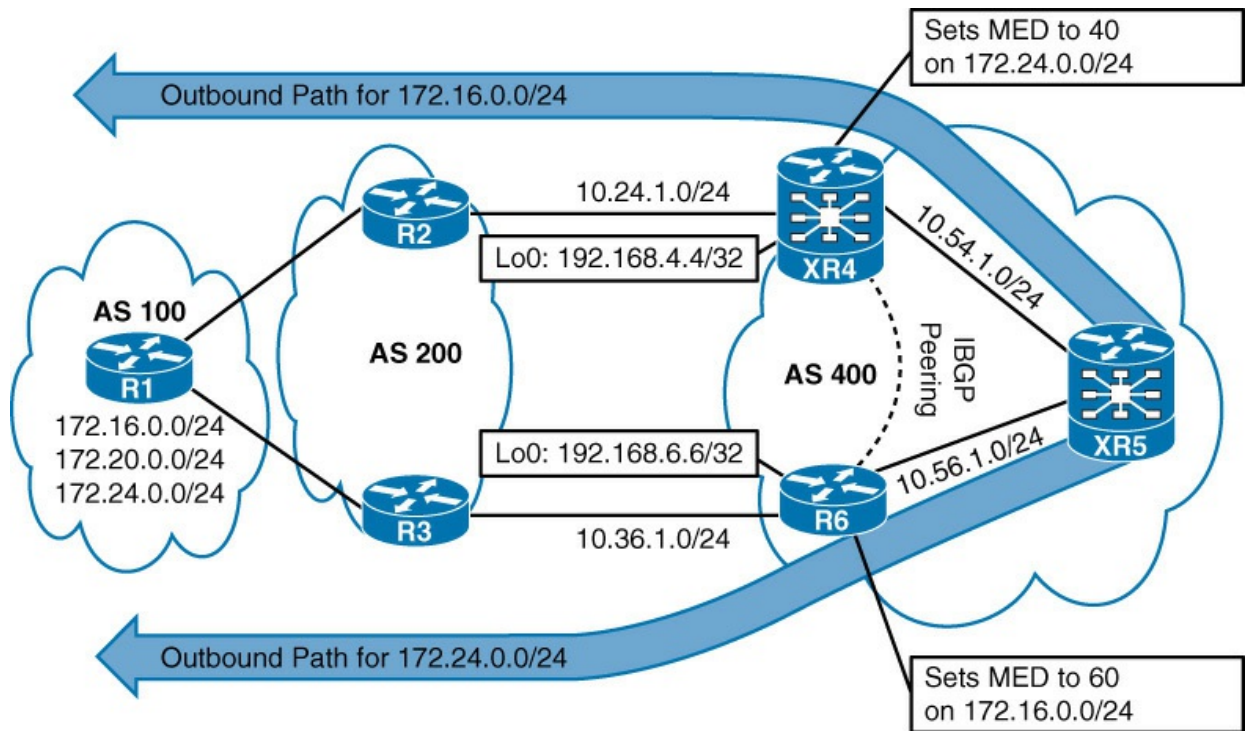


Figure 15-9 BGP MED Manipulation

Example 15-20 demonstrates the configuration for manipulation of the MED on XR4 and R6.

**Example 15-20 BGP MED Manipulation Configuration**

[Click here to view code image](#)

**XR4**

```
router bgp 400
  address-family ipv4 unicast
  !
  neighbor-group AS400
    remote-as 400
    update-source Loopback0
    address-family ipv4 unicast
    next-hop-self
  !
  neighbor 192.168.5.5
    use neighbor-group AS400
  !
  neighbor 192.168.6.6
    use neighbor-group AS400
  !
  neighbor 10.24.1.2
    remote-as 200
    address-family ipv4 unicast
    route-policy AS200 in
  !
  route-policy AS200
    if destination in (172.24.0.0/24) then
      set med 40
    endif
    pass
  end-policy
```

**R6**

```
router bgp 400
  no bgp default ipv4-unicast
  neighbor AS400 peer-group
  neighbor AS400 remote-as 400
  neighbor AS400 update-source Loopback0
  neighbor 10.36.1.3 remote-as 200
  neighbor 192.168.4.4 peer-group AS400
  neighbor 192.168.5.5 peer-group AS400
  !
  address-family ipv4
    neighbor AS400 next-hop-self
    neighbor 10.36.1.3 activate
    neighbor 10.36.1.3 route-map AS200 in
    neighbor 192.168.4.4 activate
    neighbor 192.168.5.5 activate
  exit-address-family
  !
  ip prefix-list PRE172 permit 172.16.0.0/24
  !
  route-map AS200 permit 10
    match ip address prefix-list PRE172
    set metric 60
  route-map AS200 permit 20
```

**Example 15-21** displays the BGP table for XR4, XR5, and R6. All three AS400 routers will send



traffic toward the 172.16.0.0/24 network through XR4's link to R2, and all three AS400 routers will send traffic toward the 172.24.0.0/24 network through R6's link to R3. The 172.20.0.0/24 prefix uses a later step of the BGP best path algorithm for best path selection.

### Example 15-21 XR4, XR5, and R6 BGP Table After MED Modification

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/24	10.24.1.2			0 200	100 i
*> 172.20.0.0/24	10.24.1.2			0 200	100 i
* i	192.168.6.6	0	100	0 200	100 i
*> 172.24.0.0/24	10.24.1.2	40		0 200	100 i
*>i	192.168.6.6	0	100	0 200	100 i

Processed 3 prefixes, 5 paths

---

```
RP/0/0/CPU0:XR5# show bgp ipv4 unicast
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i172.16.0.0/24	192.168.4.4		100	0 200	100 i
*>i172.20.0.0/24	192.168.4.4		100	0 200	100 i
* i	192.168.6.6	0	100	0 200	100 i
*>i172.24.0.0/24	192.168.6.6	0	100	0 200	100 i

Processed 3 prefixes, 4 paths

---

```
R6#show bgp ipv4 unicast
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i172.16.0.0/24	192.168.4.4		100	0 200	100 i
*	10.36.1.3	60		0 200	100 i
* i172.20.0.0/24	192.168.4.4		100	0 200	100 i
*>	10.36.1.3			0 200	100 i
*> 172.24.0.0/24	10.36.1.3			0 200	100 i

Table 15-7 explains the processing logic for the BGP best path selection for routers XR4, XR5, and R6.

Phase	Device	Processes
Phase 1	XR4	<ul style="list-style-type: none"> <li>■ XR4 receives the prefix for 172.24.0.0/24 from R2 and sets the MED to 40.</li> <li>■ XR4 receives the 172.16.0.0/24 and 172.20.0.0/24 prefixes from R2.</li> <li>■ No other paths exist for the prefixes, so all paths are marked as best path.</li> <li>■ XR4 advertises these paths to XR5 and R6.</li> <li>■ (XR4 does not include the MED for 172.16.0.0/24 and 172.20.0.0/24 in the advertisements.)</li> </ul>
	R6	<ul style="list-style-type: none"> <li>■ R6 receives the prefix for 172.16.0.0/24 from R3 and sets the MED to 60.</li> <li>■ R6 receives the 172.24.0.0/24 and 172.20.0.0/24 prefixes from R3.</li> <li>■ No other paths exist for the prefixes, so all paths are marked as best path.</li> <li>■ R6 advertises these paths to XR4 and XR5.</li> <li>■ (R6 sets the MED to 0 for the 172.16.0.0/24 and 172.20.0.0/24 in the advertisements.)</li> </ul>
Phase 2	XR4	<ul style="list-style-type: none"> <li>■ XR4 receives R6's paths for all the prefixes from R2.</li> <li>■ XR4 detects that the 172.16.0.0/24 path from R6 has a higher MED (60) than R2's path, which is 0. (The default MED if a MED is not present.) XR4 marks that path from R2 as the best path for the prefix.</li> <li>■ XR4 detects that the 172.20.0.0/24 path from R6 has an identical MED to R2's path. Because of the tie, the best path is selected using steps after MED in the best path algorithm.</li> <li>■ XR4 detects that the 172.24.0.0/24 path from R2 has a MED (40) that is higher than R6's path with a MED of 0. XR4 marks that path from R6 as the best path for the prefix and sends route withdraws for the path from R2.</li> </ul>
	XR5	<ul style="list-style-type: none"> <li>■ XR5 detects that the 172.16.0.0/24 path from R6 has a higher MED (60) than XR4's path, which is 0. XR5 marks the path from XR4 as the best path for the prefix.</li> <li>■ XR5 detects that both the 172.20.0.0/24 paths have the same MED and uses steps after MED in the best path algorithm.</li> <li>■ XR5 detects that the 172.24.0.0/24 path from XR4 has a higher MED (40) than R6's path, which is 0. XR5 marks that path from R6 as the best path for the prefix.</li> </ul>
	R6	<ul style="list-style-type: none"> <li>■ R6 receives XR4's path advertisement for all the prefixes from R2.</li> <li>■ R6 detects that the 172.16.0.0/24 path from R3 has a higher MED (60) than the XR4's path, which does not have a MED. R6 marks the path from XR4 as the best path for the prefix and sends out route withdraws for the path from R3.</li> <li>■ R6 detects that the 172.20.0.0/24 path from R3 has the same MED from XR4. Because of the tie, the best path is selected using steps after MED in the best path algorithm.</li> <li>■ R6 detects that the 172.24.0.0/24 path from R2 has a higher MED (40) than the path from R3, which uses the default value of 0. R6 marks this path R3 as the best path for the prefix.</li> </ul>
Phase 3	XR4	<ul style="list-style-type: none"> <li>■ XR4 receives R6's withdraw for 172.16.0.0/24 and removes it from the BGP table.</li> </ul>
	XR5	<ul style="list-style-type: none"> <li>■ XR5 receives R6's withdraw for 172.16.0.0/24 and removes it from the BGP table.</li> <li>■ XR5 receives XR4's withdraw for 172.24.0.0/24 and removes it from the BGP table.</li> </ul>
	R6	<ul style="list-style-type: none"> <li>■ R6 receives XR4's for withdraw 172.24.0.0/24 and removes it from the BGP table.</li> </ul>

**Table 15-7** BGP MED Processing Logic for XR4, XR5, and R6

Note

IOS and IOS XR routers by default do not compare MED between member autonomous systems (sub-autonomous systems) in BGP confederations. The BGP configuration command **bgp bestpath med confed** changes the behavior so that MED is compared between member autonomous systems.

## Missing MED behavior

In [Figure 15-9](#), the MED was not included on the prefixes advertised from AS200 or AS300, and those paths without the MED were preferred to the paths with a MED. Scenarios like this could lead to some unintended routing behavior. The command `bgp bestpath med missing-as-worst` under the BGP router process will set the MED to infinity (232 – 1) or (4,294,967,295) if the MED is missing from a path.

[Example 15-22](#) demonstrates setting the MED to infinity when a value is not present on routers XR4 and R6. The configuration is added to XR5, too, because it is important to keep the best path algorithm configuration settings the same on all routers in the autonomous system.

### Example 15-22 BGP MED Missing-as-Worst Configuration

[Click here to view code image](#)

#### XR4

```
router bgp 400
  bgp bestpath med missing-as-worst
```

#### R6

```
router bgp 400
  bgp bestpath med missing-as-worst
```

[Example 15-23](#) displays the BGP table for XR4, XR5, and R6 with the BGP MED missing-as-worst feature enabled. The best path has changed from XR4 to R6 for the 172.16.0.0/24 network and from R6 to XR4 for the 172.24.0.0/24 network because the MED infinity metric is less desirable

### Example 15-23 XR4, XR5, and R6 BGP Table with MED Missing-as-Worst

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
* 172.16.0.0/24	10.24.1.2	4294967295		0	200 100 i
*>i	192.168.6.6	60	100	0	200 100 i
*> 172.20.0.0/24	10.24.1.2	4294967295		0	200 100 i
* i	192.168.6.6	4294967295	100	0	200 100 i
*> 172.24.0.0/24	10.24.1.2	40		0	200 100 i

Processed 3 prefixes, 5 paths

```
RP/0/0/CPU0:XR5# show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i172.16.0.0/24	192.168.6.6	60	100	0 200	100 i
*>i172.20.0.0/24	192.168.4.4	4294967295	100	0 200	100 i
* i	192.168.6.6	4294967295	100	0 200	100 i
*>i172.24.0.0/24	192.168.4.4	40	100	0 200	100 i

```
Processed 3 prefixes, 4 paths
```

```
R6#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/24	10.36.1.3	60		0 200	100 i
* i172.20.0.0/24	192.168.4.4	4294967295	100	0 200	100 i
*>	10.36.1.3	4294967295		0 200	100 i
*>i172.24.0.0/24	192.168.4.4	40	100	0 200	100 i
*	10.36.1.3	4294967295		0 200	100 i

Note

The BGP configuration command **default-metric metric** sets the metric to the value specified when a path is received without a MED. This allows IOS and IOS XR routers to calculate the BGP best path for prefixes without the MED attribute with a value different from 0 or infinity.

### Always Compare Med

The default MED comparison mechanism requires the AS\_Path to be identical because the policies used to set the MED could vary from autonomous system to autonomous system. This means that MED can influence traffic only when multiple links are from the same service provider. Typically, organizations use different service providers for redundancy. In these situations, the default BGP rules for MED comparison need to be relaxed.

The always compare MED feature allows for the comparison of MED regardless of the AS\_Path. The always compare MED feature is enabled with the BGP configuration command **bgp always-compare-med** on IOS routers, and with the BGP configuration command **bgp bestpath med always** on IOS XR routers.

Note

This feature should be enabled on all BGP routers in the autonomous system; otherwise, routing loops can occur.

Figure 15-10 modifies the previous topology so that AS200 contains only R2, and AS300 contains R3:

- XR4 sets the MED to 40 for the 172.24.0.0/24 path received from R2 making it the least preferred path for AS400.
- R6 sets the MED to 60 for the 172.16.0.0/24 path received from R3 making it the least preferred path for AS400.

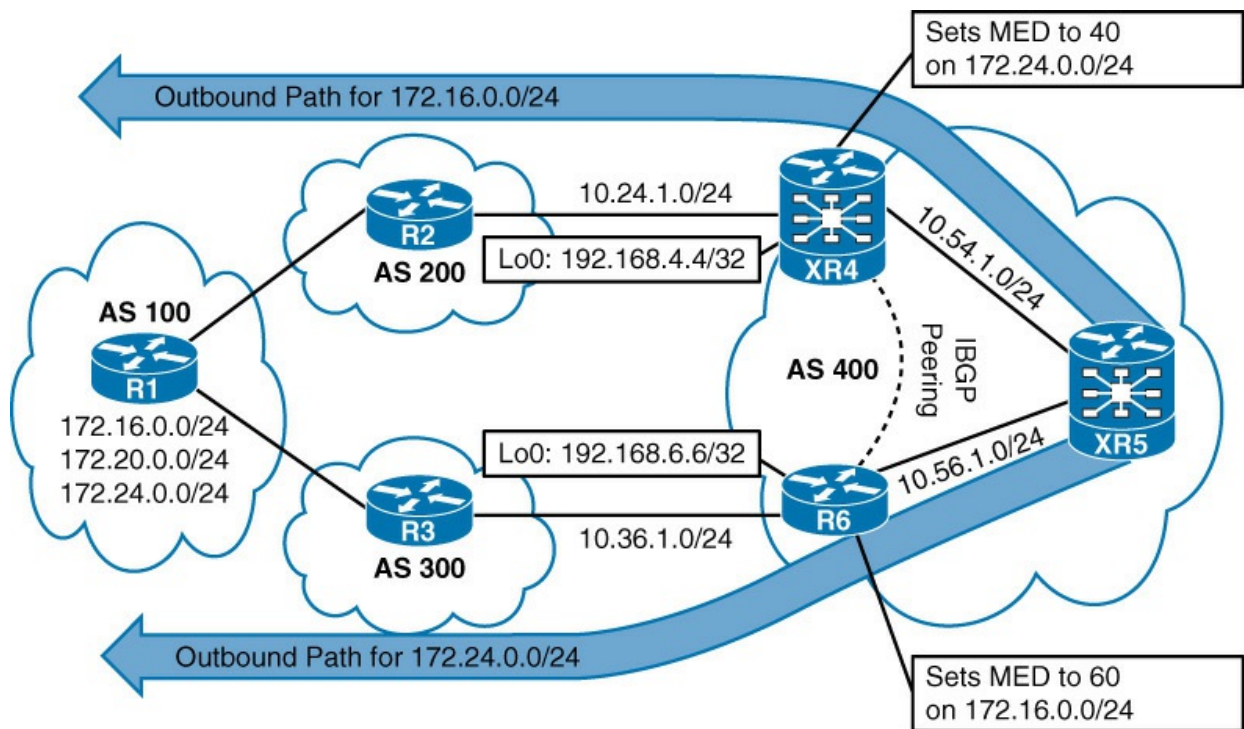


Figure 15-10 Always Compare MED Topology

Example 15-24 demonstrates the supplemental BGP configuration so that the MED attribute is compared between different AS\_Paths.

### Example 15-24 Always Compare MED Configuration

[Click here to view code image](#)

```
XR4
router bgp 400
  bgp bestpath med always
```

```
R6
router bgp 400
  bgp always-compare-med
```

Example 15-25 provides the BGP table on XR4, XR5, and R6. Notice that the MED was compared for the 172.24.0.0/24 network on XR4 and for the 172.16.0.0/24 network on R6 even though the AS\_Paths differ.

### Example 15-25 XR4, XR5, and R6 BGP Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/24	10.24.1.2			0 200	100 i
*> 172.20.0.0/24	10.24.1.2			0 200	100 i
* i	192.168.6.6	0	100	0 300	100 i
* 172.24.0.0/24	10.24.1.2	40		0 200	100 i
*>i	192.168.6.6	0	100	0 300	100 i

```
Processed 3 prefixes, 5 paths
```

```
RP/0/0/CPU0:XR5#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i172.16.0.0/24	192.168.4.4		100	0 200	100 i
*>i172.20.0.0/24	192.168.4.4		100	0 200	100 i
* i	192.168.6.6	0	100	0 300	100 i
*>i172.24.0.0/24	192.168.6.6	0	100	0 300	100 i

```
Processed 3 prefixes, 4 paths
```

```
R6#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
* 172.16.0.0/24	10.36.1.3	60		0 300	100 i
*>i	192.168.4.4		100	0 200	100 i
*> 172.20.0.0/24	10.36.1.3			0 300	100 i
* i	192.168.4.4		100	0 200	100 i
*> 172.24.0.0/24	10.36.1.3			0 300	100 i

## BGP Deterministic MED

The best path algorithm compares a route update to the existing best path and processes the paths in the order they are stored in the Loc-RIB table. The paths are stored in the order that they are received in the BGP table. If always compare MED is not enabled, the path MED is compared only against the existing best path and not all the paths in the Loc-RIB table, which can cause variations in the MED best path comparison process.

**Figure 15-11** demonstrates a topology where MED is not compared due to the order of the path advertisement:

- R4 advertises the 172.16.0.0/24 prefix with a MED of 200, and R5 selects R4's path as the best path because no other paths exist.
- R3 advertises the 172.16.0.0/24 prefix with a MED of 100. The AS\_Path is from a different autonomous system compared to R4's, so MED is not considered in the BGP best path calculation. R4's path remains the best path because it is the oldest eBGP-learned route.
- R2 advertises the 172.16.0.0/24 prefix with a MED of 150. The AS\_Paths differ from R4's, so

MED is not considered in the BGP best path calculation. R4's path remains the best path because it is the oldest eBGP-learned route.

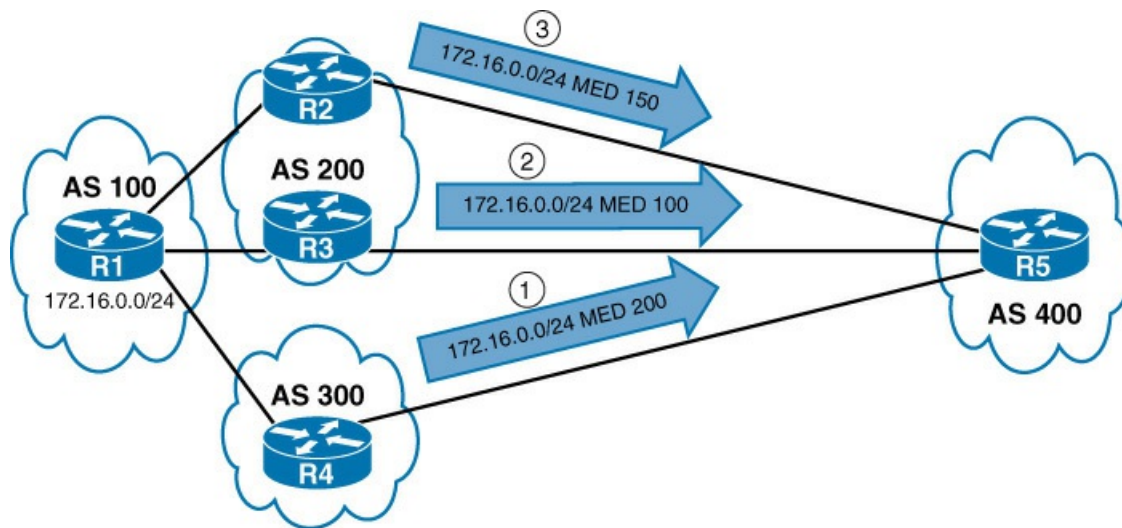


Figure 15-11 Problems with MED Comparison

BGP deterministic MED corrects the problem by grouping paths together with an identical AS\_Path as part of the best path identification process. Each group's MED is compared against the other group's MED.

With BGP deterministic MED enabled, the best path selection outcome is different. R2's and R3's paths are grouped together because they have an identical AS\_Path (200 100). R4 is placed into a separate group, by itself, because of its AS\_Path (300 100). R3 is the best path for AS\_Path group 200 100, and R4 is the best path for AS\_Path group 300 100. The two AS\_Path groups are then compared against each other, and because R3's MED is lower than R4's, R3's path is chosen as the best path regardless of the order the routes are advertised.

In IOS, BGP deterministic MED is enabled with the BGP configuration command **bgp deterministic-med** and is recommended for all BGP deployments. In IOS XR, deterministic MED is enabled by default and cannot be disabled.

#### eBGP over iBGP

The next BGP best path decision factor is whether the route comes from an iBGP, eBGP, or confederation member autonomous system (sub-autonomous system) peering. The best path selection order is as follows:

1. eBGP peers (most desirable)
2. Confederation member autonomous system peers
3. iBGP peers (least desirable)

Figure 15-12 demonstrates the eBGP over iBGP selection process:

- XR4, XR5, and R6 are in AS400, with AS200 and AS300 providing connectivity to AS100.
- AS100 is advertising the 172.16.0.0/24 network.

- XR4 and R6 set the local preference to 100 and MED to 0 for all routes to simplify the BGP table.
- XR4 prefers to send traffic to R2 for the 172.16.0.0/24 network, and R6 prefers to send traffic to R3 for the same 172.16.0.0/24 network.

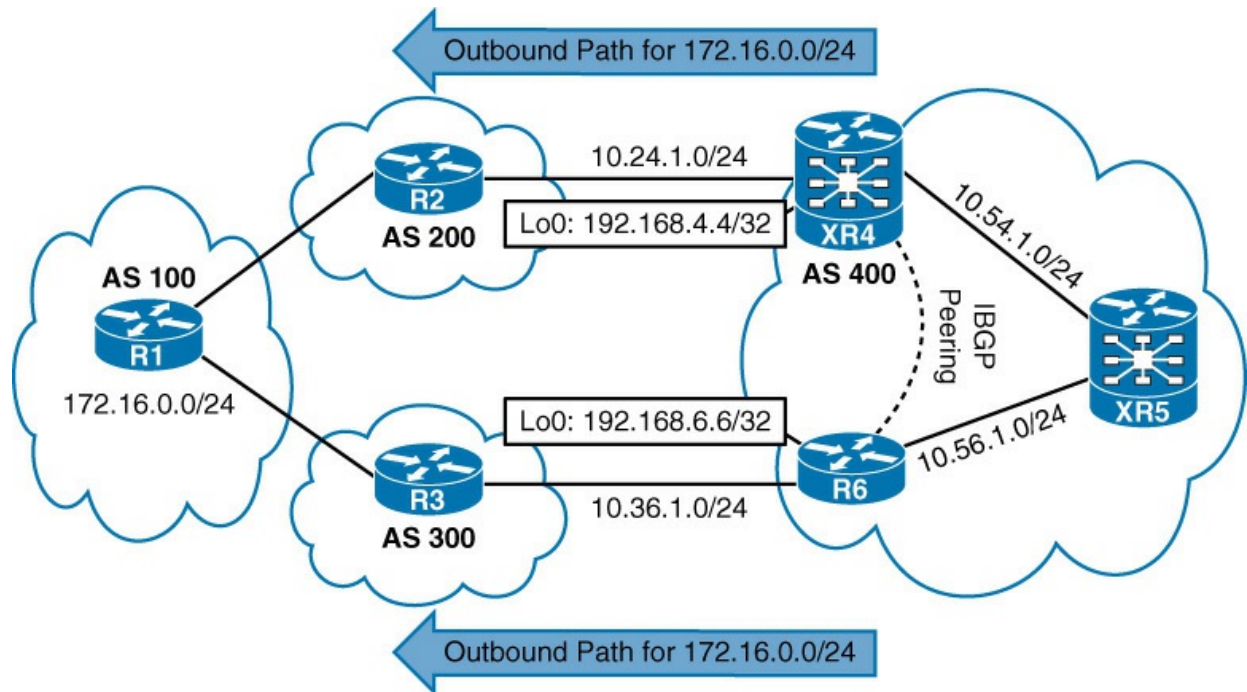


Figure 15-12 eBGP over iBGP Topology

Example 15-26 demonstrates the BGP configuration for XR and R6, where the MED is set to 0 and the local preference is set to 100 to make the BGP table easier to read.

**Example 15-26** BGP Configuration for eBGP over iBGP

[Click here to view code image](#)



**XR4**

```
router bgp 400
  address-family ipv4 unicast
  !
  neighbor-group AS400
    remote-as 400
    update-source Loopback0
    address-family ipv4 unicast
    next-hop-self
  !
  neighbor 192.168.5.5
    use neighbor-group AS400
  !
  neighbor 192.168.6.6
    use neighbor-group AS400
  !
  neighbor 10.24.1.2
    remote-as 200
    address-family ipv4 unicast
    route-policy AS200 in
  !
  route-policy AS200
    set local-preference 100
    set med 0
  end-policy
```

**R6**

```
router bgp 400
  no bgp default ipv4-unicast
  neighbor 192.168.4.4 remote-as 400
  neighbor 192.168.4.4 update-source Loopback0
  neighbor 192.168.5.5 remote-as 400
  neighbor 192.168.5.5 update-source Loopback0
  neighbor 10.36.1.3 remote-as 300
  !
  address-family ipv4
    neighbor 192.168.4.4 activate
    neighbor 192.168.4.4 next-hop-self
    neighbor 192.168.5.5 activate
    neighbor 192.168.5.5 next-hop-self
    neighbor 10.36.1.3 activate
    neighbor 10.36.1.3 route-map AS300 in
  exit-address-family
  !
  route-map AS300 permit 10
    set metric 0
    set local-preference 100
```

**Example 15-27** displays the BGP table for XR4, XR5, and R6. XR4 prefers R2's eBGP path compared to R6's iBGP path. R6 prefers R3's eBGP path compared to XR4's iBGP path. XR5 uses later steps in the best path selection algorithm to identify the best path. Notice that XR4 and R6 maintain the other router's path even though it is the best path. This is because both routers think their paths are the best and never sent out a route withdraw for the prefix.

Note

iBGP-learned routes contain an *i* next to the prefix (*Network* column). This is the quickest way to differentiate between an eBGP and iBGP learned routes.

### Example 15-27 XR4, XR5, and R6 BGP Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/24	10.24.1.2	0	100	0 200	100 i
* i	192.168.6.6	0	100	0 300	100 i

```
Processed 1 prefixes, 2 paths
```

```
RP/0/0/CPU0:XR5#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i172.16.0.0/24	192.168.4.4	0	100	0 200	100 i
* i	192.168.6.6	0	100	0 300	100 i

```
Processed 1 prefixes, 2 paths
```

```
R6#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/24	10.36.1.3	0	100	0 300	100 i
* i	192.168.4.4	0	100	0 200	100 I

Table 15-8 explains the best path selection logic for routers XR4, XR5, and R6 as they receive updates.

**Phase Device Processes**

Phase 1	XR4	<ul style="list-style-type: none"> <li>XR4 receives the 172.16.0.0/24 prefix from R2 and sets the LOCAL_PREF to 100 and MED to 0.</li> <li>Only one path for the 172.16.0.0/24 prefix exists, and it is marked as the best path.</li> <li>XR4 advertises this path to XR5 and R6.</li> </ul>
	R6	<ul style="list-style-type: none"> <li>R6 receives the 172.16.0.0/24 paths from R3 and sets the LOCAL_PREF to 100 and MED to 0.</li> <li>Only one path for the 172.16.0.0/24 prefix exists, and it is marked as the best path.</li> <li>R6 advertises this path to XR4 and XR5.</li> </ul>
Phase 2	XR4	<ul style="list-style-type: none"> <li>XR4 receives the 172.16.0.0/24 path from R6, but because it is an iBGP-learned prefix, XR4 does not change the current best path to R6's path.</li> </ul>
	XR5	<ul style="list-style-type: none"> <li>XR5 receives the 172.16.0.0/24 paths from XR4 and R6, but because they are both iBGP-learned paths, XR5 selects the best path for the prefixes using later steps after eBGP over iBGP in the best path algorithm.</li> </ul>
	R6	<ul style="list-style-type: none"> <li>R6 receives the 172.16.0.0/24 path from XR4, but because it is an iBGP-learned prefix, R6 does not change the current best path to XR4's path.</li> </ul>

Table 15-8 eBGP over iBGP Processing Logic for XR4, XR5, and R6

**Lowest IGP Metric**

The next decision step is to use the lowest IGP cost to the BGP next-hop address.

Figure 15-13 illustrates a topology where XR2, XR3, R4, and R5 are in AS400. AS400 peers in a full mesh and establishes BGP sessions using Loopback 0 interfaces. R1 advertises the 172.16.0.0/24 network to XR2 and R4.

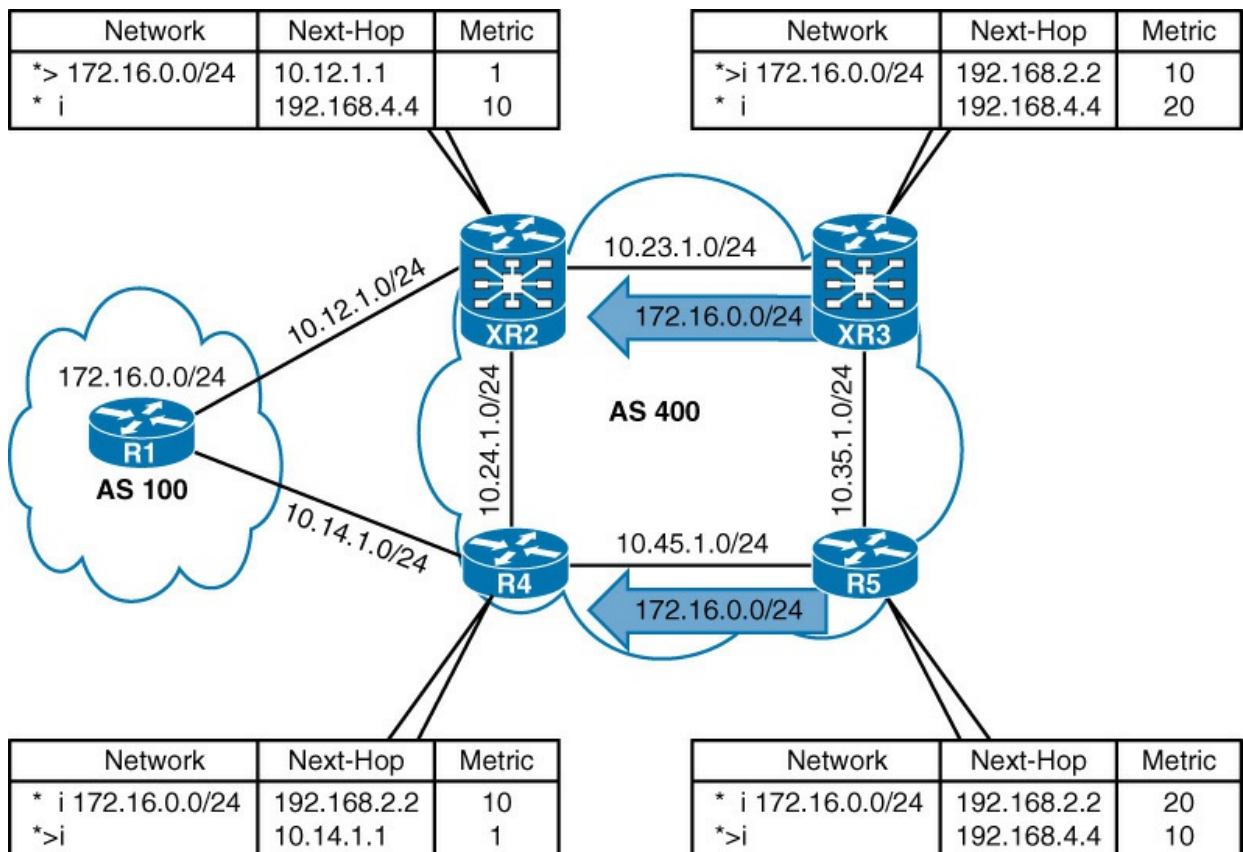


Figure 15-13 Lowest IGP Metric Topology

XR3 prefers the path from XR2 compared to the iBGP path from R4 because the metric to reach the next-hop address is lower. R5 prefers the path from R4 compared to the iBGP path from XR2 because the metric to reach the next-hop address is lower.

[Example 15-28](#) displays XR3's and R5's IP routing table for the Loopback 0 interfaces of their BGP peers. XR3 has a path metric of 20 to reach XR2's interface and a path metric of 30 to reach R4's interface. R5 has a path metric of 20 to reach R4's interface and a path metric of 40 to reach XR2's interface. XR3 should use XR2's path and R5 should use R4's path to reach the 172.16.0.0/24 network.

### **Example 15-28** *OSPF Routing Tables for XR3 and R5*

[Click here to view code image](#)

```
RP/0/0/CPU0:XR3#show route ospf
```

```
! Output omitted for brevity
```

```
O 192.168.2.2/32 [110/20] via 10.23.1.2, 00:47:56, GigabitEthernet0/0/0/0
O 192.168.4.4/32 [110/30] via 10.35.1.5, 00:00:40, GigabitEthernet0/0/0/2
  [110/30] via 10.23.1.2, 00:00:40, GigabitEthernet0/0/0/0
O 192.168.5.5/32 [110/20] via 10.35.1.5, 00:50:39, GigabitEthernet0/0/0/2
```

```
R5#show ip route ospf
```

```
! Output omitted for brevity
```

```
O 192.168.2.2 [110/30] via 10.45.1.4, 00:49:27, GigabitEthernet0/0
  [110/30] via 10.35.1.3, 00:49:27, GigabitEthernet0/2
O 192.168.3.3 [110/20] via 10.35.1.3, 00:52:09, GigabitEthernet0/2
O 192.168.4.4 [110/20] via 10.45.1.4, 00:02:11, GigabitEthernet0/0
```

[Example 15-29](#) verifies that XR3 chose XR2 because the other path had a higher metric.

### **Example 15-29** *XR3 BGP Best Path Compare*

[Click here to view code image](#)

```
RP/0/0/CPU0:XR3#show bgp ipv4 unicast 172.16.0.0/24 bestpath-compare
```

```
! Output omitted for brevity
```

```
BGP routing table entry for 172.16.0.0/24
```

```
Paths: (2 available, best #1)
```

```
Not advertised to any peer
```

```
Path #1: Received by speaker 0
```

```
Not advertised to any peer
```

```
100
```

```
192.168.2.2 (metric 20) from 192.168.2.2 (192.168.2.2)
```

```
Origin IGP, metric 0, localpref 100, valid, internal, best, group-best
```

```
Received Path ID 0, Local Path ID 1, version 11
```

```
best of AS 100, Overall best
```

```
Path #2: Received by speaker 0
```

```
Not advertised to any peer
```

```
100
```

```
192.168.4.4 (metric 30) from 192.168.4.4 (192.168.4.4)
```

```
Origin IGP, metric 0, localpref 100, valid, internal
```

```
Received Path ID 0, Local Path ID 0, version 0
```

```
Higher IGP metric than best path (path #1)
```

Note

IOS and IOS XR routers can disable the lowest IGP metric step with the BGP address family configuration command `bgp bestpath igp-metric ignore`.

### Prefer the Oldest EBGPath Path

BGP can maintain large routing tables, and unstable sessions result in the BGP best path calculation to execute frequently. BGP maintains stability in a network by preferring the oldest (established) BGP session.

Note

This step can be skipped on IOS and IOS XR routers with the BGP configuration command `bgp bestpath compare-routerid`.

### Router ID

The next step for the BGP best path algorithm is to select the best path using the lowest router ID (RID) of the advertising eBGP router. If the route was received by a route reflector (RR), the originator ID is substituted for the RID.

The topology in [Figure 15-12](#) and the BGP configurations from [Example 15-26](#) are revisited to explain the best path selection process when comparing RIDs. The focus will be on XR5's calculation of the BGP best path for the 172.16.1.0/24 network. [Example 15-30](#) provides XR5's BGP table.

### Example 15-30 XR5's BGP Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR5#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i172.16.0.0/24	192.168.4.4	0	100	0	200 100 i
* i	192.168.6.6	0	100	0	300 100 i

```
Processed 1 prefixes, 2 paths
```

**Example 15-31** displays the BGP best path compare for the 172.16.0.0/24 network so that you can see all the BGP path attributes. Path 1 is using XR4's RID of 192.168.4.4, and path 2 is using R6's RID of 192.168.6.6. Path 1 is a lower RID and is therefore the BGP best path. The last line confirms that the higher router ID was the reason that path 2 was not selected.

### **Example 15-31** BGP Best Path Compare for 172.16.0.0/24 Network

[Click here to view code image](#)

```
RP/0/0/CPU0:XR5#show bgp ipv4 unicast 172.16.0.0/24 bestpath-compare
```

```
BGP routing table entry for 172.16.0.0/24
```

```
Paths: (2 available, best #1)
```

```
Not advertised to any peer
```

```
Path #1: Received by speaker 0
```

```
Not advertised to any peer
```

```
200 100
```

```
192.168.4.4 (metric 2) from 192.168.4.4 (192.168.4.4)
```

```
Origin IGP, metric 0, localpref 100, valid, internal, best, group-best
```

```
Received Path ID 0, Local Path ID 1, version 14
```

```
best of AS 200, Overall best
```

```
Path #2: Received by speaker 0
```

```
Not advertised to any peer
```

```
300 100
```

```
192.168.6.6 (metric 2) from 192.168.6.6 (192.168.6.6)
```

```
Origin IGP, metric 0, localpref 100, valid, internal, group-best
```

```
Received Path ID 0, Local Path ID 0, version 0
```

```
best of AS 300
```

```
Higher router ID than best path (path #1)
```

### **Minimum Cluster List Length**

The next step in the BGP best path algorithm is to select the best path using the lowest *cluster list* length. The cluster list is a nontransitive BGP attribute that is appended (not overwritten) by a route reflector with its *cluster ID*. The cluster ID attribute is used by RRs as a loop-prevention mechanism. The cluster ID is not advertised between autonomous systems and is locally significant. In simplest terms, this step locates the path that has traveled the least number of iBGP advertisement hops.

**Figure 15-14** demonstrates the minimum length cluster list criteria:

- R3 advertises the 172.16.0.0/24 network to RR1 and RR2 with only the originator ID.
- RR1 reflects the advertisement to RR2 after appending its RID to the cluster list.
- RR2 selects the path advertisement directly from R3. R3's cluster list length is 0, which is more

desirable compared to RR1's cluster list length of 1.

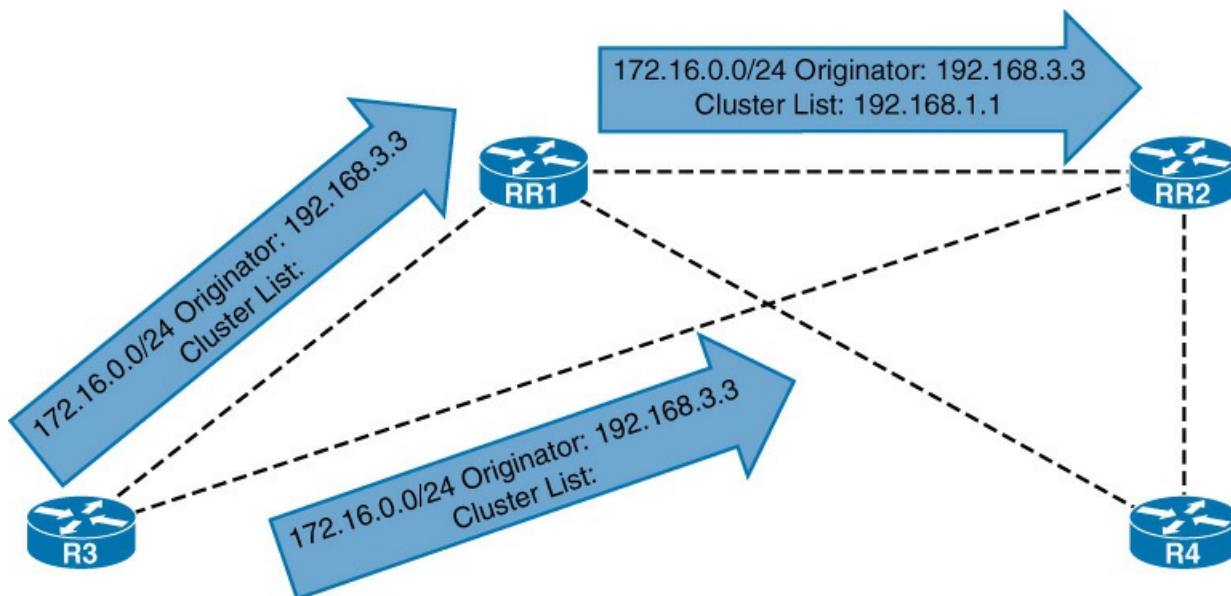


Figure 15-14 Minimum Cluster List Length

### Lowest Neighbor Address

The last step of the BGP best path algorithm selects the path that comes from the lowest BGP neighbor address. This step is limited to iBGP peerings because eBGP peerings use the oldest received path as the tiebreaker.

Figure 15-15 demonstrates the concept of choosing the router with the lowest neighbor address. R1 is advertising the 172.16.0.0/24 network to R2. R1 and R2 have established two BGP sessions using the 10.12.1.0/24 and 10.12.2.0/24 network. R2 will select the path advertised from 10.12.1.1 as it is the lower IP address.

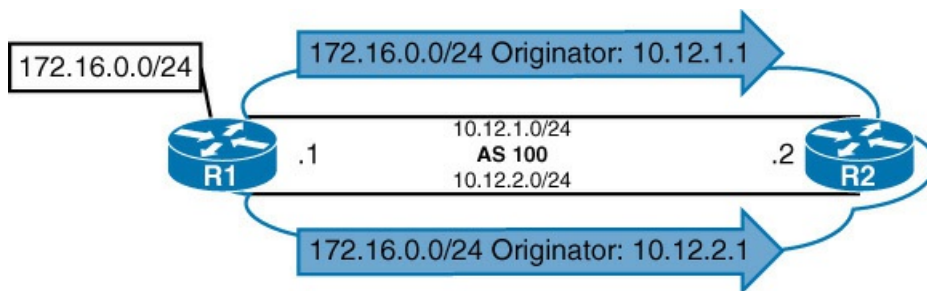


Figure 15-15 Lowest IP Address

Note

This design is inefficient in most scenarios because only one link is used. A better solution is to peer via a loopback interface and let the IGP load balance across the link accordingly or use 802.3ad Link Aggregation (LAG) bundles.

### BGP ECMP

All the IGP routing protocols explained in this book support equal-cost load balancing (equal-cost multi-path [ECMP]). ECMP provides load balancing by installing multiple paths into the RIB for that protocol. BGP selects only one best path but does allow for the installation of multiple routes into the RIB. BGP multipath has three different variances in behavior:

- eBGP multipath

- iBGP multipath

- eiBGP multipath

Enabling BGP multipath does not alter the best path algorithm or change the behavior of paths advertisement to other BGP peers. Only the BGP best path is advertised to peers. The following sections explain the various iterations of BGP multipath.

#### eBGP and iBGP Multipath

Upon configuring BGP multipath, the additional paths will need to match the following best path BGP path attributes:

- Weight.
- Local preference.
- AS\_Path length.
- AS\_Path content. (Confederations can contain a different AS\_CONFED\_SEQ path.)
- Origin.
- MED.
- Advertisement method must match (iBGP or eBGP). If the prefix is learned via an iBGP advertisement, the IGP cost must match to be considered equal.

eBGP multipath is enabled on IOS routers with the BGP configuration command **maximum-paths number-paths**. The number of paths indicates the allowed number of eBGP paths to install in the RIB. The command **maximum-paths ibgp number-paths** sets the number of iBGP routes to install in the RIB. The commands are placed under the appropriate address family.

IOS XR routers set the number of paths for installation into the RIB with the BGP configuration command **maximum-paths {ebgp | ibgp} number-paths [unequal-cost]**. The **unequal-cost** keyword allows for iBGP paths to install if the metrics to the next hop do not match.

Note

Multipath eligibility begins only after the BGP best path algorithm executes. Alternate paths must match the exact BGP best path attributes listed earlier for BGP multipath consideration.

Figure 15-16 provides a basic topology to demonstrate eBGP multipath on XR4 and R5. R1 advertises the 172.16.0.0/24 network, and transits AS200 to reach AS400. XR4 and R5 enable BGP multipath for two eBGP paths to install into the RIB.



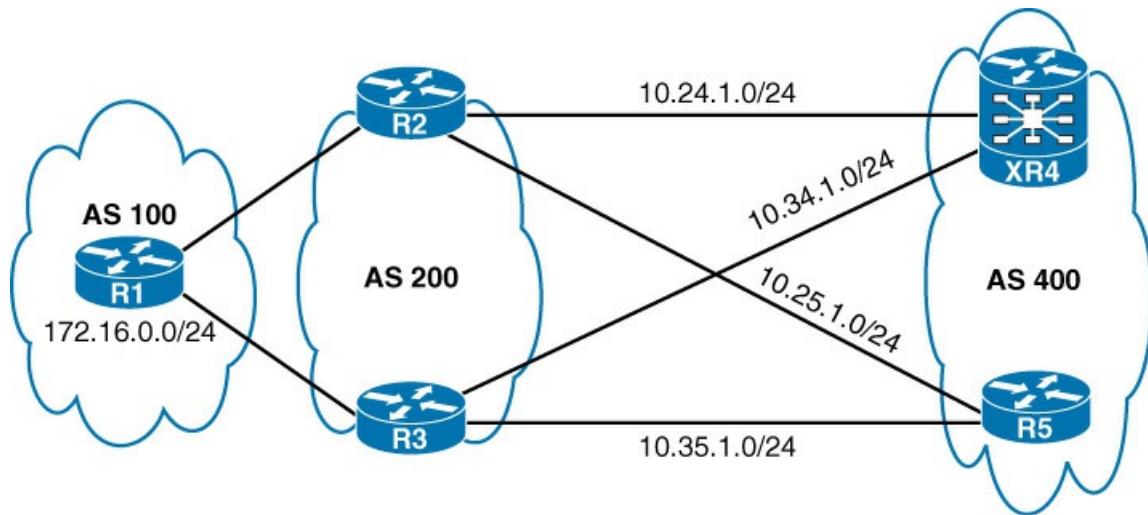


Figure 15-16 eBGP Multipath Topology

Example 15-32 demonstrates the BGP configuration to enable eBGP multipath for a maximum of two paths.

### Example 15-32 eBGP Multipath Configuration

[Click here to view code image](#)

```
XR4
router bgp 400
address-family ipv4 unicast
maximum-paths ebgp 2
```

```
R5
router bgp 400
!
address-family ipv4
maximum-paths 2
```

Example 15-33 displays XR4's and R5's BGP table. Notice that the path via R2 was selected as the BGP best path because of the lower RID.

### Example 15-33 XR4 and R5 BGP Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast
! Output omitted for brevity

Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.0.0/24  10.24.1.2         0 200 100 i
*                 10.34.1.3         0 200 100 i

Processed 1 prefixes, 2 paths
```

```
R5#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

	Network	Next Hop	Metric	LocPrf	Weight	Path
*	172.16.0.0/24	10.35.1.3			0 200 100	i
*>		10.25.1.2			0 200 100	i

[Example 15-34](#) displays the 172.16.0.0/24 prefix and its path attributes for both paths.

### Example 15-34 BGP Attributes for 172.16.0.0/24 Network

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast 172.16.0.0/24
```

```
! Output omitted for brevity
```

```
BGP routing table entry for 172.16.0.0/24
```

```
Paths: (2 available, best #1)
```

```
Path #1: Received by speaker 0
```

```
200 100
```

```
10.24.1.2 from 10.24.1.2 (192.168.2.2)
```

```
Origin IGP, localpref 100, valid, external, best, group-best, multipath
```

```
Path #2: Received by speaker 0
```

```
200 100
```

```
10.34.1.3 from 10.34.1.3 (192.168.3.3)
```

```
Origin IGP, localpref 100, valid, external, multipath
```

```
R5#show bgp ipv4 unicast 172.16.0.0
```

```
! Output omitted for brevity
```

```
BGP routing table entry for 172.16.0.0/24, version 3
```

```
Paths: (2 available, best #2, table default)
```

```
Multipath: eBGP
```

```
Advertised to update-groups:
```

```
1
```

```
Refresh Epoch 1
```

```
200 100
```

```
10.25.1.2 from 10.25.1.2 (192.168.2.2)
```

```
Origin IGP, localpref 100, valid, external, multipath(oldest)
```

```
Refresh Epoch 1
```

```
200 100
```

```
10.35.1.3 from 10.35.1.3 (192.168.3.3)
```

```
Origin IGP, localpref 100, valid, external, multipath, best
```

[Table 15-9](#) places the path attributes from [Example 15-34](#) into a table so that all the required path attributes for eBGP multipath match.

BGP Attribute	Path 1	Path 2	Conditions Match
Weight	Not present	Not present	✓
Local preference	100	100	✓
AS_Path length	2	2	✓
AS_Path content	200 100	200 100	✓
Origin	IGP	IGP	✓
MED	Not present	Not present	✓
Learned via	eBGP	eBGP	✓

Table 15-9 eBGP Multipath Requirement Check

Example 15-35 displays XR4's and R5's RIB confirming that BGP successfully installed multiple eBGP paths for the 172.16.0.0/24 network into the RIB.

### Example 15-35 XR4's and R5's RIB

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show route bgp
```

```
B    172.16.0.0/24 [20/0] via 10.34.1.3, 00:01:57
      [20/0] via 10.24.1.2, 00:01:57
```

```
R5#show ip route bgp
```

```
B    172.16.0.0 [20/0] via 10.35.1.3, 00:01:38
      [20/0] via 10.25.1.2, 00:01:38
```

### eiBGP Multipath

The rules of iBGP or eBGP multipath prevent a router from installing an eBGP and iBGP route into the RIB at the same time. However, in some environments, this can prevent an organization from using the full bandwidth available.

Figure 15-17 demonstrates a topology where load balancing between eBGP and iBGP peers is beneficial. AS400 has two 1-Gbps circuits connected at XR4 and R5 for connectivity. At times, servers directly attached to R7 are generating 1.5 Gbps of traffic destined to the 172.16.0.0/24 network in AS100 via the R5 exit point. R5 does not have enough bandwidth to send all the data across the 10.35.1.0/24 network, or the ability to load balance the traffic across the two 1-Gbps circuits. Traffic exceeding the 1-Gbps rate is dropped.

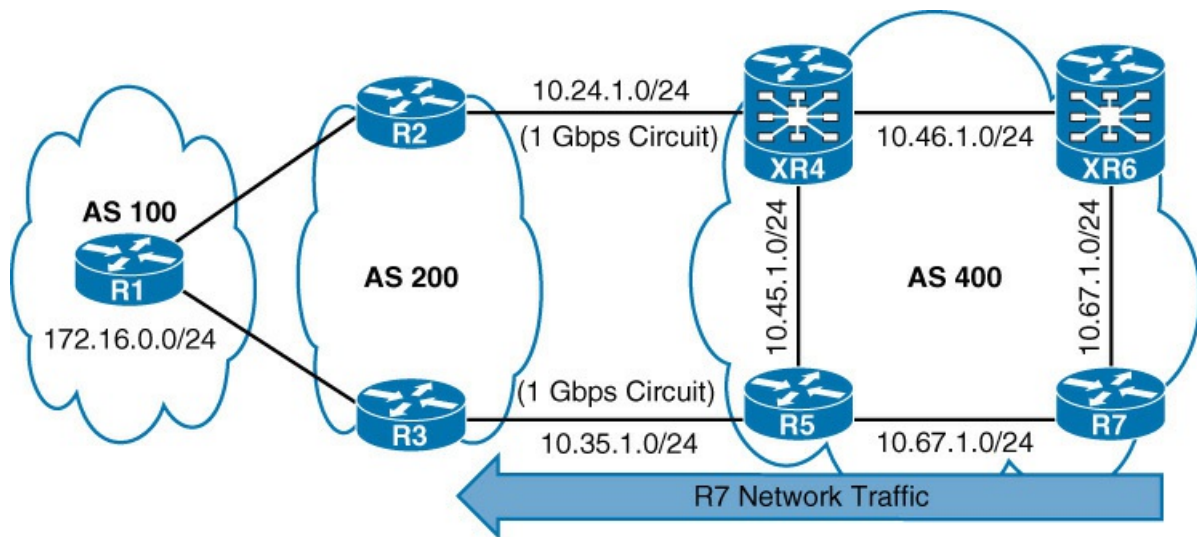


Figure 15-17 Topology with Throughput Issues

eiBGP multipath allows for an eBGP and iBGP path with unequal next-hop metrics to be selected as the best path. If an iBGP route is used with a primary eBGP path, the AD is changed to 20 so that both routes install into the RIB.

Note

Configuring iBGP and eBGP multipath is not the same behavior as configuring eiBGP multipath. In fact, they cannot be configured at the same time on a router.

Figure 15-18 demonstrates eiBGP multipath in action. The feature is enabled on the edge routers (XR4 and R5) for AS400. R7 installs the route to 172.16.0.0/24 with R5 as the next hop. Upon receipt of the packet, R5 has two entries available and forwards packets directly to R3 or to XR4 (so it can be sent out the 10.24.1.0/24 link). Traffic can now utilize the full bandwidth between AS200 and AS400.

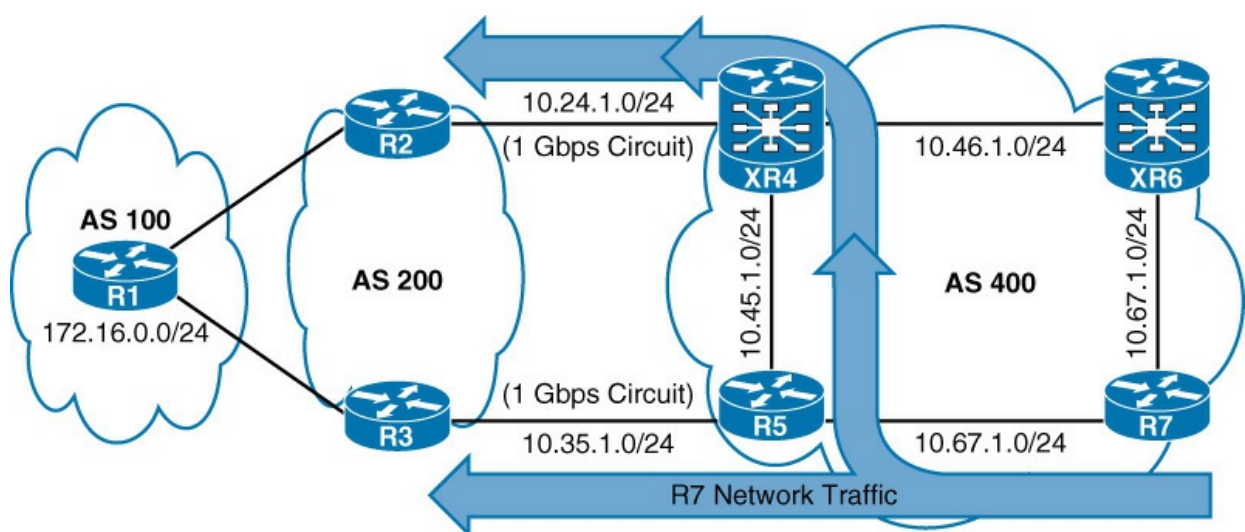


Figure 15-18 eiBGP Multipath on BGP Edge Routers

Figure 15-19 demonstrates eiBGP multipath enabled on core BGP routers (XR6 and R7) for AS400. Even though R5 and XR4 are two different IGP metric values, R7 is able to install R5 and XR6 as next-hop addresses for traffic destined for the 172.16.0.0/24 network.

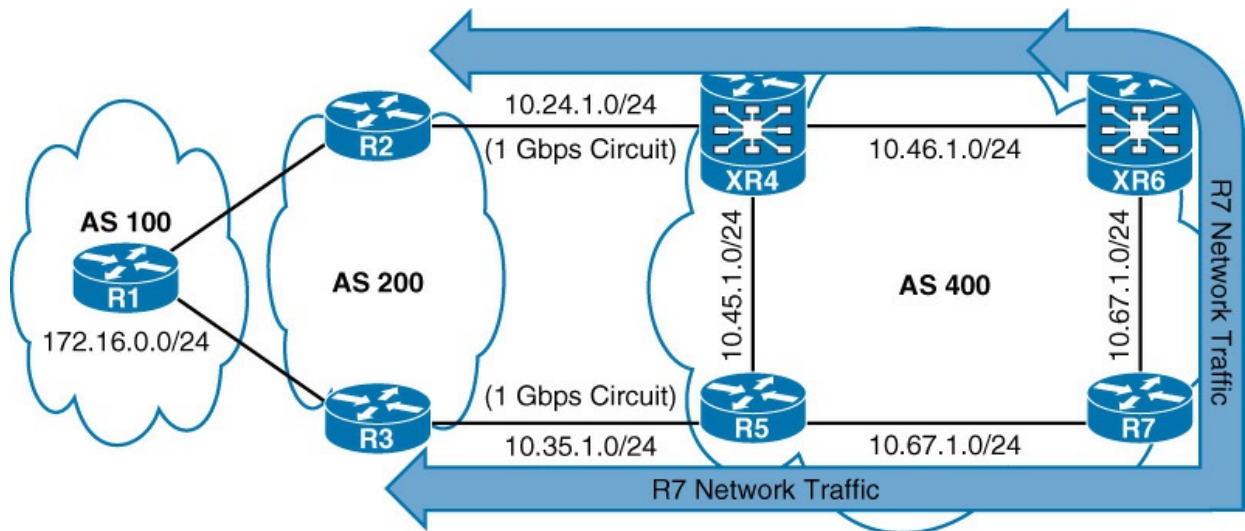


Figure 15-19 eiBGP Multipath with Core Deciding Path

Traffic received by XR6 is forwarded on to XR4 and then on to R2, while traffic received on R5 is forwarded on to R3. This allows for full link utilization, but the decision is made closer to the source forwarding the traffic. Figure 15-19 illustrates the traffic flow from R7 to R1 using unequal IGP metrics.

eiBGP multipath support (non-VRF) was released in Cisco IOS 15.4(1)T, IOS XE 3.10S, and IOS XR 3.72. eiBGP multipath configuration on IOS and IOS XR routers uses the BGP address family configuration command **maximum-paths eibgp number-paths**.

Figure 15-20 provides a topology to demonstrate the eiBGP multipath calculation. The BGP table for the 172.16.0.0/24 network has been included in the figure:

- R1, R2, XR3, XR4, and XR5 have established an iBGP full mesh for AS100.
- R1, XR3, and X5 set the next-hop-self to iBGP peers.
- AS100 routers are using OSPF as the IGP with the appropriate link costs in parentheses.
- eiBGP maximum paths have been set to 5 for all routers in AS100.
- The BGP table for all the routers is displayed after BGP best path calculation.

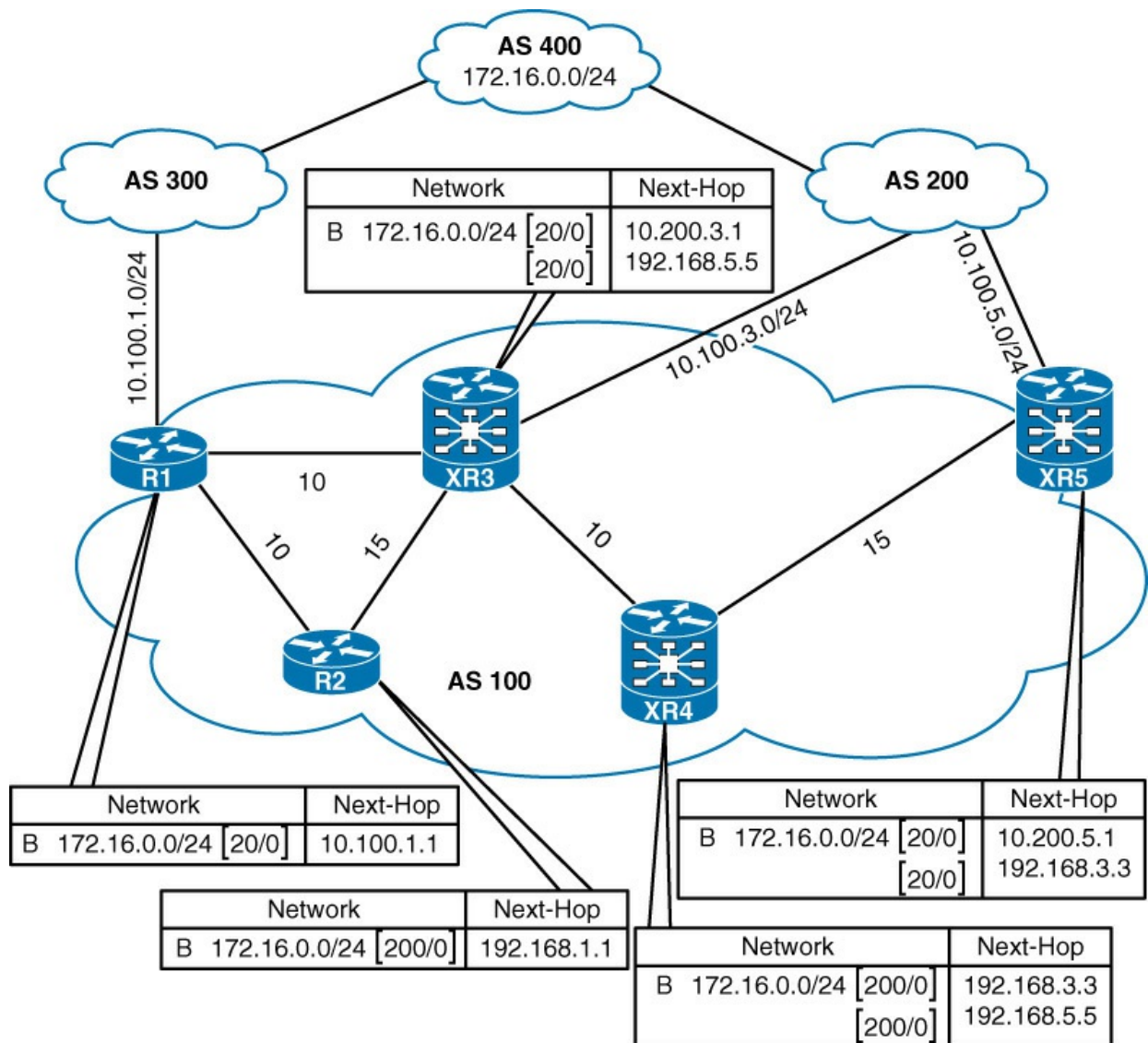


Figure 15-20 eBGP Multipath Topology

In the illustration, notice that R1 and R2 only installed one BGP path for the 172.16.0.0/24 network, whereas XR3, XR4, and XR5 installed multiple routes.

## R1

R1 identifies the eBGP path via 10.100.1.1 as the best path because eBGP paths are preferred over iBGP paths (see [Example 15-36](#)). In [Figure 15-20](#), notice that the other 172.16.0.0/24 paths are ineligible for multipath installation because the AS\_Path does not match the AS\_Path of the best path (300 400).

## Example 15-36 R1's BGP Table

[Click here to view code image](#)

```
R1#show bgp ipv4 unicast
! Output omitted for brevity
  Network          Next Hop          Metric LocPrf Weight Path
* i 172.16.0.0/24  192.168.3.3       100      0 200 400 i
* i                192.168.5.5       100      0 200 400 i
* i                10.100.1.1        0 300 400 i
```

## R2

XR2 identifies the iBGP path via R1 as the best path because the IGP metric to R1 is lower than the IGP metric to XR3 and XR5 (see [Example 15-37](#)). In [Figure 15-20](#), notice that the other 172.16.0.0/24 paths are ineligible for multipath because the AS\_Path does not match the AS\_Path of the best path (300 400).

### Example 15-37 R2's BGP Table

[Click here to view code image](#)

```
R2#show bgp ipv4 unicast
! Output omitted for brevity
      Network          Next Hop           Metric LocPrf Weight Path
* >i 172.16.0.0/24     192.168.1.1        0      100     0 300 400 i
* i                   192.168.3.3        0      100     0 200 400 i
* i                   192.168.5.5        0      100     0 200 400 i
```

## XR3

XR3 identifies the BGP best path via 10.100.3.1 as the best path because eBGP paths are preferred over iBGP paths (see [Example 15-38](#)). In [Figure 15-20](#), a second 172.16.0.0/24 path has an identical AS\_Path to the BGP best path (300 400) and can be installed into the RIB even though it uses an iBGP path.

### Example 15-38 XR3's BGP Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR3#show bgp ipv4 unicast
! Output omitted for brevity
      Network          Next Hop           Metric LocPrf Weight Path
* > 172.16.0.0/24     10.200.3.1         0      100     0 200 400 i
* i                   192.168.1.1        0      100     0 300 400 i
* i                   192.168.5.5        0      100     0 200 400 i
```

## XR4

XR4 identifies the BGP best path through XR3 because the IGP cost is lower than the path through XR5 (see [Example 15-39](#)). In [Figure 15-20](#), a second 172.16.0.0/24 path has an identical AS\_Path to the BGP best path (300 400) and can be installed into the RIB even though the IGP metric differs from the metric to XR3.

### Example 15-39 XR4's BGP Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show bgp ipv4 unicast
! Output omitted for brevity
      Network          Next Hop           Metric LocPrf Weight Path
* i172.16.0.0/24     192.168.1.1        0      100     0 300 400 i
* >i                 192.168.3.3        0      100     0 200 400 i
* i                 192.168.5.5        0      100     0 200 400 i
```

Processed 1 prefixes, 3 paths



## XR5

XR5 identifies the BGP best path via 10.200.5.1 as the as best path because the eBGP path is preferred over iBGP paths (see Example 15-40). In Figure 15-20, XR3's 172.16.0.0/24 path has an identical AS\_Path to the BGP best path (300 400) and can be installed into the RIB even though it uses an iBGP path.

### Example 15-40 XR5's BGP Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR5#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 172.16.0.0/24	10.200.5.1			0	200 400 i
* i	192.168.1.1	0	100	0	300 400 i
* i	192.168.3.3		100	0	200 400 i

```
Processed 1 prefixes, 3 paths
```

#### Note

The use of eIBGP multipath can introduce a routing loop into a topology and should be used only after mapping out all the traffic flows.

### AS\_Path Relax

BGP multipath works well if an organization uses the same service provider for Internet connectivity. However, if an organization uses different service providers, the AS\_Path will not be identical and will not meet the requirements for BGP multipath. The AS\_Path relax feature allows for BGP multipath to work when the AS\_Path length is the same but the AS\_Paths are different.

In Figure 15-21, AS400 connects through AS200 and AS300 for connectivity to AS100. BGP multipath will not work unless AS\_Path relax is configured on XR4 and R5.

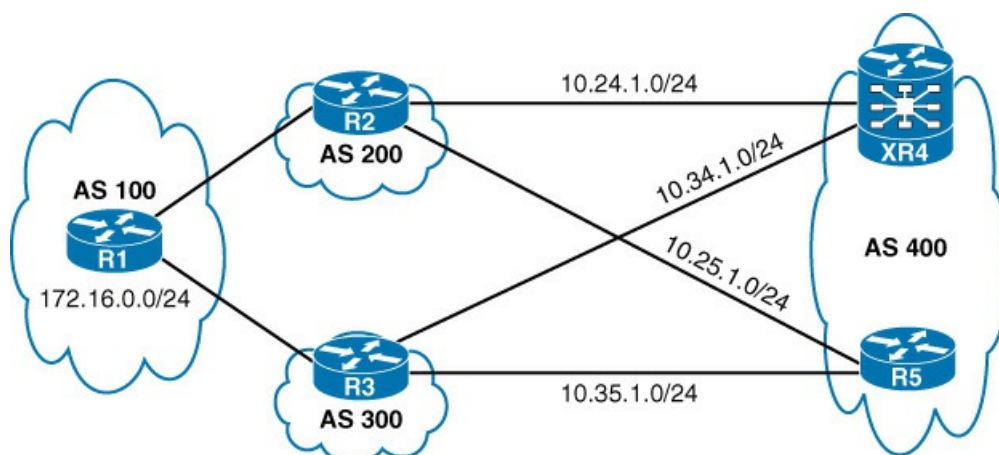


Figure 15-21 EIBGP Multipath Topology

The AS\_Path relax feature is configured with the BGP configuration command **bgp bestpath as-path multipath-relax** on IOS and IOS XR routers. Example 15-41 demonstrates the AS\_Path relax configuration for XR4 and R5.

### Example 15-41 AS\_Path Relax Configuration



[Click here to view code image](#)

```
IOS XR
router bgp 400
  bgp bestpath as-path multipath-relax
  !
  address-family ipv4 unicast
    maximum-paths ebgp 2
```

```
IOS
router bgp 400
  bgp bestpath as-path multipath-relax
  !
  address-family ipv4
    maximum-paths 2
```

#### Note

Some versions of IOS do not have a context-sensitive help for this command and require the complete command typed in correctly to function. Many network engineers consider this a hidden feature.

[Example 15-42](#) displays XR4's and R5's routing table, which confirms the eBGP multipath will work with different AS\_Paths with the AS\_Path relax feature enabled.

### Example 15-42 XR4's and R5's Routing Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show route bgp

B      172.16.0.0/24 [20/0] via 10.34.1.3, 00:03:04
                   [20/0] via 10.24.1.2, 00:03:04
```

```
R5#show ip route bgp

B          172.16.0.0 [20/0] via 10.35.1.3, 00:03:42
                   [20/0] via 10.25.1.2, 00:03:42
```

## SUBOPTIMAL ROUTING WITH ROUTE REFLECTORS

Large autonomous systems use route reflectors (RRs) to overcome the scalability issue with the iBGP full-mesh requirement. BGP advertises only the best path to other BGP peers, and RR placement in the autonomous system can potentially cause suboptimal routing.

Route reflectors do not pass all of the paths to a client. An RR only advertises the BGP best path, which is calculated based on the RR's location and not the RR client's location.

[Figure 15-22](#) demonstrates suboptimal routing because of path information loss due to RRs. AS100 advertises the 172.16.0.0/24 network to AS400's edge routers (R4 and R5). R4 and R5 advertise their route to RR1. RR1 computes the best path using IGP metric, and finds R5's path

as the best path. RR1 advertises only R5's path to R6. Looking at R6's routing table to R4 (192.168.4.4) and R5 (192.168.5.5), it is more optimal to send traffic directly to R4. Unfortunately, RR1 advertised only the path that used R5, so R4 will use the suboptimal IGP path to reach the 172.16.0.0/24 network. Notice that the BGP route on R6 does a recursive lookup to the R5's next-hop address.

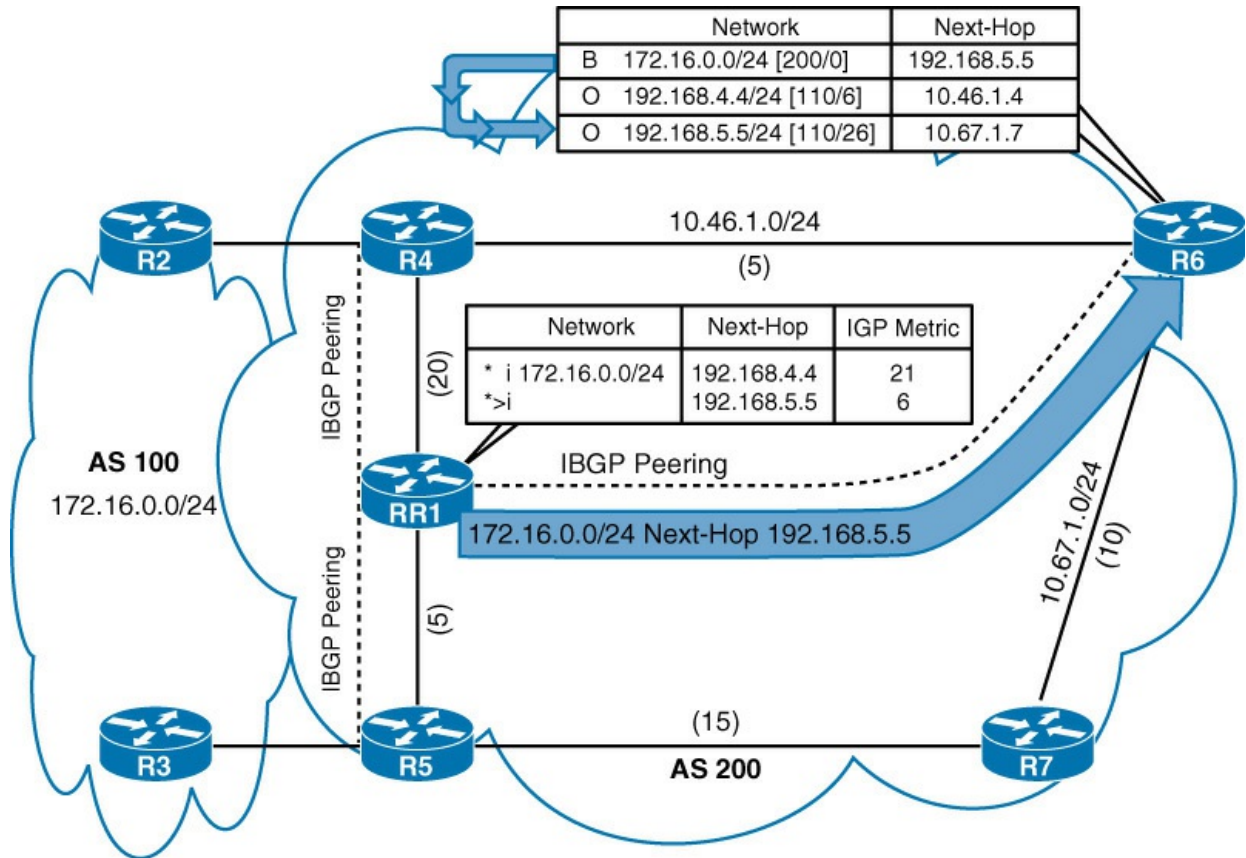


Figure 15-22 Suboptimal Routing

If both of the paths from R4 and R5 had been advertised to R6, R6 could decide the best path based on its IGP metric to reach each of the edge routers (R4 and R5). The following sections describe solutions for advertising multiple paths to downstream RR clients.

Note

This issue mainly deals with routes learned from another autonomous system. Routes that are advertised within the autonomous system will either have a source of 0.0.0.0 if the network is directly connected, or use the next-hop IP address from the static route or IGP when advertised into BGP.

### Additional Route Reflector

The simplest solution to avoid path information loss is to add additional RRs near the BGP edge routers for an autonomous system. In Figure 15-23, RR1 is closer to R4 than R5, and RR2 is closer to R5 than R4. RR1 and RR2 have different best paths and advertise them to R6. R6 receives both best paths and can select the most desirable exit point using its best path calculations.

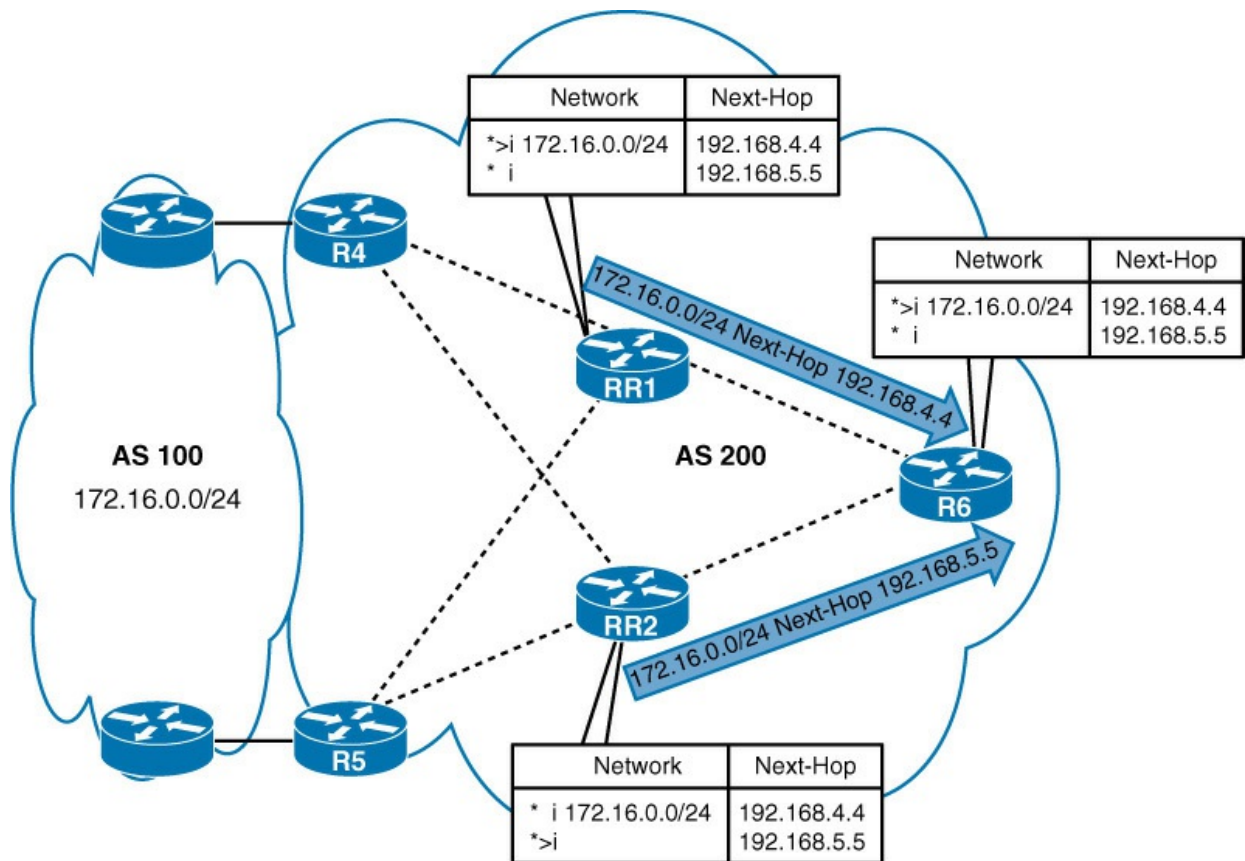


Figure 15-23 Additional Route Reflector

Unfortunately, this solution does not scale well. If there were a large number of peerings, the number of sessions between core routers and RRs would be high. There is an administrative burden for maintaining multiple RRs. The next topic provides a more scalable solution.

### Shadow Route Reflector

Deploying an additional RR into the same location introduces the concept of RR groupings or clusters. A group of RRs would form a *shadow RR cluster*, which does not require the same deployment ratio as deploying an RR for each BGP edge peering location. The second router reflector, known as a *shadow RR*, establishes BGP sessions like the first RR. Instead of advertising the best path from the second RR (shadow RR), BGP would use the *diverse path* feature to determine the second best path, known as the *backup path*, and advertise that route to its BGP peers. Only the backup path is advertised with the diverse path configuration, so two routers are required to advertise the best path and the backup path to the RR clients.

BGP diverse path was introduced in Cisco IOS 15.4(2)M and IOS XE 3.1.0S and is not supported on IOS XR. The BGP diverse path functionality was introduced in RFC 6774, but the only software changes that are required occur on the RR to support the backup path functionality.

Figure 15-24 demonstrates the BGP diverse path feature enabled on the shadow route reflector RR2 so that R6 receives the optimal path, and the suboptimal path in BGP.

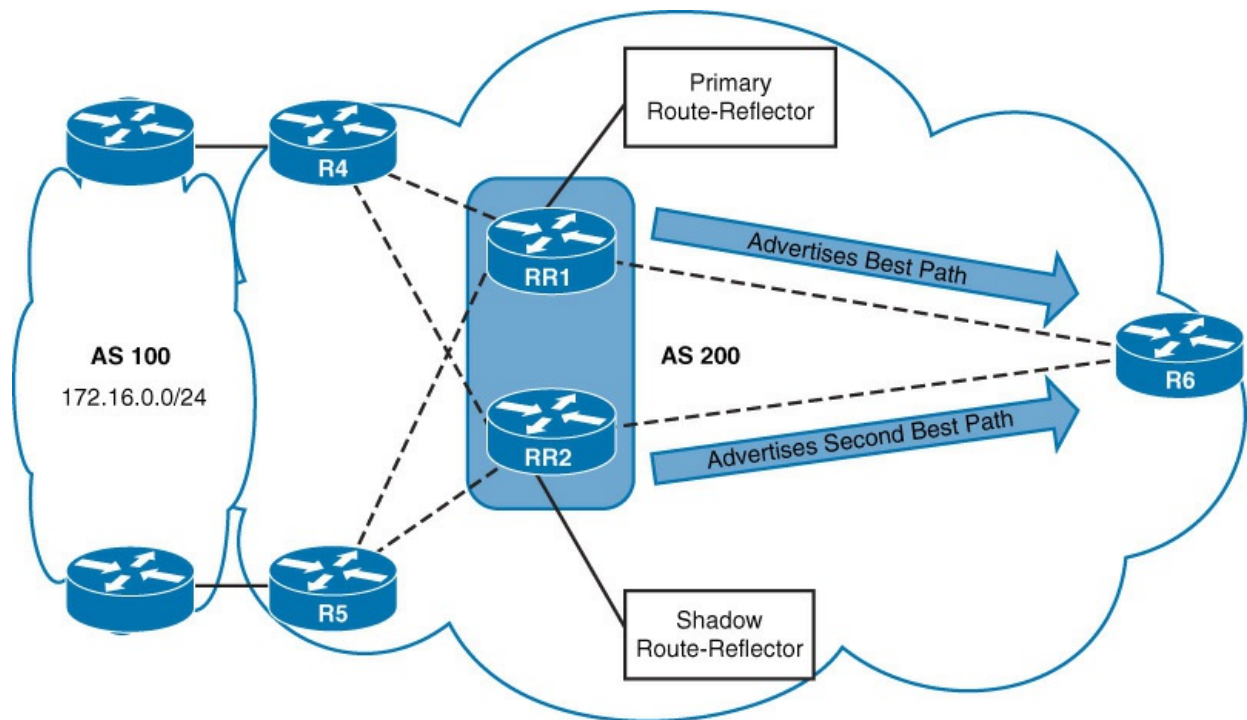


Figure 15-24 Shadow Route Reflector

The shadow RR needs to modify the BGP best path algorithm so that it calculates the backup path for the address family with the BGP address family configuration command **bgp additional-paths select backup**. Every neighbor session needs to be explicitly configured to receive the backup path in lieu of the best path. The command **neighbor ip-address advertise diverse-path backup** identifies the peers that should receive the backup path.

Example 15-43 demonstrates the commands executed on RR2 for advertisement of the backup path to R6.

### Example 15-43 Diverse Path Configuration

[Click here to view code image](#)

```
IOS#show bgp ipv4 unicast

router bgp 200
 neighbor 192.168.6.6 remote-as 200
 neighbor 192.168.6.6 update-source Loopback0
 !
 address-family ipv4
  bgp additional-paths select backup
  neighbor 192.168.6.6 advertise diverse-path backup
  neighbor 192.168.6.6 route-reflector-client
 exit-address-family
```

Example 15-44 displays R6's BGP table before and after diverse path was enabled on RR2 while the BGP session was inactive (shut down) on RR1. This reinforces the concept that the diverse path advertises only the backup path.

### Example 15-44 BGP Table Before and After Diverse Path Enablement

[Click here to view code image](#)

```
! R6's BGP table before enabling diverse-path on RR2
R6#show bgp ipv4 unicast
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i 172.16.0.0/24	192.168.5.5	0	100	0	100 i

```
! R6's BGP table after enabling diverse-path on RR2
R6#show bgp ipv4 unicast
! Output omitted for brevity
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i 172.16.0.0/24	192.168.4.4	0	100	0	100 i

Example 15-45 displays the BGP paths that are advertised to R6. Notice that the status codes of *b* for backup path and *a* for additional path are included.

### Example 15-45 RR2's Diverse Path Advertisement

[Click here to view code image](#)

```
RR2#show bgp ipv4 unicast neighbors 192.168.6.6 advertised-routes
BGP table version is 4, local router ID is 192.168.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*bia172.16.0.0/24	192.168.4.4	0	100	0	100 i

Total number of prefixes 1

#### Note

Both RRs in the shadow RR cluster should share equal distance to all the BGP edge routers to ensure that the best path calculations use identical metrics. If path costs differ, IGP metrics should be disabled with the command **bgp bestpath igp-metric ignore**.

### Shadow Session Route Reflector

Deploying a shadow RRs incurs the same costs and burdens as when deploying additional routers. An RR client needs at least two sessions to receive the best path and the backup path advertisement, but a session does not need to be sourced from different routers. The second session can be established between the RR and the RR client using different interfaces. The first session advertises the BGP best path normally, but the second session advertises the backup path to the RR client. This technique is known as using a *shadow session* on a RR.

Figure 15-25 demonstrates a shadow session in an RR topology. R1 establishes the first session sourced from Loopback 0 with R6's Loopback 0 interface for advertising the best path. R1 establishes a second shadow session sourced from Loopback 1 with R6's Loopback 1 interface for

advertising the backup path.

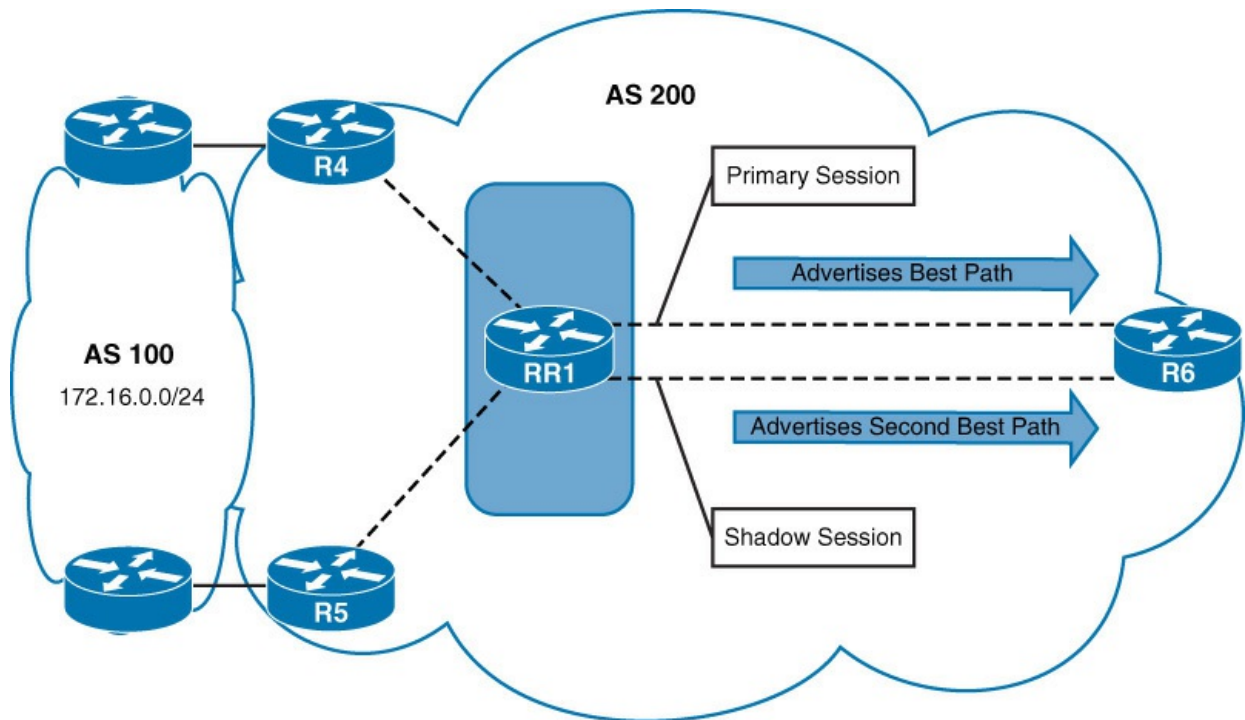


Figure 15-25 Shadow Session Route Reflector

**Example 15-46** demonstrates the shadow session configuration for RR1 and R6, where two sessions are established between RR1 and R6. Notice that the diverse path is added only to the second session between Loopback 1 interfaces.

### Example 15-46 Shadow Session RR Configuration

[Click here to view code image](#)

```
RR1
router bgp 200
 neighbor 192.168.6.6 remote-as 200
 neighbor 192.168.66.66 remote-as 200
 neighbor 192.168.6.6 update-source Loopback0
 neighbor 192.168.66.66 update-source Loopback1
!
address-family ipv4
  bgp additional-paths select backup
  neighbor 192.168.6.6 activate
  neighbor 192.168.6.6 route-reflector-client
  neighbor 192.168.66.66 activate
  neighbor 192.168.66.66 advertise diverse-path backup
  neighbor 192.168.66.66 route-reflector-client
exit-address-family
```

**R6**

```
router bgp 200
 neighbor 192.168.1.1 remote-as 200
 neighbor 192.168.11.11 remote-as 200
 neighbor 192.168.1.1 update-source Loopback0
 neighbor 192.168.11.11 update-source Loopback1
!
 address-family ipv4
  neighbor 192.168.1.1 activate
  neighbor 192.168.11.11 activate
 exit-address-family
```

**Note**

In most failure scenarios, if a BGP peer loses reachability to one loopback interface, it loses reachability to the second loopback interface too. This could be a single point of failure if only one RR is used with shadow sessions.

**BGP Add-Path**

The diverse path feature in BGP allows RR to overcome the loss of path information when an RR client calculates the best path. Only the RR needs to support diverse path functionality, as no modifications occurred within the BGP packet structure.

Through further evaluation, service providers identified that the advertisement of only one route (best path or backup path) per BGP session was insufficient to meet the needs of continuously growing number of services. In 2008, an informational RFC draft was released that overcomes the scalability issue of advertising one path for session using BGP additional paths (Add-Path). It adds extensions to BGP that allow BGP peers to advertise multiple paths for the same prefix.

The BGP Add-Path functionality requires that both BGP peers negotiate BGP Add-Path capabilities when the session is established and is the preferred solution for overcoming loss of path information from a RR. BGP Add-Path support was added to Cisco IOS in 15.3T, 3.10S in IOS XE, and 4.0.0 in IOS XR.

Figure 15-26 demonstrates the concept of BGP Add-Path. RR1 and R6 have negotiated Add-Path capability in their session. RR1 sends multiple paths, and R6 agrees to receive multiple paths per prefix. RR1 advertises both paths for the 172.16.0.0/24 network, and R6 receives both of them, and computes the best path from R6's local perspective.



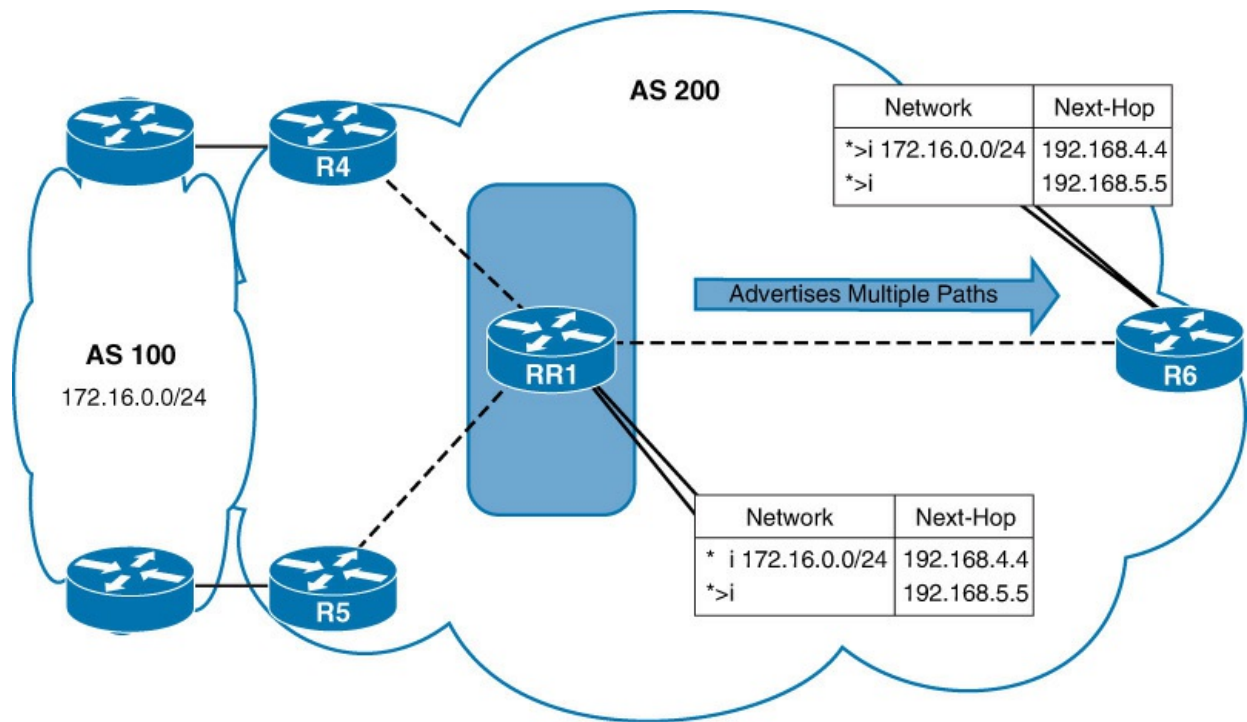


Figure 15-26 BGP Add-Path

The BGP Add-Path capability is enabled on BGP peers via two methods. The RR will *send* and the RR client will *receive*:

- On IOS and IOS XR routers, BGP Add-Path is enabled for all routers in the address family with the BGP address family configuration command **bgp additional-paths {send | receive}**.
- Add-Path functionality is enabled per neighbor:
- IOS routers use the BGP neighbor address family configuration command **neighbor ip-address additional-paths {disable | send [receive] | receive}**.
- IOS XR routers use the BGP neighbor session configuration command **capability additional-paths {send | receive} [disable]**.

Note

The BGP session has to establish after configuring BGP Add-Path functionality and may need to be hard reset so that the routers can negotiate the BGP Add-Path capability. BGP Add-Path functionality can be enabled bidirectionally even though it is shown with unidirectional advertisements.

The advertising router must identify the additional paths that are can be sent to the RR client:

- IOS routers use the BGP address family configuration command:
- **bgp additional-paths select best number**: The best 2 or best 3 command option means that the best path along with the second or second and third best paths are advertised to the neighbor router.
- **bgp additional-paths select group-best**: The group-best option chooses the best path



option from among the same autonomous system. So if there are multiple paths to reach a prefix from an autonomous system, only one path is selected. If there are multiple paths to reach the prefix from two neighboring autonomous systems, two best paths are chosen, one best path for each autonomous system.

■ **bgp additional-paths select all:** All paths with a unique next-hop forwarding address are advertised to the neighbor

■ In IOS XR, the address family command **additional-paths selection route-policy route-policy-name** advertises the additional paths. The route policy may match a specific prefix or all prefixes based on the match criteria. Within the RPL, the Add-Path capability is set with the command **set path-selection {backup number | group-best | all | best-path} advertise**. IOS XR allows only the advertisement of the best and second best path for a prefix. The RPL policy action has the following options in the route policy:

■ **Backup:** The second best path

■ **Group-best:** The best path for each neighboring autonomous system

■ **All:** All BGP paths

■ **Best-path:** Only the best path

IOS RRs require a third step, which is to activate the additional path advertisement per neighbor with the command **neighbor ip-address advertise additional-paths [best number] [group-best] [all]**. This step is not required for IOS XR routers.

Example 15-47 demonstrates the BGP Add-Path configuration for IOS routers.

### Example 15-47 BGP Add-Path Configuration for IOS Routers

[Click here to view code image](#)

```
IOS Route Reflector
router bgp 200
!
address-family ipv4
  bgp additional-paths select all
  bgp additional-paths send
  neighbor 192.168.6.6 route-reflector-client
  neighbor 192.168.6.6 advertise additional-paths all
exit-address-family
```

```
IOS Route Reflector Client
router bgp 200
!
address-family ipv4
  bgp additional-paths receive
exit-address
```

[Example 15-48](#) demonstrates the BGP Add-Path configuration for IOS XR routers.

### **Example 15-48** BGP Add-Path Configuration for IOS XR Routers

[Click here to view code image](#)

#### **IOS XR Route Reflector**

```
router bgp 200
  address-family ipv4 unicast
    additional-paths send
    additional-paths selection route-policy ALLPATHS
  !
  route-policy ALLPATHS
    set path-selection all advertise
  end-policy
```

#### **IOS XR Route Reflector Client**

```
router bgp 200
  address-family ipv4 unicast
    additional-paths receive
  !
  neighbor 192.168.1.1
    remote-as 200
    update-source Loopback0
    address-family ipv4 unicast
```

The BGP Add-Path feature is an essential component to the BGP prefix-independent convergence (PIC) feature and is explained in more detail in [Chapter 21](#).

## **SUMMARY**

BGP does not use metrics to identify the best path in a network. BGP uses path attributes to identify the best path on a router. BGP's best path selection algorithm compares the paths in the following order:

- 1. Weight**
- 2. Local preference**
- 3. Local originated (**network** statement, redistribution, aggregation)**
- 4. AIGP**
- 5. Shortest AS\_Path**
- 6. Origin type**
- 7. Lowest MED**
- 8. eBGP over iBGP**

9. Lowest IGP next hop
10. If both paths are external (eBGP), prefer the first (oldest)
11. Prefer the route that comes from the BGP peer with the lower RID
12. Prefer the route with the minimum cluster list length
13. Prefer the path that comes from the lowest neighbor address

BGP path attributes can be modified upon receipt or advertisement to influence routing in the local autonomous system or neighboring autonomous system. A basic rule for traffic engineering with BGP is that modifications in an outbound routing policy influence inbound traffic, and modifications to inbound routing policies influence outbound traffic.

BGP supports three types of ECMP: eBGP multipath, iBGP multipath, or eiBGP multipath. eBGP multipath requires that the weight, local preference, AS\_Path length, AS\_Path content, origin, and MED match for a second route to install into the RIB.

Route reflectors overcome the scalability issues associated with iBGP full-mesh requirements, but introduce the potential of suboptimal routing due to path loss. RRs identify the BGP best path based on their location to the edge BGP routers, and not the router that the prefix is being advertised to. BGP diverse path allows for the advertisement of a backup path from a second BGP session and requires only the RR to support the feature. BGP Add-Path provides multiple path advertisements within one session, but requires that the RR and the RR client support the feature.

## FURTHER READING

This chapter completes all the topics involving BGP. Successfully deploying BGP requires knowing the history of the Internet and some of the design theories when working with BGP. The following books will reinforce the concepts covered in this book:

Halabi, Sam. *Internet Routing Architectures*. Indianapolis: Cisco Press, 2000.

Zhang, Randy and Micah Bartell. *BGP Design and Implementation*. Indianapolis: Cisco Press 2003.

White, Russ, Alvaro Retana, Don Slice. *Optimal Routing Design*. Indianapolis: Cisco Press, 2005.

## REFERENCES IN THIS CHAPTER

RFC 1966, *BGP Route Reflection, An alternative to full mesh iBGP*, T. Bates, R. Chandra, <http://www.ietf.org/rfc/rfc1966.txt>, June 1996

RFC 4451, *BGP MULTI\_EXIST\_DISC (MED) Considerations*, D. McPherson, V. Gill, <http://www.ietf.org/rfc/rfc4451.txt>, March 2006

RFC 6774, *Distribution of Diverse BGP Paths*, D. McPherson, et al., <http://tools.ietf.org/html/rfc6774>, November 2012

Informational RFC, *The Accumulated IGP Metric Attribute for BGP*, P. Mohapatra, et al., <http://tools.ietf.org/html/draft-ietf-idr-aigp-18>, April 2014

Informational RFC, *Advertisement of Multiple Paths in BGP*, D. Walton, et al., <http://tools.ietf.org/html/draft-ietf-idr-add-paths-09>, October 2013

Doyle, Jeff and Jennifer Carrol. *Routing TCP/IP Volume I, 2nd Edition*. Indianapolis: Cisco Press, 2006.

Luc De Ghein. "Scaling BGP" (presented at Cisco Live, Milan, 2014).

Cisco. Cisco IOS Software Configuration Guides. <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides. <http://www.cisco.com>

## Part V: Multicast

## Chapter 16. IPv4 Multicast Routing

This chapter covers the following topics:

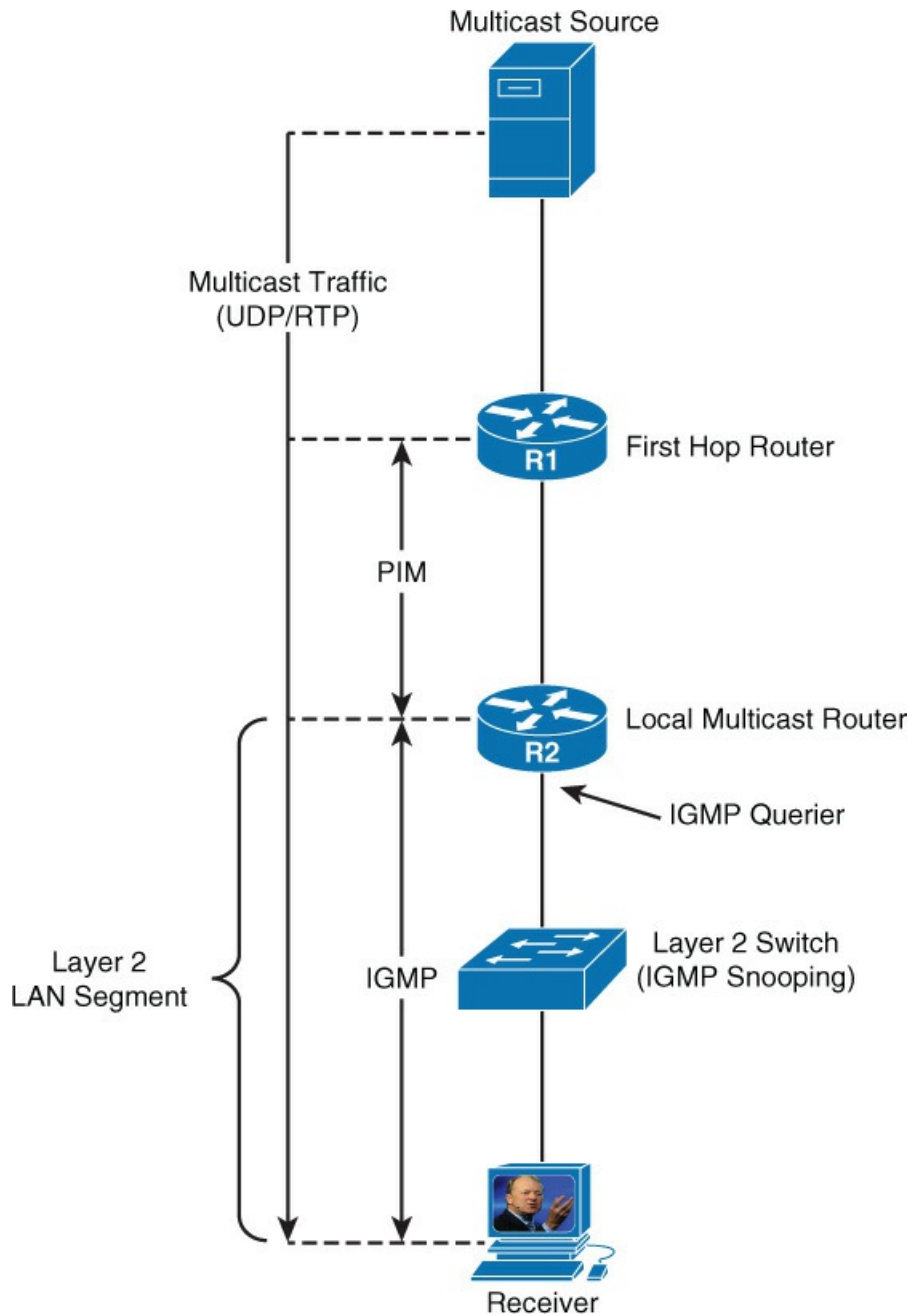
- Multicast fundamentals
- Internet Group Messaging Protocol (IGMP)
- Multicast distribution trees
- Protocol Independent Multicast (PIM)
- Rendezvous points (RPs)
- Bidirectional PIM

Traditional IP communication between network hosts occurs on a one-to-one (unicast) or a one-to-all (broadcast) basis. A third option allows for one-to-many or many-to-many communication, known as *multicast*, which allows a host to send data packets to a group of hosts (receivers).

### MULTICAST FUNDAMENTALS

Multicast communication is a technology that optimizes network bandwidth utilization and conserves server load. It relies on Internet Group Management Protocol (IGMP) and Protocol Independent Multicast (PIM) for its operation in Layer 2 and Layer 3 networks.

Figure 16-1 illustrates that IGMP operates between the receivers and the local multicast router and that PIM operates between routers. These two technologies work hand in hand to allow multicast traffic to flow from the source to the receivers, and they are explained in this chapter.

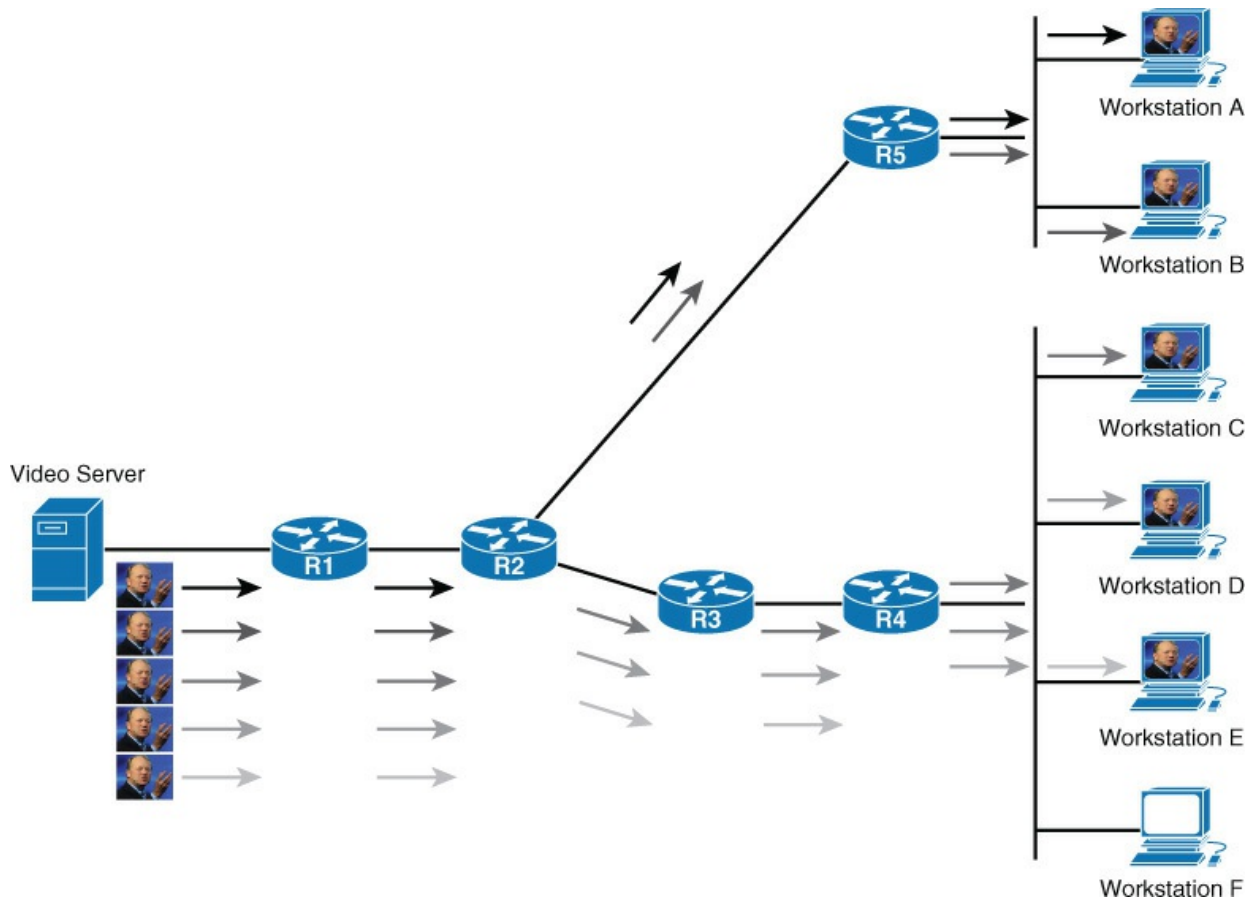


**Figure 16-1** Multicast Architecture

Note

The switch shown in [Figure 16-1](#) will not be shown in the rest of the figures in this chapter unless it is necessary to describe a specific concept.

[Figure 16-2](#) shows an example where six workstations are watching the same video that is advertised by a server using unicast traffic (one to one). Each arrow represents a data stream of the same video going to five different hosts. If each stream is 10 Mbps, the network link between R1 and R2 would need 50 Mbps of bandwidth. The network link between R2 and R4 would require 30 Mbps of bandwidth, and the link between R2 and R5 would require 20 Mbps of bandwidth. The server must maintain session state information for all the sessions between the hosts. The bandwidth and load on the server increases as more receivers request the same video feed.

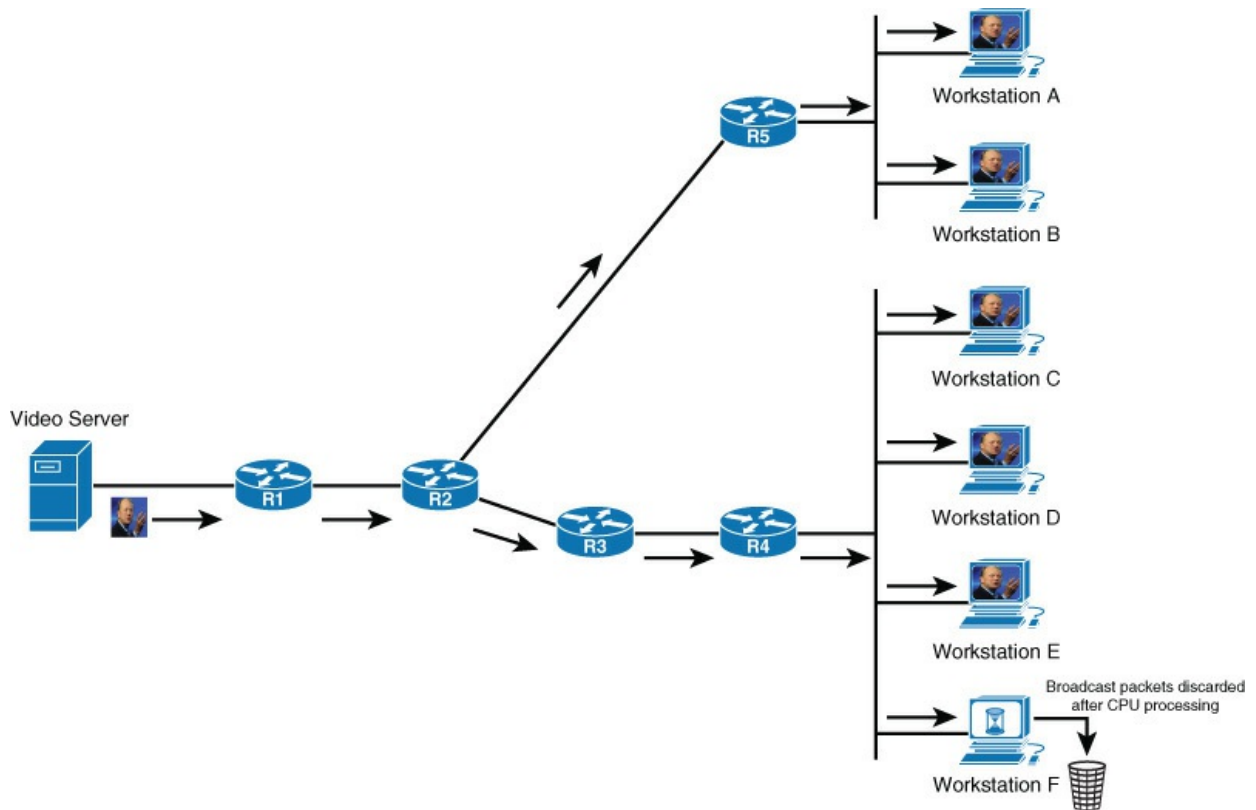


**Figure 16-2** *Unicast Video Feed*

An alternative method for all five workstations to receive the video is to send it from the server via broadcast traffic (one to all). [Figure 16-3](#) illustrates an example of how the same video stream is transmitted using IP directed broadcasts. The load on the server is reduced because it needs to maintain only one session state versus many. The same video stream consumes only 10 Mbps of bandwidth on all network links. However, this approach does have disadvantages:

- IP directed broadcast functionality is not enabled by default on Cisco routers, and enabling it exposes the router to distributed denial-of-service (DDoS) attacks.
- The network interface cards (NICs) of uninterested workstations must still process the broadcast packets and send on to the workstation's CPU, which wastes processor resources. In [Figure 16-3](#), Workstation F is processing unwanted packets.



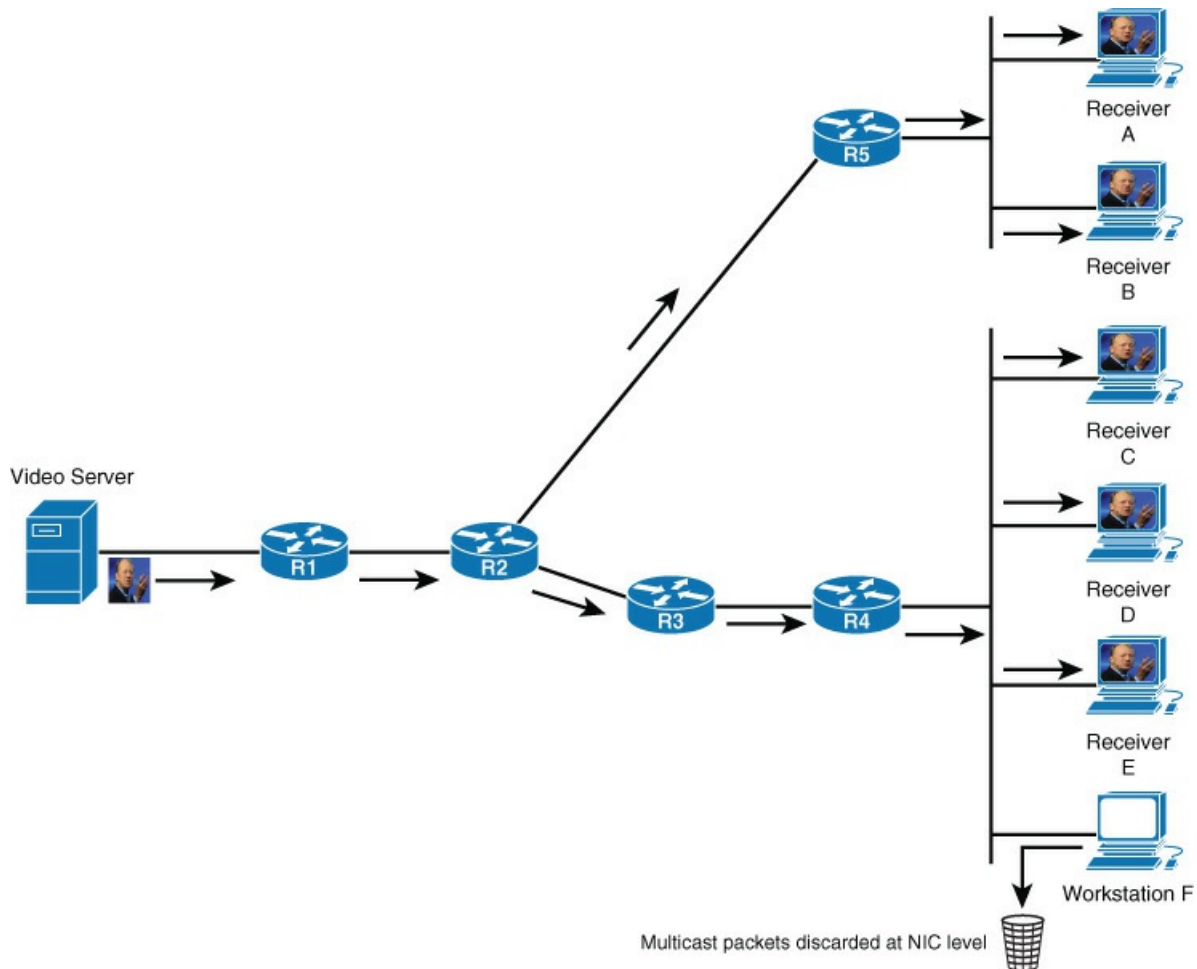


**Figure 16-3** Broadcast Video Feed

For these reasons, broadcast traffic is generally not recommended.

Multicast traffic provides one-to-many communication, where only one data packet is sent on a link as needed and then is replicated between links as the data forks (splits) on a network device along the multicast distribution tree. The data packets are known as a stream that use a special destination IP address, known as a *group address*. Servers for a stream still manage only one session, and network devices selectively request to receive the stream. Recipient devices of a multicast stream are known as receivers. Common applications that take advantage of multicast traffic include video, IPTV, stock ticker, distance learning, video/audio conferencing, music on hold, gaming, and so on.

Figure 16-4 shows an example of the same video feed using multicast. All network links consume only 10 Mbps of bandwidth, similar to broadcast traffic, but the difference is that only receivers that want the video stream process the multicast traffic. Workstation F would drop the multicast traffic at the NIC level because it would not be programmed to accept the multicast traffic.



**Figure 16-4** Multicast Video Feed

Note

Workstation F would not receive any multicast traffic if the switch for that network segment enables Internet Group Management Protocol (IGMP) snooping. IGMP is covered in more detail later in this chapter.

## MULTICAST ADDRESSING

The Internet Assigned Number Authority (IANA) assigned the IP Class D address space 224.0.0.0/4 for multicast addressing. Addresses in the Class D range from 224.0.0.0 to 239.255.255.255. This first 4 bits of this whole range start with 1110. In the multicast address space, multiple blocks of addressing are reserved for specific purposes, as shown in [Table 16-1](#).

<b>Designation</b>	<b>Multicast Address Range</b>
Local Network Control Block	224.0.0.0 – 224.0.0.255
Internetwork Control Block	224.0.1.0 – 224.0.1.255
AD-HOC Block I	224.0.2.0 – 224.0.255.255
Reserved	224.1.0.0 – 224.1.255.255
SDP/SAP Block	224.2.0.0 – 224.2.255.255
AD-HOC Block II	224.3.0.0 – 224.4.255.255
Reserved	224.5.0.0 – 224.255.255.255
Reserved	225.0.0.0 – 231.255.255.255
Source-Specific Multicast Block	232.0.0.0 – 232.255.255.255
GLOP Block	233.0.0.0 – 233.255.255.255
AD-HOC Block III	233.252.0.0 – 233.255.255.255
Reserved	234.0.0.0 – 238.255.255.255
Administratively Scoped Block	239.0.0.0 – 239.255.255.255

**Table 16-1** IP Multicast Addresses Assigned by IANA

Out of the multicast blocks mentioned in [Table 16-1](#), the most important are discussed in the list that follows:

- **Local Network Control Block (224.0.0/24):** Addresses in the Local Network Control Block are used for protocol control traffic that is not forwarded out of a broadcast domain. An example of this type of multicast control traffic is the *All Hosts in this subnet* (224.0.0.1), *All routers in this subnet* (224.0.0.2), and the *All PIM routers* (224.0.0.13).

- **Internetwork Control Block (224.0.1.0/24):** Addresses in the Internetwork Control Block are used for protocol control traffic that may be forwarded through the Internet. Examples include Network Time Protocol (NTP) (224.0.1.1), Cisco-RP-Announce (224.0.1.39), and Cisco-RP-Discovery (224.0.1.40).

[Table 16-2](#) lists some of the well-known Local Network Control Block and Internetwork Control Block multicast addresses.

IP Multicast Address	Description
224.0.0.0	Base address (reserved)
224.0.0.1	All hosts in this subnet (all-hosts group)
224.0.0.2	All routers in this subnet
224.0.0.5	All OSPF routers (AllSPFRouters)
224.0.0.6	All OSPF DRs (AllDRouters)
224.0.0.9	All RIPv2 routers
224.0.0.10	All EIGRP routers
224.0.0.13	All PIM routers
224.0.0.18	VRRP
224.0.0.22	IGMPv3
224.0.0.102	HSRPv2 and GLBP
224.0.1.1	NTP
224.0.1.39	Cisco-RP-Announce (Auto-RP)
224.0.1.40	Cisco-RP-Discovery (Auto-RP)

Table 16-2 Well-Known Reserved Multicast Addresses

■ **Source-Specific Multicast Block (232.0.0.0/8):** SSM, described in RFC 4607, is an extension of Protocol Independent Multicast (PIM), described later in this chapter, in which traffic is forwarded to receivers from only those multicast sources for which the receivers have explicitly expressed interest and is primarily targeted to one-to-many applications.

■ **GLOP Block (233.0.0.0/8):** Addresses in the GLOP Block are globally scoped statically assigned addresses. The assignment is made for domains with a 16-bit autonomous system number (ASN), by mapping the domain's ASN, expressed in octets as X.Y, into the middle two octets of the GLOP Block, yielding an assignment of 233.X.Y.0/24. The mapping and assignment is defined in RFC 3180. Domains with a 32-bit ASN may apply for space in AD-HOC Block III or consider using IPv6 multicast addresses.

■ **Administratively Scoped Block (239.0.0.0/8):** These addresses, described in RFC 2365, *Administratively Scoped IP Multicast*, are limited to a local group or organization. These addresses are similar to the reserved IP unicast ranges (such as 10.0.0.0/8) defined in RFC 1918 and will not be assigned by the IANA to any other group or protocol. In other words, network administrators are free to use multicast addresses in this range inside of their domain without worry of conflicting with others elsewhere in the Internet.

### Layer 2 Multicast Addresses

Historically, NICs on a LAN segment could receive only packets destined for their burned-in MAC address or the broadcast MAC address. Using this logic can cause a burden on routing resources during packet replication for LAN segments. Another method for multicast traffic was created so that replication of multicast traffic did not require packet manipulation, and a method of using a common destination MAC address was created.

A MAC address is a unique value associated with a NIC that is used to uniquely identify the NIC

on a LAN segment. MAC addresses are 12-digit hexadecimal numbers (48 bits in length) and they are typically stored in 8-bit segments separated by hyphens (-) or colons (:). (for example, 00-12-34-56-78-00 or 00:12:34:56:78:00).

Every multicast group address (IP address) is mapped to a special MAC address that allows Ethernet interfaces to identify multicast packets to a specific group. A LAN segment can have multiple streams, and a receiver knows which traffic to send to the CPU for processing based on the MAC address assigned to the multicast traffic.

The first 24 bits of a multicast MAC address always start with 01:00:5E, the low-order bit of the first byte is the Individual/Group bit (I/G) bit, also known as the *Unicast/Multicast bit*, and when it is set to 1 it indicates the frame is a multicast frame and the 25th bit is always 0. The lower 23 bits of the multicast MAC are copied from the lower 23 bits of the multicast group IP address.

Figure 16-5 shows an example of mapping the multicast IP address 239.255.1.1 into multicast MAC address 01:00:5E:7F:01:01. The first 25 bits are always fixed; what varies are the last 23 bits that are copied directly from the multicast IP address.

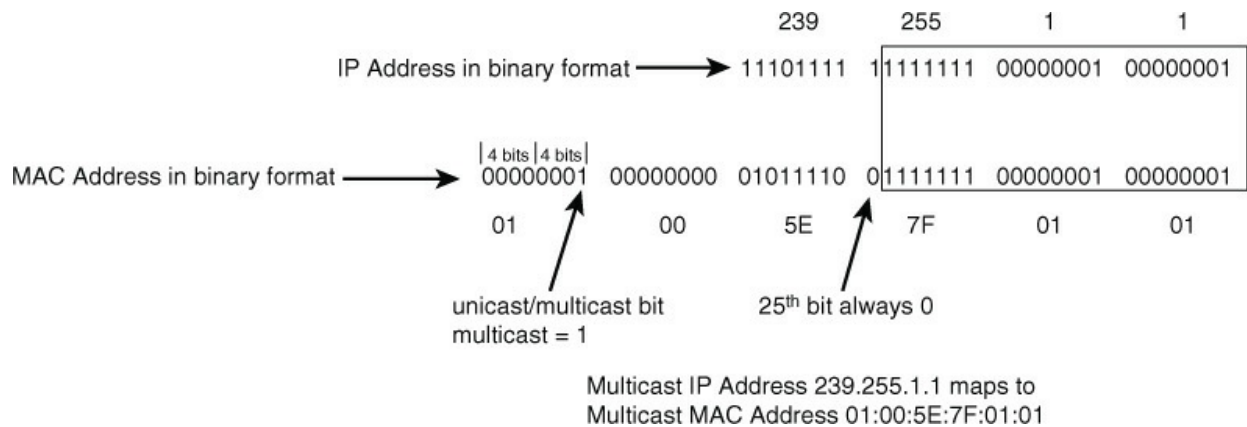
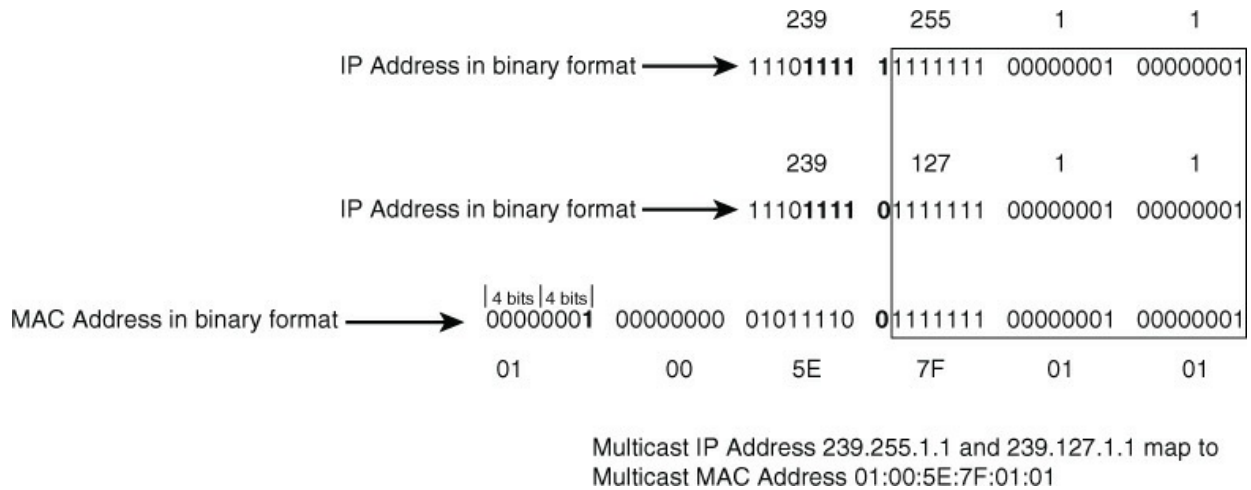


Figure 16-5 Multicast IP Address to Multicast MAC Address Mapping

Out of the 9 bits from the multicast IP address that are not copied into the multicast MAC address, the high-order bits 1110 are fixed; that leaves 5 bits that are variable that are not transferred into the MAC address. Because of this reason, there are 32 ( $2^5$ ) multicast IP addresses that are not universally unique and could correspond to a single MAC address. Figure 16-6 shows an example of two multicast IP addresses that map to the same multicast MAC address.



**Figure 16-6** Multicast IP Address to Multicast MAC Address Mapping Overlap

When a receiver wants to receive a specific multicast feed, it sends an IGMP join using the multicast IP group address for that feed. The receiver reprograms its interface to accept the multicast MAC group address that correlates to the group address. For example, a PC could send a join to 239.255.1.1 and would reprogram its NIC to receive 01:00:5E:7F:01:01. If the PC were to receive an OSPF update sent to 224.0.0.5 and its corresponding MAC 01:00:5E:00:00:05, it would ignore it and eliminate wasted CPU cycles on an undesired multicast traffic.

## INTERNET GROUP MANAGEMENT PROTOCOL

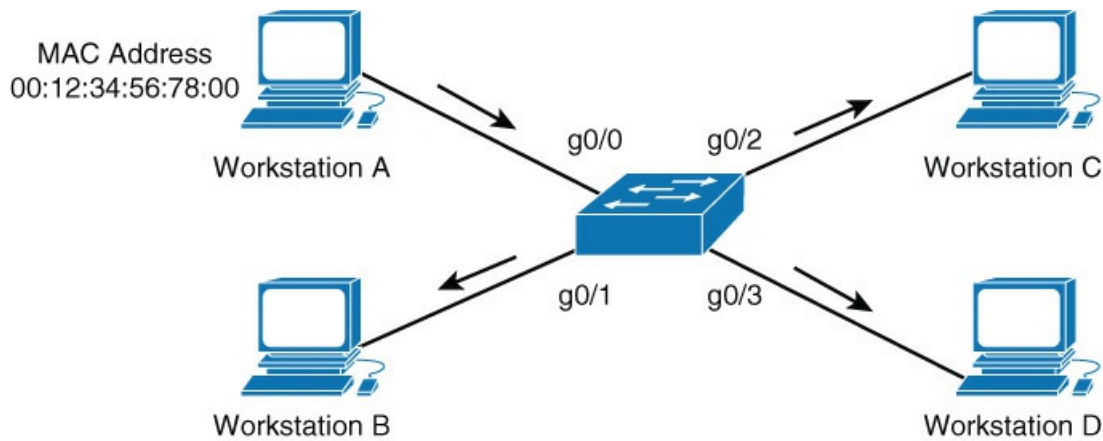
Internet Group Management Protocol (IGMP) is the protocol used by receivers to join multicast groups and start receiving traffic from those groups. IGMP must be supported by receivers and the router interfaces facing the receivers. When a receiver wants to receive multicast traffic from a source, it sends an IGMP join to its router. If the router does not have IGMP enabled on the interface, the request is ignored.

Three versions of IGMP exist. RFC 1112 defines IGMPv1, which is old and rarely used. RFC 2236 defined IGMPv2, which is common in most multicast networks, and IGMPv3 was defined in RFC 3376. Only IGMPv2 and IGMPv3 are described in this chapter.

### IGMP Snooping

Cisco switches learn about Layer 2 MAC addresses and what ports they belong to by inspecting the Layer 2 MAC address source and they store this information in the MAC address table. If they receive a Layer 2 frame with a destination MAC that is not in this table, they treat it as an unknown frame and flood it out all of the ports within the same VLAN except the interface the frame was received on. Uninterested workstations will notice the destination MAC in the frame is not theirs and will discard the packet.

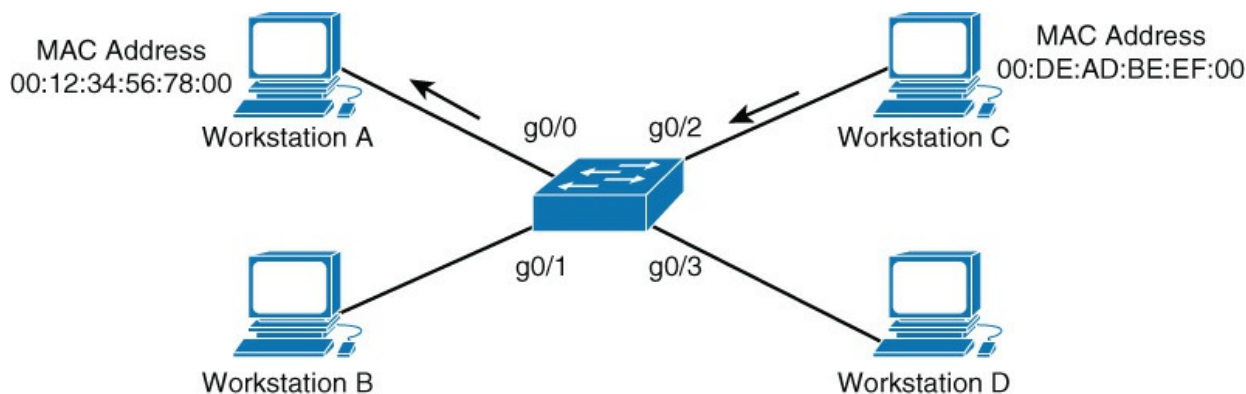
In [Figure 16-7](#), Switch 1 starts with an empty MAC address table. When Workstation A sends a frame, it stores its source MAC address and interface in the MAC address table and flood the frame it received out all ports.



Switch MAC Address Table	
MAC Address	Interface
00:12:34:56:78:00	g0/0

**Figure 16-7** Unknown Frame Flooding

If any other workstation sends a frame destined to the MAC address of Workstation A, the frame would not be flooded anymore because it's already in the MAC address table and it would be sent only to Workstation A, as shown in Figure 16-8.



Switch MAC Address Table	
MAC Address	Interface
00:12:34:56:78:00	g0/0
00:DE:AD:BE:EF:00	g0/2

**Figure 16-8** Known Destination Is Not Flooded

Because a multicast MAC address is never used as a source MAC address, they qualify as unknown frames and are flooded out all ports, and all workstations process these frames. It is then up to the workstations to select interested frames for processing and the frames that should be discarded. The flooding of multicast traffic on a switch wastes bandwidth utilization on each LAN segment.

Cisco switches use three methods to reduce multicast flooding on a LAN segment:

- IGMP Snooping
- Cisco Group Management Protocol (CGMP)
- Creating static MAC address entries

IGMP snooping, defined in RFC 4541, is the most widely used method and works by examining the IGMP join sent by a receiver and maintaining a table of interfaces to IGMP joins. When the switch receives a multicast frame destined for that multicast group, it forwards the packet only out the ports where the IGMP joins were received.

Figure 16-9 illustrates Workstation A and Workstation B sending IGMP joins to 239.255.1.1, which translates to the multicast MAC address 01:00:5E:7F:01:01. Switch 1 has IGMP snooping enabled and populates the MAC address table with this information.

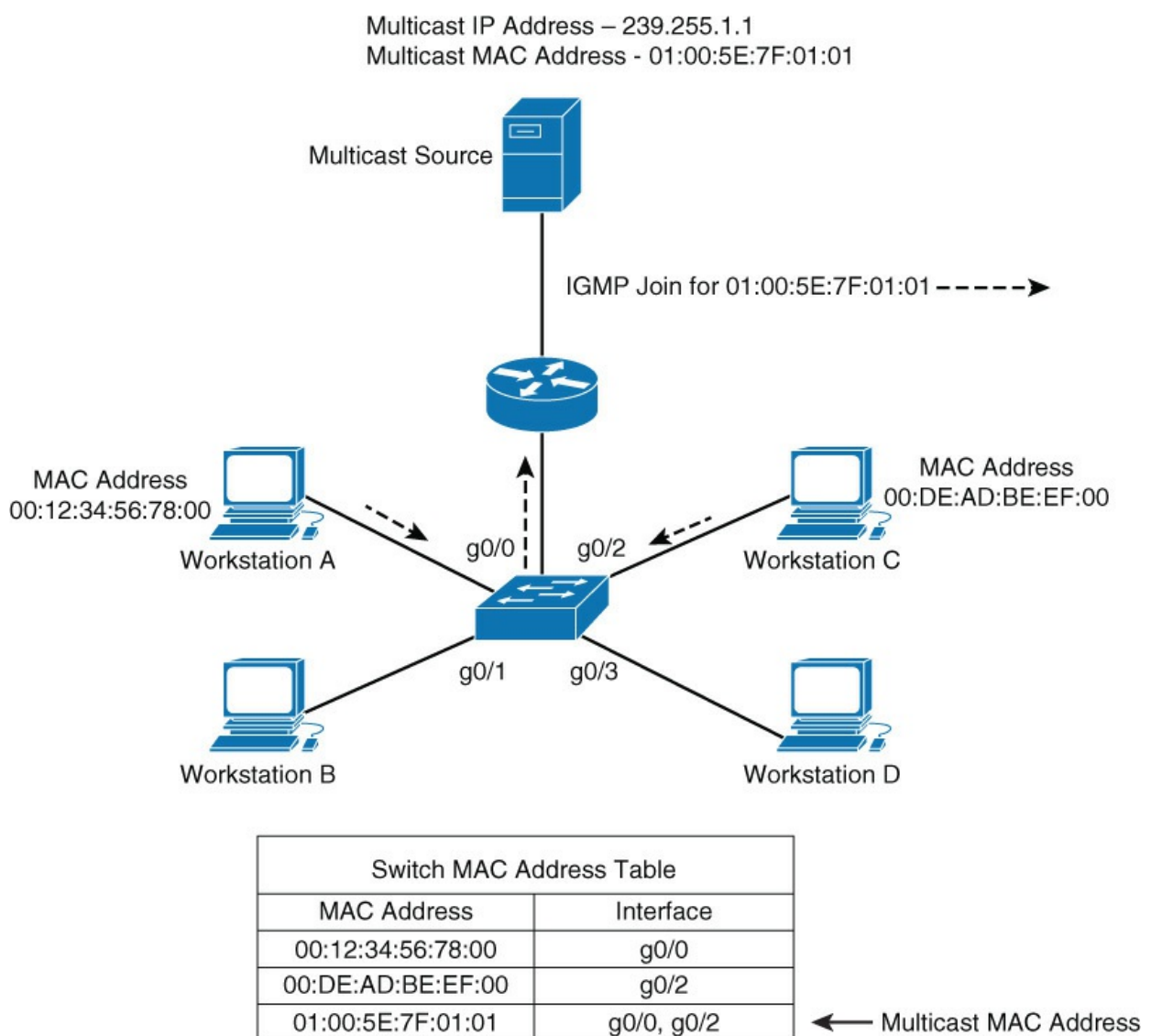
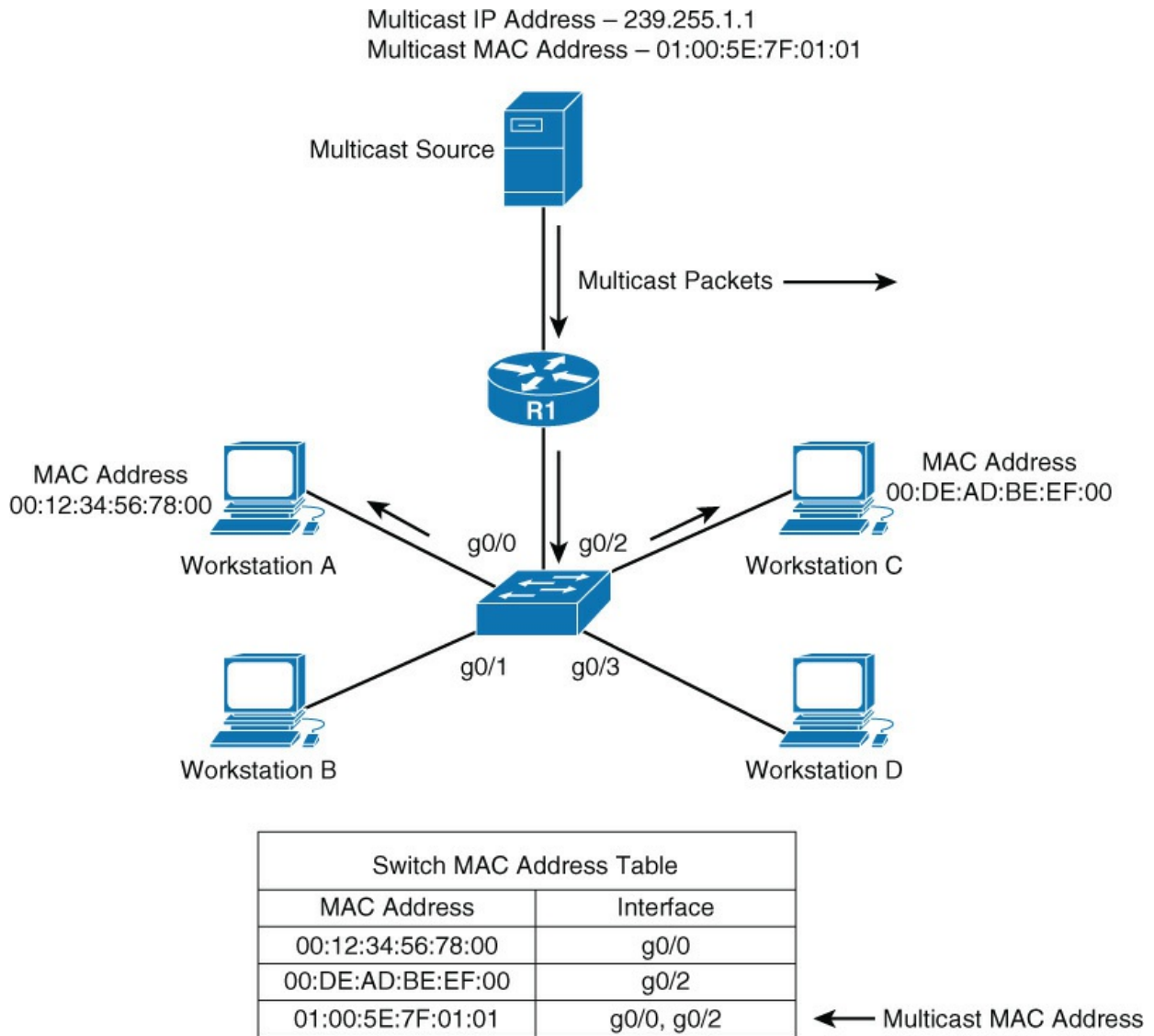


Figure 16-9 IGMP Snooping Example

Figure 16-10 illustrates the source sending traffic to 239.255.1.1(01:00:5E:7F:01:01). Switch 1 receives this traffic, and it forwards it out only the g0/0 and g0/2 interfaces because those are the only ports that received IGMP joins for that group.





**Figure 16-10** No Flooding with IGMP Snooping

CGMP is a Cisco proprietary mechanism that works in a similar fashion to IGMP snooping. IGMP snooping is preferred because it is based on open standards.

A multicast static entry can manually be programmed into the MAC address table but is not a scalable solution and cannot react dynamically to changes; for this reason, it is not a recommended approach.

### IGMPv2

IGMPv2 uses the message format shown in [Figure 16-11](#). This message is encapsulated in an IP packet with a protocol number of 2. They are sent with the IP Router Alert option set, which indicates this packet should be examined more closely and a Time-To-Live (TTL) of 1. TTL is an 8 bit field in an IP packet header that is set by the sender of the IP packet and decremented by every router on the route to its destination. When the TTL reaches 0 before reaching the destination, the packet is discarded. IGMP packets are sent with a TTL of 1 so that packets are processed by the local router and not forwarded by any router.

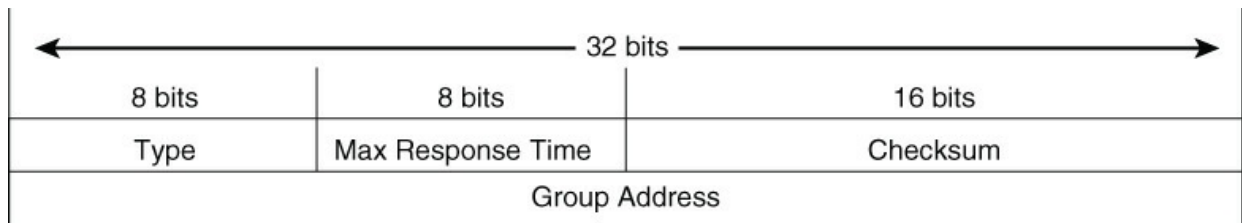


Figure 16-11 IGMP Message Format

The fields are defined as follows

- **Type** describes five different types of IGMP messages used by routers and receivers:
  - **Version 2 membership report** (0x16) is a message type also known as an *IGMP join*, and it is used by receivers to join a multicast group or to respond to a local router's membership query message.
  - **Version 1 membership report** (0x12) is used by receivers for backward compatibility with IGMPv1.
  - **Version 2 leave group** (0x17) is used by receivers to indicate they want to stop receiving multicast traffic for a group they joined.
  - **General membership query** (0x11) is sent periodically sent to the all-hosts group address 224.0.0.1 to see whether there are any receivers in the attached subnet. It sets the group address field to 0.0.0.0.
  - **Group specific query** (0x11) is sent in response to a leave group message to the group address the receiver requested to leave. The group address is the destination IP address of the IP packet and the group address field.
  - **Max response time** is set only in membership query messages and specifies the maximum allowed time before sending a responding report in units of one-tenth of a second. In all other messages, it is set to 0x00 by the sender and ignored by receivers.
  - **Checksum** is the 16-bit 1s complement of the 1s complement sum of the IGMP message. This is the standard checksum algorithm used by TCP/IP.
  - **Group address** is set to 0.0.0.0 in general query messages and is set to the group address in group-specific messages. Membership report messages carry the address of the group being reported in this field; group leave messages carry the address of the group being left in this field.

When a receiver wants to receive a multicast stream, it sends an unsolicited membership report, also known as an IGMP join, to the local router for the group it wants to join (for example, 239.1.1.1). The local router then sends this request upstream toward the source. When the local router starts receiving the multicast stream, it forwards it downstream to the subnet where the receiver that requested it resides.

The router then starts periodically sending general membership query messages into the subnet

sent to the all-hosts group address 224.0.0.1 to see whether any members are in the attached subnet. The general query message contains a Max Response Time field that is set to 10 seconds by default. This can be changed in IOS with the command **ip igmp query-max-response-time seconds** under interface configuration mode and the IOS XR command **query-max-response-time seconds** under IGMP configuration mode or IGMP interface configuration mode.

The receivers receiving this query set an internal random timer between 0 and 10 seconds. When the timer expires, they send membership reports for each group they belong to. If the receiver receives another receiver's report for one of the groups, it belongs to while it has a timer running, it stops its timer for the specified group and does not send a report; this is to suppress duplicate reports.

When a receiver wants to leave a group, it sends a leave group message to the all-routers group address 224.0.0.2 only if it was the last receiver to respond to a query. Otherwise, it can leave quietly because there must be another receiver in the subnet.

When the leave group message is received by the router, it follows with a specific membership query to the group multicast address to determine whether there are any receivers interested in the group remaining in the subnet. If there are none, the router removes the IGMP state for that group.

If there is more than one router in a LAN segment, an IGMP querier election takes place to determine which router will be the querier. IGMPv2 routers send general membership query messages with a source IP of their interface address and destined to the 224.0.0.1 multicast address. When an IGMPv2 router receives this message, it checks the source IP address and compares it to its own interface IP address. The router with the lowest interface IP address in the LAN subnet is elected as the IGMP querier. At this point, all the nonquerier routers start a timer that resets each time they receive a membership query report from the querier router.

If the querier router stops sending membership queries for some reason (for instance, it is powered down), the timer expires and a new querier election takes place. For IOS and IOS XR, the query interval is by default 60 seconds, and it can be changed with the IOS command **ip igmp query-interval seconds** under interface configuration mode and the IOS XR command **query-interval seconds** under IGMP configuration mode or IGMP interface configuration mode. In IOS and IOS XR, the default query interval timer is 2 times the query interval time, which is 120 seconds. The query interval timer can be changed with the IOS command **ip igmp querier-timeout seconds** under interface configuration mode, and with the IOS XR command **query-timeout seconds** under IGMP configuration mode or IGMP interface configuration mode.

### IGMPv3

In IGMPv2, when a receiver sends a membership report to join a multicast group, it does not specify which source it would like to receive multicast traffic from. IGMPv3 is an extension of IGMPv2 that adds support for multicast source filtering, which give the receivers the capability to pick which source from which they will accept multicast traffic.

IGMPv3 is designed to coexist with IGMPv1 and IGMPv2.

IGMPv3 supports all IGMPv2's IGMP message types and is backward compatible with it. The differences between the two are that IGMPv3 added new fields to the IGMP membership query and introduced a new IGMP message type called version 3 membership report to support source filtering.

IGMPv3 supports applications that explicitly signal sources from which they want to receive traffic. With IGMPv3, receivers signal membership to a multicast group address using a membership report in the following two modes:

■ **Include mode:** In this mode, the receiver announces membership to a multicast group address and provides a list of source addresses (the Include list) from which it wants to receive traffic.

■ **Exclude mode:** In this mode, the receiver announces membership to a multicast group address and provides a list of source addresses (the Exclude list) from which it does not want to receive traffic. The receiver will receive traffic only from sources whose IP addresses are *not* listed in the Exclude list. To receive traffic from all sources, which is the behavior of IGMPv2, a receiver uses Exclude mode membership with an empty Exclude list.

IGMPv3 is used primarily for Source Specific Multicast (SSM) and is covered in [Chapter 17, “Advanced IPv4 Multicast Routing.”](#)

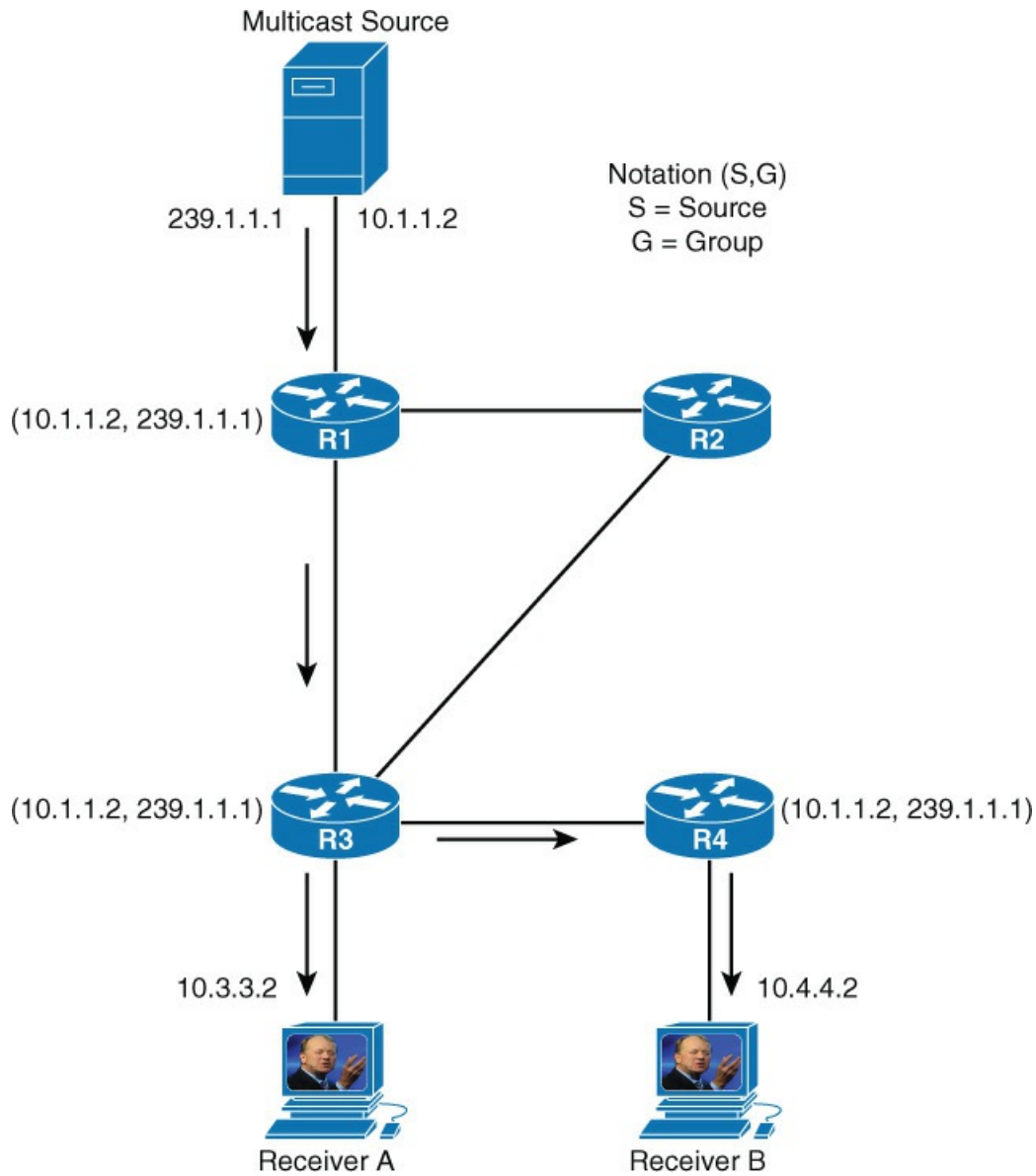
## MULTICAST DISTRIBUTION TREES

Multicast routers create distribution trees that define the path that IP multicast traffic follows through the network to reach the receivers. The two basic types of multicast distribution trees are source trees, also known as *shortest path trees* (SPTs) and shared trees.

### Source Trees

A source tree is a multicast distribution tree where the source is the root of the tree and branches form a distribution tree through the network all the way down to the receivers. When this tree is built, it uses the shortest path through the network from the source to the leaves of the tree; for this reason it is also referred to as a *shortest path tree*.

The forwarding state of the SPT is known by the notation (S,G) pronounced *S comma G*. Where *S* is the source of the multicast stream (server) and *G* is the multicast group address. Using this notation, the SPT state for the example shown in [Figure 16-12](#) is (10.1.1.2, 239.1.1.1), where the multicast source *S* is 10.1.1.2 and the multicast group *G* 239.1.1.1 joined by the Receivers A and B.

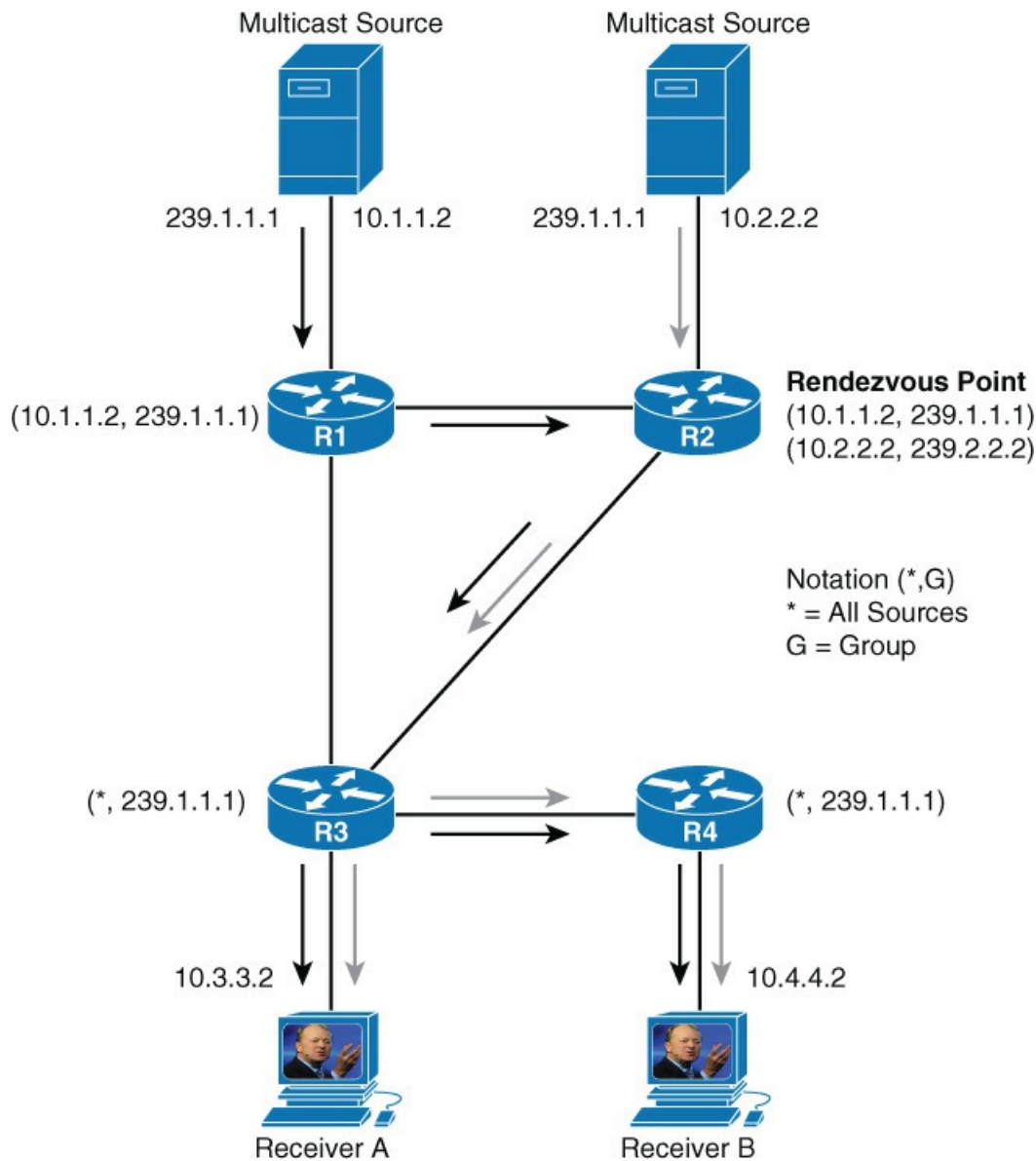


**Figure 16-12** Source Tree Example

Because every SPT is rooted at the source **S**, every source sending to a multicast group will require an SPT.

### Shared Trees

A shared tree is a multicast distribution tree where the root of the shared tree is not the source but a router designated as the rendezvous point (RP). For this reason, shared trees are also referred to as *RP trees* (RPTs). Multicast traffic is forwarded down the shared tree according to the group address **G** that the packets are addressed to, regardless of the source address. For this reason the forwarding state on the shared tree is referred to by the notation **(\*G)** (pronounced *star comma G*). **Figure 16-13** illustrates a shared tree where **R2** is the RP and the **(\*G)** is **(\*239.1.1.1)**.



**Figure 16-13** Shared Tree Between RP and LHRs

One of the benefits of shared trees over source trees is that they require fewer multicast entries (for example, S,G and \*,G). For instance, as more sources are introduced into the network sending traffic to the same multicast group, the number of multicast entries of R3 and R4 will always remain the same (\*,239.1.1.1).

The major drawback of shared trees is that the receivers will receive traffic from all the sources sending traffic to the same multicast group. Even though the receiver's applications can filter out the unwanted traffic, this situation will still generate a lot of unwanted network traffic, wasting bandwidth. In addition, because shared trees can allow multiple sources in an IP multicast group, there is a potential network security issue because unintended sources could send unwanted packets to receivers.

## PROTOCOL INDEPENDENT MULTICAST

Receivers use IGMP to join a multicast group, which is sufficient if the group's source connects to the same router where the receiver is attached. A multicast routing protocol is necessary to route the multicast traffic throughout the network so that routers can locate and request multicast streams from other routers. Multiple multicast routing protocols exist, but Cisco fully supports only Protocol Independent Multicast (PIM).

PIM is a multicast routing protocol that routes multicast traffic between network segments. PIM can use any of the unicast routing protocols to identify the path between the source and receivers and operates.

Figure 16-14 provides a reference topology for some multicast routing terminology, which is further defined in the list that follows.

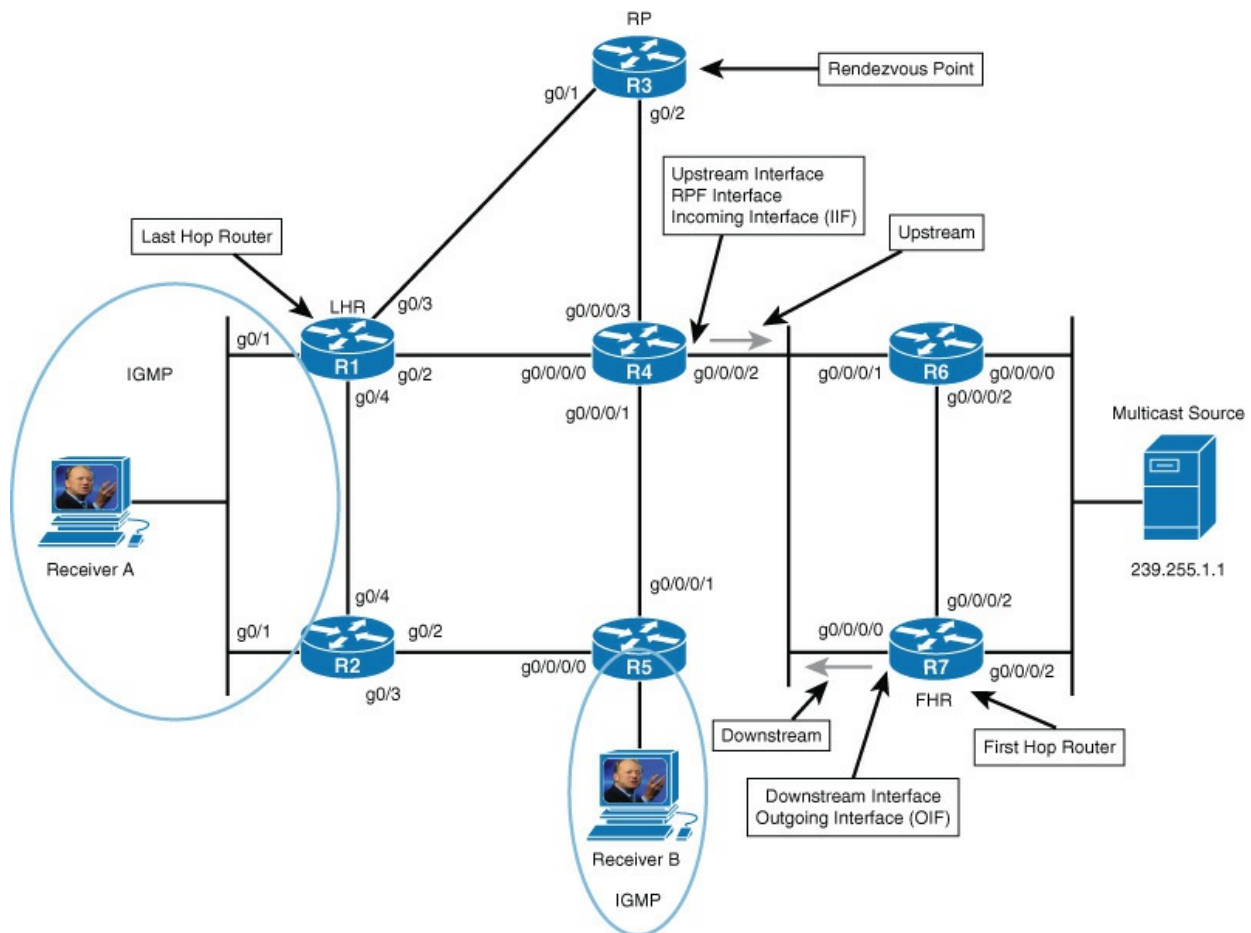


Figure 16-14 PIM Terminology Illustration

■ **Reverse path forwarding (RPF) interface:** The RPF interface is the interface with the lowest cost path (based on administrative distance [AD] and metric) to the IP address of the source or the RP. If multiple interfaces have the same cost, the interface with the highest IP address is chosen as the tiebreaker. An example of this type of interface is `g0/0/0/2` on R4 because the source is known through this interface and it is the only interface on this router that has reachability to the source. Another example is `g0/3` on R1 because the shortest path to the RP is via this interface.

■ **RPF neighbor:** The RPF neighbor is the router that sends a PIM join message using the RPF interface for a multicast group. If the RPF interface is a LAN interface with multiple PIM neighbors, the PIM neighbor with the lowest cost path (based on AD and metric) to the IP address of the source or the RP is elected as the RPF neighbor. If multiple PIM neighbors have the same cost, the PIM neighbor with the highest IP address is elected as the RPF neighbor. For example, R6 and R7 have the same AD and metric to the source, so the highest IP is used as the tiebreaker, and R7 is elected as the RPF neighbor. Another example is R1. If it is using the RPT,

the RPF neighbor is R3, which is the lowest cost to the RP. If it is using an SPT, R4 is its RPF neighbor because it offers the lowest cost to the source.

- **Upstream:** Toward the source of the tree, which could be the actual source in source-based trees or the RP in shared trees. A PIM join travels upstream toward the source.
- **Upstream interface:** Interface toward the source of the tree. It is also known as the *RPF interface* or the *incoming interface* (IIF). An example of an upstream interface is R4's `go/0/0/2` interface, which can send PIM joins upstream to its RPF neighbor.
- **Downstream:** Away from the source of the tree and toward the receivers.
- **Downstream interface:** All interfaces that are used to forward multicast traffic down the tree, also known as the *outgoing interface* (OIF). An example of a downstream interface is R7's `go/0/0/0` interface, which forwards multicast traffic to R4's `go/0/0/2` interface
- **Incoming interface:** It is the same as the RPF interface. It is the only type of interface that can accept multicast traffic coming from the source. An example of this type of interface is `go/0/0/2` on R4 because the source is known through this interface.
- **Outgoing interface (OIF):** It is the same as the downstream interface.
- **Outgoing interface list (OIL):** Group of OIFs that are forwarding multicast traffic to the same group. An example of this is R4's `go/0/0/0` and `go/0/0/1` interfaces sending multicast traffic downstream to Receiver A and Receiver B for the same multicast group.
- **Last-hop router (LHR):** Router directly attached to the receivers. Also known as a *leaf router*. It is responsible for sending PIM joins upstream toward the RP or the source.
- **First-hop router (FHR):** Router directly attached to the source. Also known as the *root router*. It is responsible for sending register messages to the RP.
- **Multicast Routing Information Base (MRIB):** This is a topology table that is also known as the *multicast route* table (mroute), which derives from the unicast routing table and PIM. MRIB contains the source S, group G, incoming interfaces (IIF), outgoing interfaces (OIFs), and RPF neighbor information for each multicast route as well as other multicast related information.
- **Multicast Forwarding Information Base (MFIB):** Uses the MRIB to program multicast forwarding information in hardware for faster forwarding.
- **Multicast state:** This is the multicast traffic forwarding state that is used by the router to forward multicast traffic. The multicast state is composed of the entries found in the mroute table (S, G, IIF, OIF, and so on).

There are currently five PIM operating modes:

- PIM Dense Mode (PIM-DM)



- PIM Sparse Mode (PIM-SM)
- PIM Sparse Dense Mode
- PIM Source Specific Multicast (PIM-SSM)
- PIM Bidirectional Mode (Bidir-PIM)

All PIM control messages use the IP protocol 103; they are either unicast (that is, *register* and *register stop* messages) or multicast with a TTL of 1 to the *all PIM routers* address 224.0.0.13.

Table 16-3 lists the PIM control messages.

Type	Message Type	Destination	PIM Protocol
0	Hello	224.0.0.13 (all PIM routers)	PIM-SM, PIM-DM, Bidir-PIM and SSM
1	Register	RP Address (unicast)	PIM-SM
2	Register stop	First Hop Router (unicast)	PIM SM
3	Join/prune	224.0.0.13 (all PIM routers)	PIM-SM, Bidir-PIM and SSM
4	Bootstrap	224.0.0.13 (all PIM routers)	PIM-SM and Bidir-PIM
5	Assert	224.0.0.13 (all PIM routers)	PIM-SM, PIM-DM and Bidir-PIM
6	Candidate RP advertisement	BSR address (unicast to bootstrap router [BSR])	PIM-SM and Bidir-PIM
7	State refresh	224.0.0.13 (all PIM routers)	PIM-DM
8	DF election	224.0.0.13 (all PIM routers)	Bidir-PIM

Table 16-3 PIM Control Message Types

PIM hello messages are sent by default every 30 seconds out of each PIM-enabled interface to learn about the neighboring PIM routers on each interface to the *all PIM routers* address shown in Table 16-3. Hello messages are also the mechanism used to elect a designated router (DR), which is described later in this chapter, and to negotiate additional capabilities. All PIM routers must record the hello information received from each PIM neighbor.

The interval at which a PIM hello packet is transmitted out of each PIM-enabled interface can be set with the IOS interface parameter command **ip pim query-interval seconds** and the IOS XR command **hello-interval seconds** under PIM interface configuration mode.

Note

PIM Dense Mode, Sparse Dense Mode, Sparse Mode, and Bidir-PIM are covered in this chapter. Chapter 17 covers PIM-SSM in more detail.

### PIM Dense Mode

PIM routers can be configured for PIM Dense Mode (PIM-DM) when it is safe to assume that the receivers of a multicast group are located on every subnet within the network. In other words, the multicast group is densely populated across the network.

For PIM-DM, the multicast tree is built by flooding traffic out of every interface from the source to every dense mode router in the network. The tree is grown from the root toward the leaves. As each router receives traffic for the multicast group, it must decide whether it already has active receivers wanting to receive the multicast traffic. If so, the router remains quiet and lets the multicast flow continue. If no receivers have requested the multicast stream for the multicast group on the LHR, the router sends a prune message toward the source. That branch of the tree is then pruned off so that the unnecessary traffic does not continue. The resulting tree is a source tree because it is unique from the source to the receivers.

Figure 16-15 shows the flood and prune operation of dense mode. The multicast traffic from the source is flooding throughout the entire network. As each router receives the multicast traffic from its upstream neighbor via its RPF interface; it forwards the multicast traffic to all of its PIM-DM neighbors. This results in some traffic arriving via a non-RPF interface, such as the case of R3 receiving traffic from R2 on its non-RPF interface. Packets arriving via the non-RPF interface are discarded.

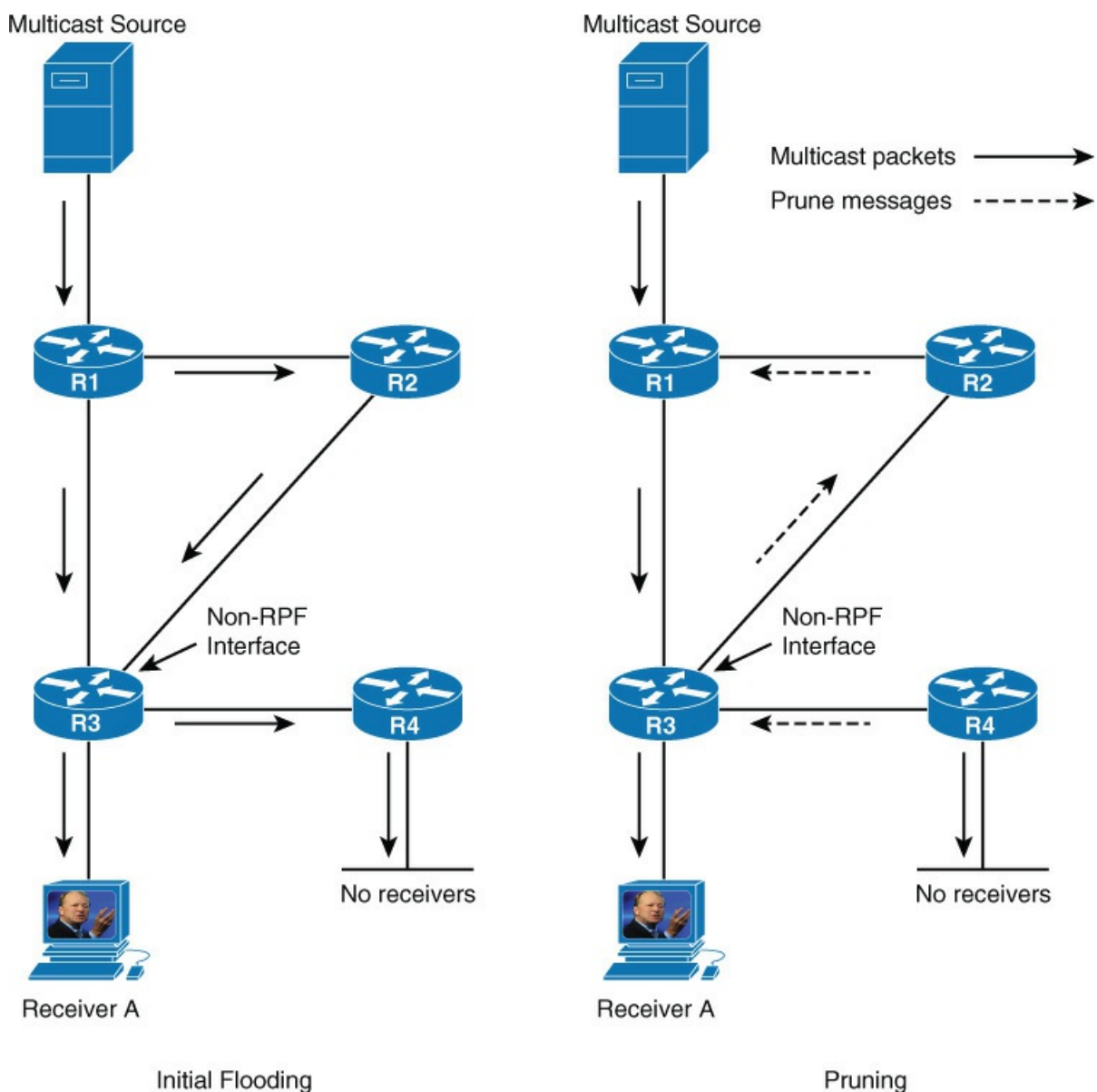


Figure 16-15 PIM-DM Flood and Prune Operation

These non-RPF multicast flows are normal for the initial flooding of multicast traffic and are corrected by the normal PIM-DM pruning mechanism. The pruning mechanism is used to stop the flow of unwanted traffic. Prunes (denoted by the dashed arrows) are sent out of the RPF interface when the router has no downstream members that need the multicast traffic, as is the case for R4, which has no interested receivers, and they are also sent out of non-RPF interfaces to stop the flow of multicast traffic that is arriving via the non-RPF interface, as is the case for R3, where multicast traffic is arriving via a non-RPF interface from R2, which results in a prune message.

Figure 16-16 illustrates the resulting topology after pruning off of all unnecessary links. This results in an SPT from the source to the receiver. Even though the flow of multicast traffic is no longer reaching most of the routers in the network, (S,G) state still remains in all routers in the network. This (S,G) state will remain until the source stops transmitting.

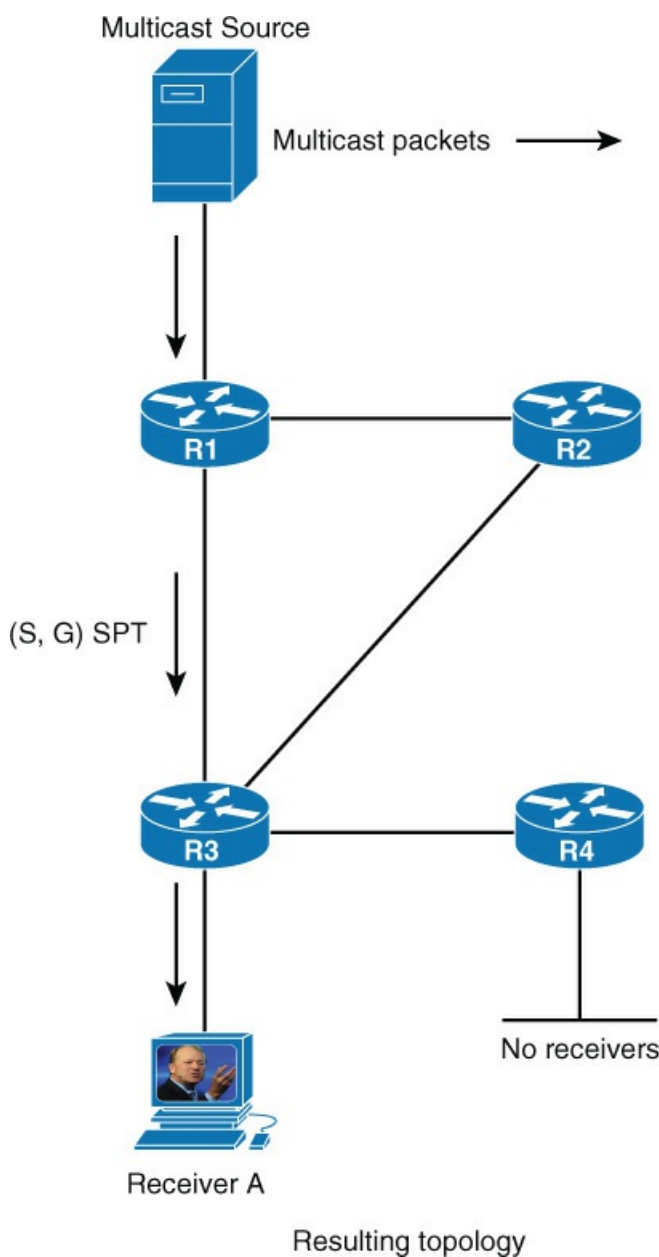


Figure 16-16 PIM-DM Resulting Topology After Pruning

In PIM-DM, prunes expire after 3 minutes. This causes the multicast traffic to be reflooded to all

routers just as was done during the initial flooding. This periodic (every 3 minutes) *flood and prune* behavior is normal and must be taken into account when a network is designed to use PIM-DM.

PIM Dense Mode is applicable to small networks where there are active receivers on every subnet of the network. Because this is rarely the case, it is not generally recommended for production environments; however, it is a good option for a lab environment because it is easy to set up.

IOS XR does not provide support for PIM Dense Mode, and IOS configuration of PIM Dense Mode uses the interface parameter command **ip pim dense-mode** on interfaces participating in multicast routing.

### **PIM Sparse Mode**

PIM Sparse Mode was designed for networks where multicast application receivers are scattered throughout the network. In other words, the multicast group is sparsely populated across the network, but it works well in densely populated networks as well. It also assumes no receivers are interested in multicast traffic unless they explicitly request it.

Just like PIM Dense Mode, it uses the unicast routing table to perform RPF checks, and it does not care which routing protocol (including static routes) populates the unicast routing table; therefore, it is protocol independent.

Note

Chapter 17 covers RPF failures and how to correct them.

### **PIM Shared and Source Path Trees**

PIM Sparse Mode uses an explicit join model where the receivers send an IGMP join to their locally connected router, which is also known as the *last-hop router* (LHR), and this join causes the LHR to send a PIM join all the way up to the root of the tree, which is either the RP in the case of a shared tree or the first-hop router (FHR) where the source transmitting the multicast streams is connected in the case of an SPT.

Multicast forwarding state is created as the result of these explicit joins, which is very different from the flood and prune or implicit join behavior of PIM-DM, where the multicast packet arriving on the router dictates the forwarding state.

Figure 16-17 shows a receiver sending an IGMP membership report, also known as an *IGMP join*, to join a multicast group. The LHR then sends a PIM join to the RP and this forms a shared tree from the RP to the LHR. The RP then sends a PIM join to the FHR forming a source tree between the source and the RP. In essence, there are two trees created, an SPT from the FHR to the RP (S,G) and a shared tree from the RP to the LHR (\*,G).

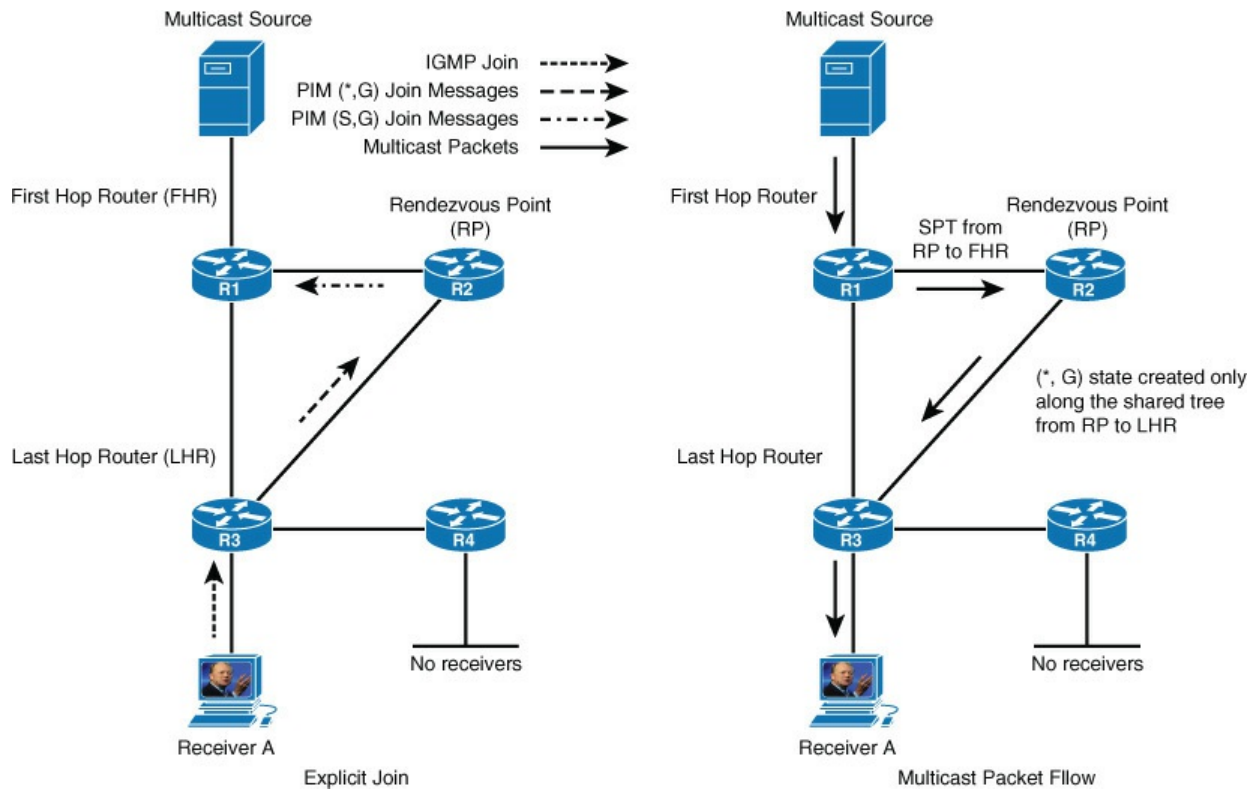


Figure 16-17 PIM-SM Multicast Distribution Tree Building

At this point, multicast starts flowing down from the source to the RP and from the RP to the LHR and then finally to the receiver. This is an oversimplified view of how PIM-SM achieves multicast forwarding. The following sections explain this in more detail.

### Shared Tree Join

Figure 16-17 shows Receiver A attached to the LHR joining multicast group G. The LHR knows the IP address of the RP for group G and it then sends a (\*,G) join for this group to the RP. If the RP were not directly connected, this (\*,G) join would travel hop by hop to the RP, building a branch of the shared tree that would extend from the RP to the LHR. At this point, group G traffic can flow down the shared tree to the receiver.

### Source Registration

In Figure 16-17, as soon as the source for a group G sends a packet, the FHR that is attached to this source is responsible for *registering* this source with the RP and requesting the RP to build a tree back to that router.

The FHR encapsulates the multicast data from the source in a special PIM-SM message called the *register message* and unicasts that data to the RP using a unidirectional PIM tunnel.

When the RP receives the register message it decapsulates the multicast data packet inside of the Register message, and if there is no active shared tree because there are no interested receivers, the RP sends a *register stop* message directly to the registering FHR, without traversing the PIM tunnel, instructing it to stop sending the register messages.

If there is an active shared tree for the group, it forwards the multicast packet down the shared tree, and it sends an (S,G) join back toward the source network S to create a (S,G) SPT. If there are multiple hops (routers) between the RP and the source, this would result in an (S,G) state being created in all the routers along the SPT, including the RP.

As soon as the SPT is built from the source router to the RP, multicast traffic begins to flow

natively from the source S to the RP.

Once the RP begins receiving data natively (that is, down the SPT) from source S, it sends a register stop message to the source's FHR to inform it that it can stop sending the unicast register messages.

At this point, multicast traffic from the source is flowing down the SPT to the RP and from there, down the shared tree to the receiver.

The PIM tunnel will remain in an active *up/up* state even when there are no active multicast streams and will remain active as long as there is a valid RPF path for the RP.

The status of a PIM tunnel is viewed in IOS with the command **show ip pim tunnel** [*tunnel-id*], and the equivalent command in IOS XR is **show pim ipv4 tunnel info** {*tunnel-id* | **all**}.

**Example 16-1** displays the PIM tunnel information for the FHR and the RP. Tunnel0 on R1 is used to send register messages to the RP. The RP router 192.168.2.2 includes one encapsulation tunnel (Tunnel0) just like the FHR router, in addition to a receive tunnel (Tunnel1) for decapsulating incoming register messages from all of the FHR DR routers in the network. These tunnels are automatically created and cannot be seen in the router's configuration.

### Example 16-1 IP PIM Tunnel

[Click here to view code image](#)

```
R2#show ip pim tunnel
Tunnel0
  Type : PIM Encap
  RP   : 192.168.2.2*
  Source: 192.168.2.2
Tunnel1*
  Type : PIM Decap
  RP   : 192.168.2.2*
  Source: -
```

```
R3#show ip interface brief | i Tunnel
Tunnel0          192.168.2.2      YES unset up
Tunnel1          192.168.2.2      YES unset up
```

```
R1#show ip pim tunnel
Tunnel0
  Type : PIM Encap
  RP   : 192.168.2.2
  Source: 10.12.1.1
```

```
R3#show ip interface brief | i Tunnel
Tunnel0          10.12.1.1        YES unset up
```

## PIM SPT Switchover

PIM-SM allows the LHR to switch from the shared tree to an SPT for a specific source. In Cisco routers, this happens immediately after the first multicast packet is received from the RP via the shared tree, but this behavior can be changed with the IOS command `ip pim spt-threshold {kpbs|infinity}[group-list acl-name]` under global configuration mode and with the IOS XR command `spt-threshold infinity [group-list acl-name]` under router PIM configuration mode. The **infinity** keyword will cause all the sources for the specified group to never switch over to an SPT and continue using the shared tree. If a source sends at a rate greater than or equal to the traffic rate (the *kpbs* value), a PIM join message is triggered toward the source to build an SPT. If the traffic rate from the source drops below the threshold traffic rate, the LHR will switch back to the shared tree and send a prune message toward the source. The **group-list** keyword is used to indicate which groups the threshold applies to.

Figure 16-18 illustrates the SPT switchover concept. When the LHR receives the first multicast packet from the RP, it becomes aware of the IP address of the multicast source. At this point, the LHR checks its unicast routing table to see whether the source can be reached through a shorter path than the path through the RP. If it cannot find one, it stays on the shared tree. If it finds one, it sends an (S,G) PIM join to the FHR hop by hop to form an SPT to the source. Once it receives a multicast packet from the FHR through the SPT, if necessary, it switches the RPF interface to be the one in the direction of the FHR, and it then sends a PIM prune message to the RP to shut off the duplicate multicast traffic coming from it.

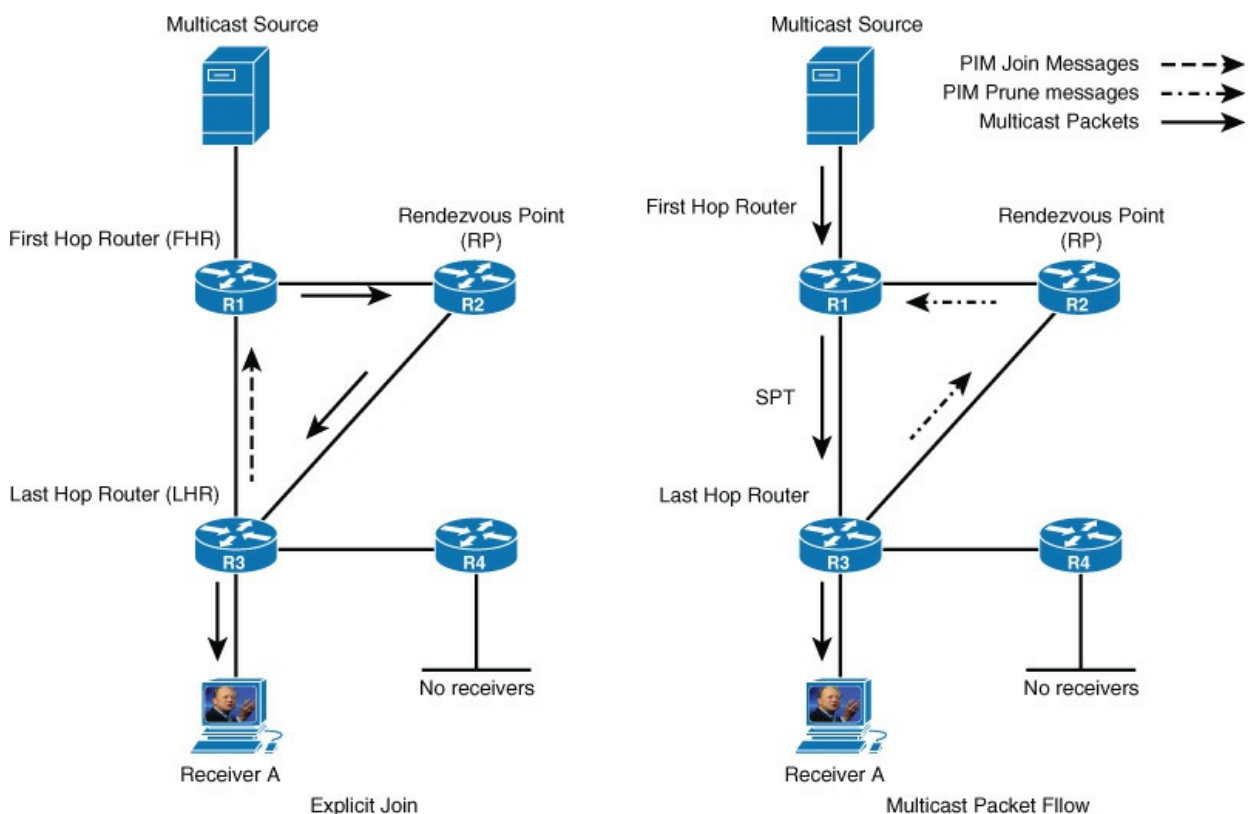


Figure 16-18 PIM-SM SPT Switchover Example

If the RP has no other interfaces that are interested in this multicast traffic, it sends a PIM prune message in the direction of the FHR. If there are any routers between the RP and the FHR, this prune message would travel hop by hop until it reaches the FHR.

## Designated Routers

When multiple PIM-SM routers exist on a LAN segment, PIM hello messages are used to elect a designated router (DR) to avoid sending duplicate multicast traffic into the LAN or the RP. By default, the DR priority value of all PIM routers is 1, and it can be set to force a particular router to become the DR during the DR election process, where a higher DR priority is preferred. If a router in the subnet does not support the DR priority option or all routers have the same DR priority, the highest IP address in the subnet is used as a tiebreaker.

On an FHR, the designated router is responsible for encapsulating in unicast register messages any multicast packets originated by a source that are destined to the RP. On an LHR, the designated router is responsible for sending PIM join and prune messages toward the RP to inform it about host group membership.

Without DRs, all LHRs on the same LAN segment would be capable of sending PIM joins upstream, which could result in duplicate multicast traffic arriving on the LAN. On the source side, if multiple FHRs exist on the LAN, they would all send register messages to the RP at the same time.

The DR priority value can be set in IOS with the interface parameter command **ip pim dr-priority *priority-value*** under interface configuration mode, and in IOS XR with the command **dr-priority *priority-value*** under PIM or PIM interface configuration mode. Configuring the value under the global context is inherited by all interfaces unless a different value is configured on them.

The DR hold time is 3.5 times the hello interval or 105 seconds. If there are no hellos after this interval, a new DR is elected. To reduce DR failover time, the hello query interval can be reduced to speed up failover with a trade-off of more control plane traffic and CPU resource utilization of the router.

## RENDEZVOUS POINTS

When PIM is configured in sparse mode or Bidir-PIM, it is mandatory to choose one or more routers to operate as a rendezvous points (RPs). An RP is a single common root placed at a chosen point of a shared distribution tree, as described earlier in this chapter. An RP can be either configured statically in each router or learned through a dynamic mechanism. A PIM router can be configured to function as an RP either statically in each router in the multicast domain or dynamically by configuring automatic route processing (Auto-RP) or PIM bootstrap router (BSR), which are described in the following sections.

### Note

BSR and Auto-RP were not designed to work together and may introduce unnecessary complexities when deployed in the same network. The recommendation is not to use them both at the same time. Choose one or the other.

### Static RP

An RP can be statically configured for a multicast group range by configuring the address of the RP on every router in the multicast domain. Configuring static RPs is relatively simple and can be achieved with one or two lines of configuration on each router. If the network does not have many different RPs defined or they do not change very often, this could be the simplest method to define RPs. This can also be an attractive option if the network is small.

However, this can increase administrative overhead in a large and complex network. Every router must have the same RP address. This means changing the RP address requires reconfiguring



every router. If several RPs are active for different groups, information about which RP is handling which multicast group must be known by all routers. To ensure this information is complete, multiple configuration commands may be required. If the manually configured RP fails, there is no failover procedure for another router to take over the function performed by the failed RP, and this method by itself does not provide any kind of load splitting.

### **Auto-RP**

Auto-RP is a Cisco proprietary mechanism that automates the distribution of group-to-RP mappings in a PIM network. Auto-RP has the following benefits:

- It is easy to use multiple RPs within a network to serve different group ranges.
- It allows load splitting among different RPs.
- It simplifies RP placement according to the location of group participants.
- It prevents inconsistent manual RP configurations that might cause connectivity problems.
- Multiple RPs can be used to serve different group ranges or to serve as backups for each other.
- The Auto-RP mechanism operates using two basic components, candidate RPs (C-RPs) and RP mapping agents (MAs).

### **Candidate RPs**

C-RPs advertise their willingness to be an RP via RP announcement messages. These messages are sent by default every RP announce interval, which is 60 seconds by default, to the reserved well-known multicast group 224.0.1.39 (Cisco-RP-Announce). The RP announcements contain the default group range 224.0.0.0/4, the C-RP's address and the hold time, which is three times the RP announce interval. If there are multiple C-RPs, the C-RP with the highest IP address is preferred.

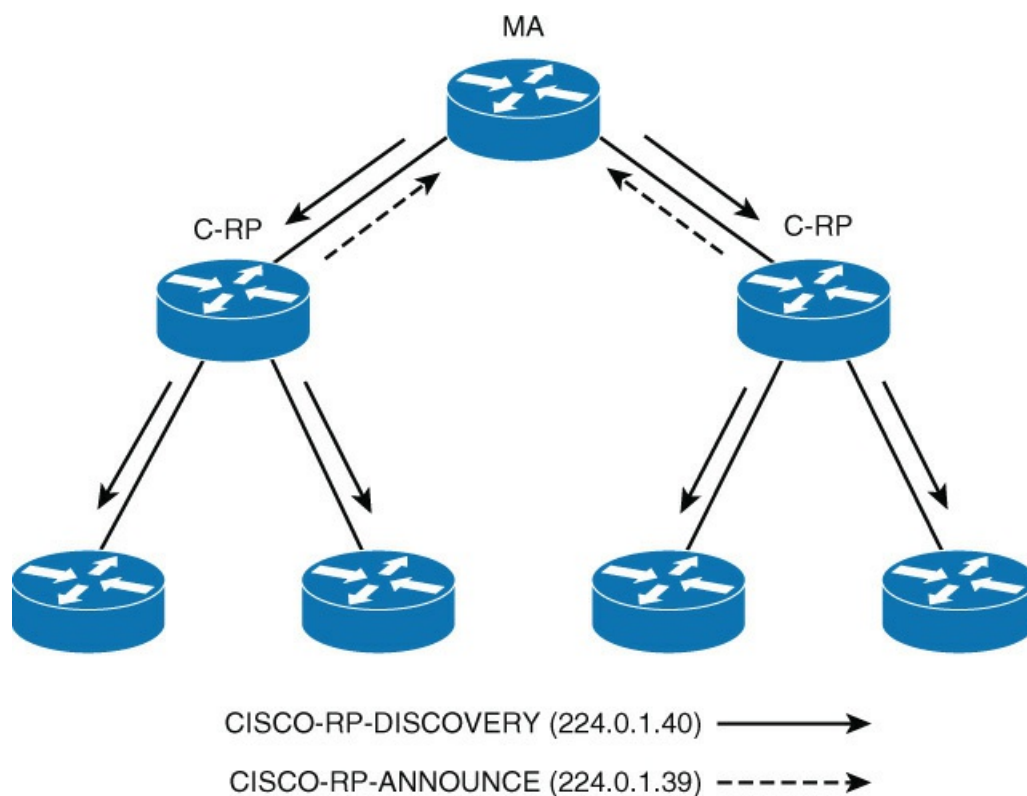
### **RP Mapping Agents**

RP MAs join group 224.0.1.39 to receive the RP announcements. They store the information contained in the announcements in a group-to-RP mapping cache along with hold times. If multiple RPs advertise the same group range, the C-RP with the highest IP address is elected.

The RP MAs advertise the RP mappings to another well-known multicast group address 224.0.1.40 (Cisco-RP-Discovery). These messages are advertised by default every 60 seconds or when there are changes detected. The MA announcements contain the elected RPs and the group-to-RP Mappings. All PIM-enabled routers join 224.0.1.40 and store the RP mappings in their private cache.

Multiple RP MAs can be configured in the same network to provide redundancy in case of failure. There is no election mechanism between them, and they act independently of each other, and they all advertise identical group-to-RP mapping information to all routers in the PIM domain.

Figure 16-19 illustrates the Auto-RP mechanism where the MA periodically receives the C-RP Cisco RP announcements to build a group-to-RP mapping cache and then how it periodically multicasts this information to all PIM routers in the network using Cisco RP discovery messages.



**Figure 16-19** Auto-RP Mechanism

With Auto-RP, all routers automatically learn the RP information, making it easier to administer and update RP information. Auto-RP permits backup RPs to be configured, thus enabling an RP failover mechanism.

### PIM Bootstrap Router

The bootstrap router (BSR) mechanism, described in RFC 5059, is a nonproprietary mechanism that provides a fault-tolerant, automated RP discovery and distribution mechanism.

PIM uses the BSR to discover and announce RP set information for each group prefix to all the routers in a PIM domain. This is the same function accomplished by Auto-RP, but the BSR is part of the PIM Version 2 specification. The RP set is a group-to-RP mapping that contains the following components:

- Multicast group range
- RP priority
- RP address
- Hash mask length
- SM/BiDir flag

Generally, BSR messages originate on the BSR and they are flooded hop by hop by intermediate routers. When a bootstrap message is forwarded, it is forwarded out of every PIM-enabled interface that has PIM neighbors (including the one over which the message was received). BSR messages use the *all PIM routers* address 224.0.0.13 with a TTL of 1.

To avoid a single point of failure, multiple candidate BSRs (C-BSRs) can be deployed in a PIM domain. All C-BSRs participate in the BSR election process by sending a PIM BSR message containing its BSR priority out all interfaces.

The C-BSR with the highest priority is elected as the BSR and sends BSR messages to all PIM routers in the PIM domain. If the BSR priorities are equal or if the BSR priority is not configured, the C-BSR with the highest IP address is elected as the BSR.

### **Candidate RPs**

Routers that are configured as candidate RPs (C-RPs) receive these BSR messages which contain the IP address of the currently active BSR. Because they know the IP address of the BSR, they can unicast candidate RP advertisement (C-RP-Adv) messages directly to it. A C-RP-Adv message carries a list of group address and group mask field pairs. This enables the C-RP to specify the group ranges for which it is willing to be the RP.

The active BSR stores all incoming C RP advertisements in its group-to-RP mapping cache. The BSR then sends the entire list of C-RPs from its group-to-RP mapping cache in BSR messages every 60 seconds by default to all PIM routers in the entire network. As each router receives a copy of these BSR messages, it updates the information in its local group-to-RP mapping cache so that it knows the IP address of all C-RPs in the network.

Unlike Auto-RP, where the mapping agent elects the active RP for a group range and announces the election results to the network, the BSR does not elect the active RP for a group. Instead, it leaves this task to each individual router in the network.

Each router in the network uses a well-known hashing algorithm to elect the currently active RP for a particular group range. Because each router is running the same algorithm against the same list of C-RPs, they will all select the same RP for a particular group range. C-RPs with a lower priority are preferred. If the priorities are the same, the C-RP with the highest IP address is elected as the RP for the particular group range.

Figure 16-20 illustrates the BSR mechanism, where the elected BSR receives candidate RP advertisement messages from all candidate RPs in the domain, and it then sends BSR messages with RP set information out on all enabled interfaces, which are flooded hop by hop to all routers in the network.

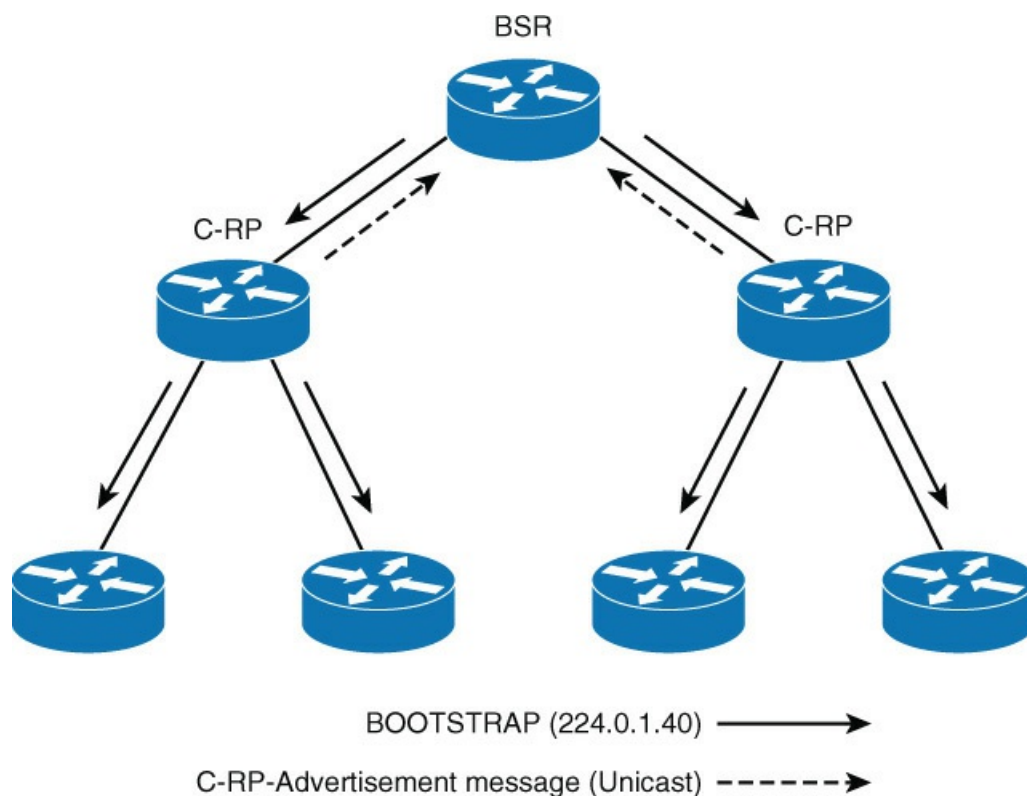


Figure 16-20 BSR Mechanism

## REVERSE PATH FORWARDING

Reverse path forwarding (RPF) is an algorithm used to prevent loops and forwarding multicast packets. It functions as follows:

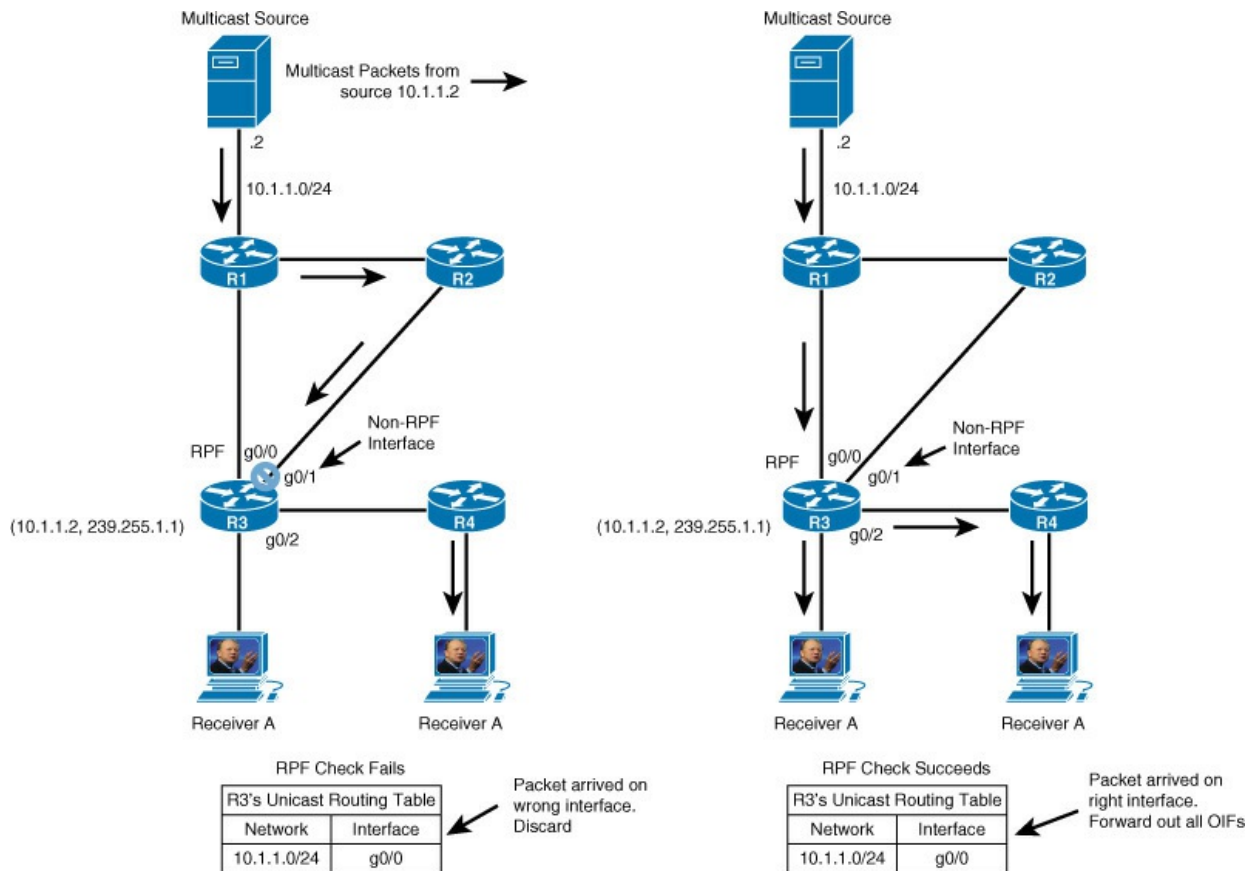
- If a router receives a multicast packet on an interface it uses to send unicast packets to the source, the packet has arrived on the RPF interface.
- If the packet arrives on the RPF interface, a router forwards the packet out the interfaces present in the outgoing interface list of a multicast routing table entry.
- If the packet does not arrive on the RPF interface, the packet is discarded to prevent loops.

PIM uses multicast source trees between the source and the LHR and between the source and the RP. It also uses multicast shared trees between the RP and the LHRs. The RPF check is performed differently for each, as follows:

- If a PIM router has an (S,G) entry present in the multicast routing table (an SPT state), the router performs the RPF check against the IP address of the source for the multicast packet.
- If a PIM router has no explicit source-tree state, this is considered a shared-tree state. The router performs the RPF check on the address of the RP, which is known when members join the group.

PIM-SM uses the RPF lookup function to determine where it needs to send joins and prunes. (S,G) joins (which are SPT states) are sent toward the source. (\*,G) joins (which are shared tree states) are sent toward the RP.

The topology on the left side of [Figure 16-21](#) illustrates a failed RPF check on R3 for the S,G entry because the packet is arriving via a non-RPF interface. The topology on the right shows the multicast traffic arriving on the correct interface on R3; it is then forwarded out all of the OIFs.

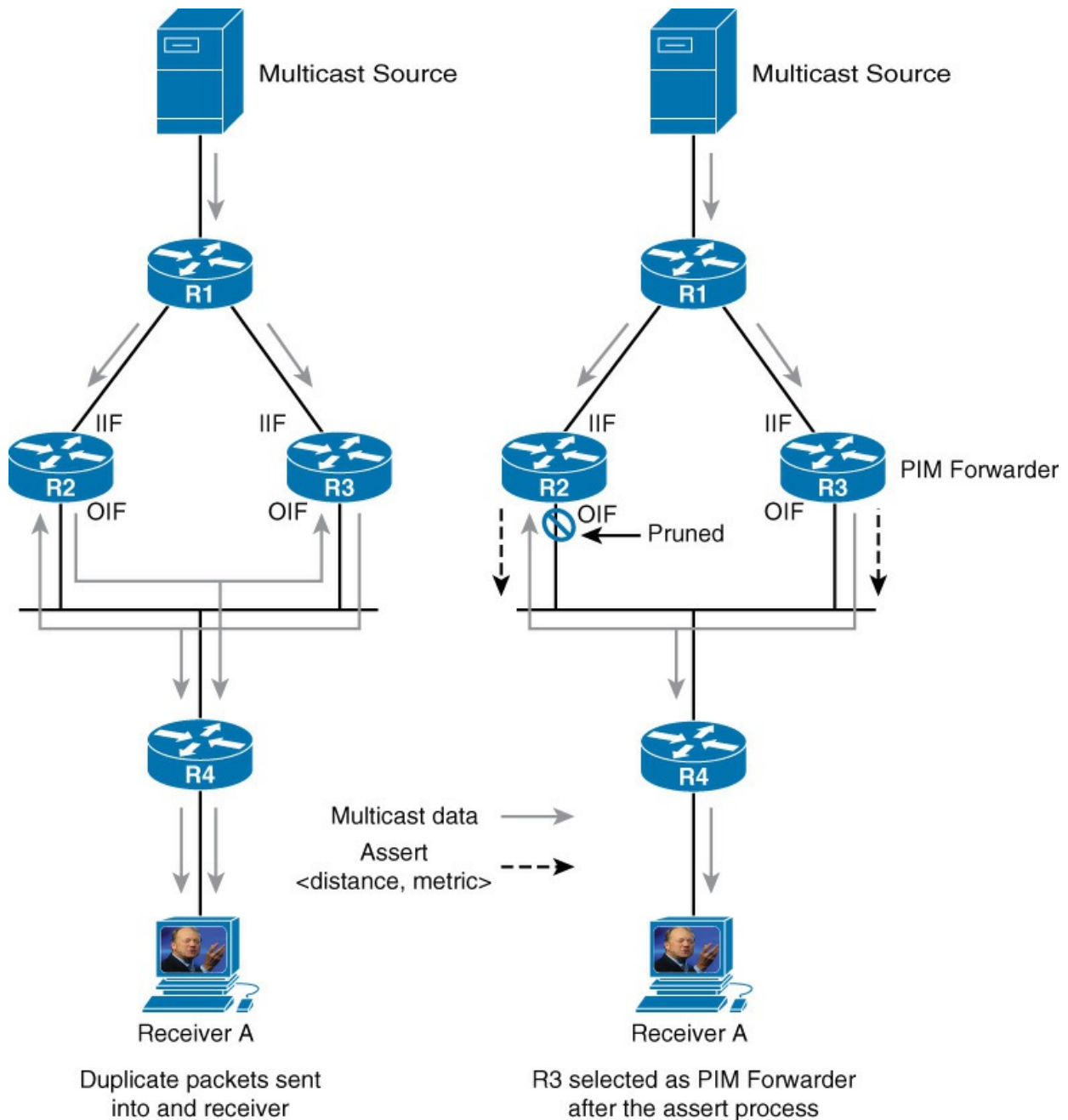


**Figure 16-21** RPF Check

## PIM FORWARDER

There are certain scenarios in which duplicate multicast packets could flow onto a multi-access network. The PIM assert mechanism stops these duplicate flows.

[Figure 16-22](#) illustrates R2 and R3 both receiving the same (S,G) traffic via their RPF interfaces and forward the packet on to the LAN segment. R2 and R3 therefore will receive an (S,G) packet via their downstream OIF interface that is in the OIF of their (S,G) entry. In other words, they detect a multicast packet for a specific (S,G) coming into their OIF that is also going out the same OIF for the same (S,G). This triggers the assert mechanism.



**Figure 16-22** PIM Forwarder Example

R2 and R3 both send PIM assert messages into the LAN. These assert messages contain their AD and route metric back to the source to determine which router should forward the multicast traffic to that network segment.

Each router compares its own values with the received values. Preference is given to the PIM message with the lowest AD. If a tie exists, the lowest route metric for the protocol; and as a final tiebreaker, the highest IP address is used.

The losing router will prune its interface just as if it had received a prune on this interface, and the winning router will be the PIM forwarder for the LAN.

Note

The prune times out after 3 minutes on the losing router and causes it to begin forwarding on this interface again. This triggers the assert process to repeat. If the winning router were to go offline, the loser would take over the job of forwarding on to this LAN segment after its prune timed out.

The PIM forwarder concept applies to PIM-DM and PIM-SM. It is commonly used by PIM-DM but rarely used by PIM-SM because the only time duplicate packets could end up in a LAN is if there is some sort of routing inconsistency.

With the topology shown in [Figure 16-22](#), PIM-SM would not send duplicate flows into the LAN as PIM-DM would, because of the way PIM-SM operates. For example, assuming R1 is the RP, when R4 sends a PIM join message upstream toward it, it sends it to the *all PIM routers* address 224.0.0.13 and R2 and R3 receive it. One of the fields of the PIM join message includes the IP address of the upstream neighbor, also known as the *RPF neighbor*. Assuming R3 is the RPF neighbor, R3 will be the only one that will send a PIM join to R1 and R2 will not because the PIM join was not meant for it. At this point, a shared tree exists between R1, R3, and R4, and no traffic duplication will exist.

[Figure 16-23](#) illustrates how duplicate flows could exist in a LAN using PIM-SM. On the topology on the left side, R2 and R4 are running Open Shortest Path First (OSPF) Protocol, and R3 and R4 are running Enhanced Interior Gateway Routing Protocol (EIGRP). R4 learns about the RP (R1) through R2, and R5 learns about the RP through R3. R4's RPF neighbor is R2, and R5's RPF neighbor is R3. Assuming Receiver A and Receiver B join the same group, R4 would send a PIM join to its upstream neighbor R2, which would in turn send a PIM join to R1. R5 would send a PIM join to its upstream neighbor R3, which would send a PIM join to R1. At this point, traffic starts flowing downstream from R1 into R2 and R3, and duplicate packets are then sent out into the LAN and to the receivers. At this point, the PIM assert mechanism kicks in, R3 is elected as the PIM forwarder, and R2's OIF interface is pruned, as illustrated on the topology on the right side.

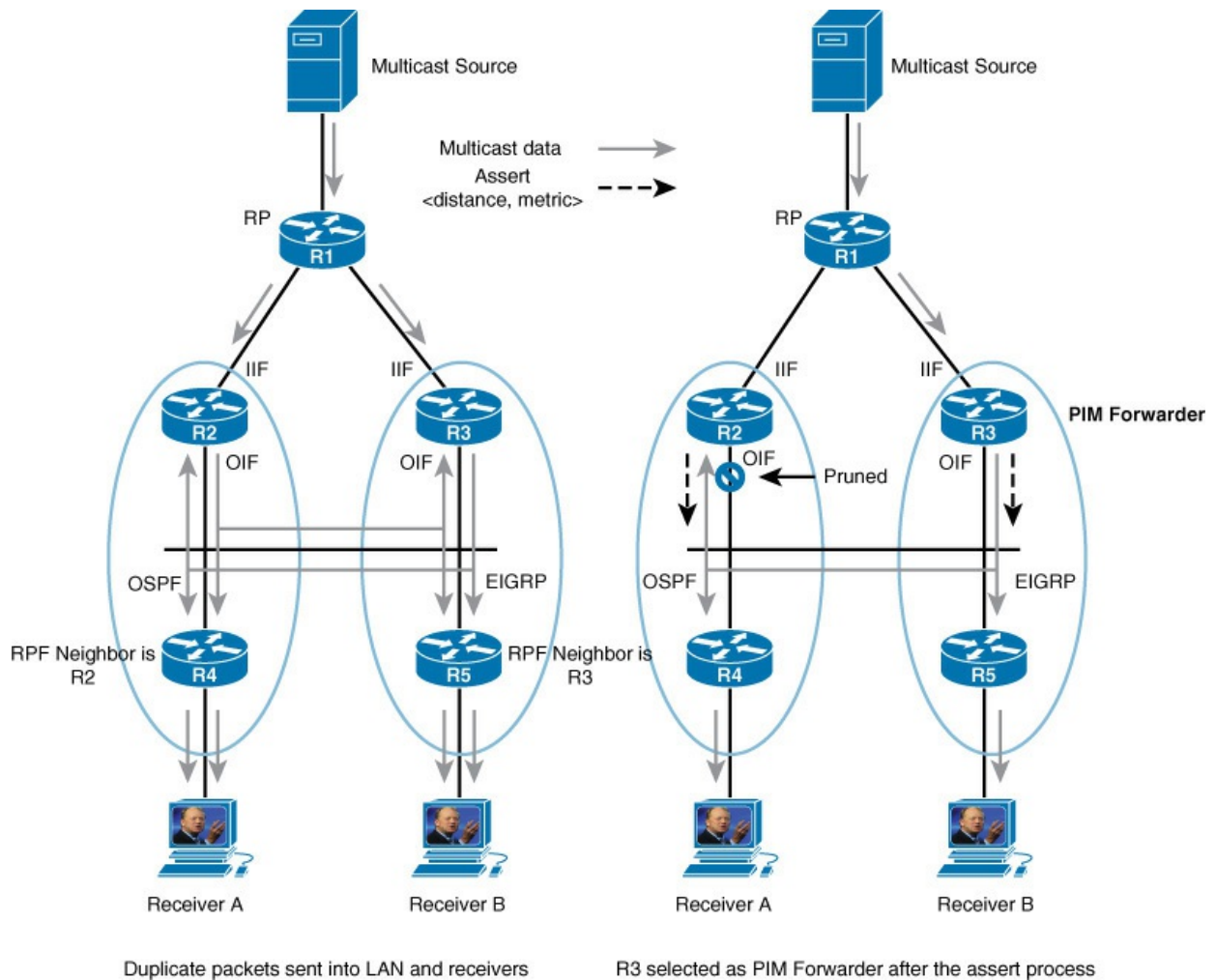


Figure 16-23 PIM-SM PIM Forwarder Example

## BASIC MULTICAST CONFIGURATION

To enable PIM-SM, two general steps are required:

**Step 1.** Enable multicast routing, PIM, and IGMP in all participating routers and selected Layer 3 interfaces.

**Step 2.** Configure one or multiple RPs.

In IOS, the following steps are used to enable IP multicast routing, PIM, and IGMP.

**Step 1.** Enable multicast routing (required). Multicast routing is enabled with the global command **ip multicast-routing**.

**Step 2.** Enable PIM and IGMP (required).

PIM and IGMPv2 are enabled by entering the command **ip pim sparse-mode** on all participating Layer 3 interfaces, including the ones facing the receivers. Enabling PIM-SM on receiver-facing interfaces will enable IGMPv2 by default.

**Step 3.** Configure one or more RPs (required).

There are multiple ways to configure RPs, and these are shown in the [“Configure Rendezvous](#)



[Points](#)” section.

In IOS XR, multicast configuration is performed in protocol-specific configuration modes to provide an easy way for enabling, disabling, and configuring multicast features on a large number of interfaces. The following are the steps used to enable IP multicast routing, PIM, and IGMP:

### Step 1. Enable multicast routing (required).

To enable PIM-SM and IGMP on the router, the following steps are required:

- a. Enter multicast routing configuration mode with the command **multicast-routing [address family ipv4]**.
- b. Enable multicast routing on a specific interface with the command **interface [interface-type interface-number] enable** or the command **interface all enable**.

#### Note

Enabling multicast routing on an interface automatically enables PIM and IGMP on the interface, too. If specific settings are needed for PIM or IGMP, that configuration resides under the router PIM or router IGMP process, as described later.

### Step 2. Specify PIM-specific parameters (optional).

Enter PIM configuration mode with the command **router pim [address family ipv4]**. PIM-specific parameters such as hello intervals, RP configuration, DR priority, and so on can be specified globally or for a specific interface. Global settings are inherited by the interfaces..

### Step 3. Specify IGMP specific parameters (optional).

Enter IGMP configuration mode by entering the command **router igmp**. IGMP-specific parameters such as query interval, query timeout, IGMP version, and so on can be specified globally or for a specific interface. Global settings are inherited by the interfaces.

### Step 4. Configure one or more RPs (required).

There are multiple ways to configure rendezvous points, and these are shown in the [“Configure Rendezvous Points”](#) section.

**Example 16-2** illustrates how to enable multicast routing. Under *IOS XR (Selective Interfaces)*, PIM-SM and IGMP are explicitly enabled on go/o/o/o and loo. Under *IOS XR (All Interfaces)*, PIM-SM is enabled for all interfaces. For IOS, multicast routing needs to be enabled globally and PIM-SM configured in all participating interfaces.

#### **Example 16-2** *Enabling Multicast, PIM, and IGMPv2*

[Click here to view code image](#)

#### **IOS XR (Selective Interfaces)**

```
multicast-routing
address-family ipv4
interface GigabitEthernet0/0/0/0
enable
interface Loopback0
enable
```

#### **IOS XR (All Interfaces)**

```
multicast-routing
address-family ipv4
interface all enable
```

#### **IOS**

```
ip multicast-routing
!
interface GigabitEthernet0/0/0/0
ip pim sparse-mode
```

#### Note

To configure multicast, the multicast software package needs to be loaded on the IOS XR router.

In IOS XR, it is possible to configure multicast commands directly under any multicast configuration mode that would allow those commands to be inherited by all interfaces in the router. To override the inheritance mechanism, interface configuration submode can be entered to explicitly enter a different command parameter.

**Example 16-3** shows a configuration that specifies that all existing and new PIM interfaces on the router will use a hello interval of 20 seconds. However, `go/o/o/o` overrides the global configuration and uses the hello interval time of 5 seconds specified under the interface configuration mode.

### **Example 16-3** *Inheritance Mechanism Override*

[Click here to view code image](#)

#### **IOS XR**

```
router pim
address-family ipv4
hello-interval 20
interface GigabitEthernet0/0/0/0
hello-interval 5
```

### **Configure Rendezvous Points**

As described in previous sections, RPs can be configured either statically or dynamically. In this section, both options are examined.

Figure 16-24 shows a topology example that illustrates the different RP multicast configurations.

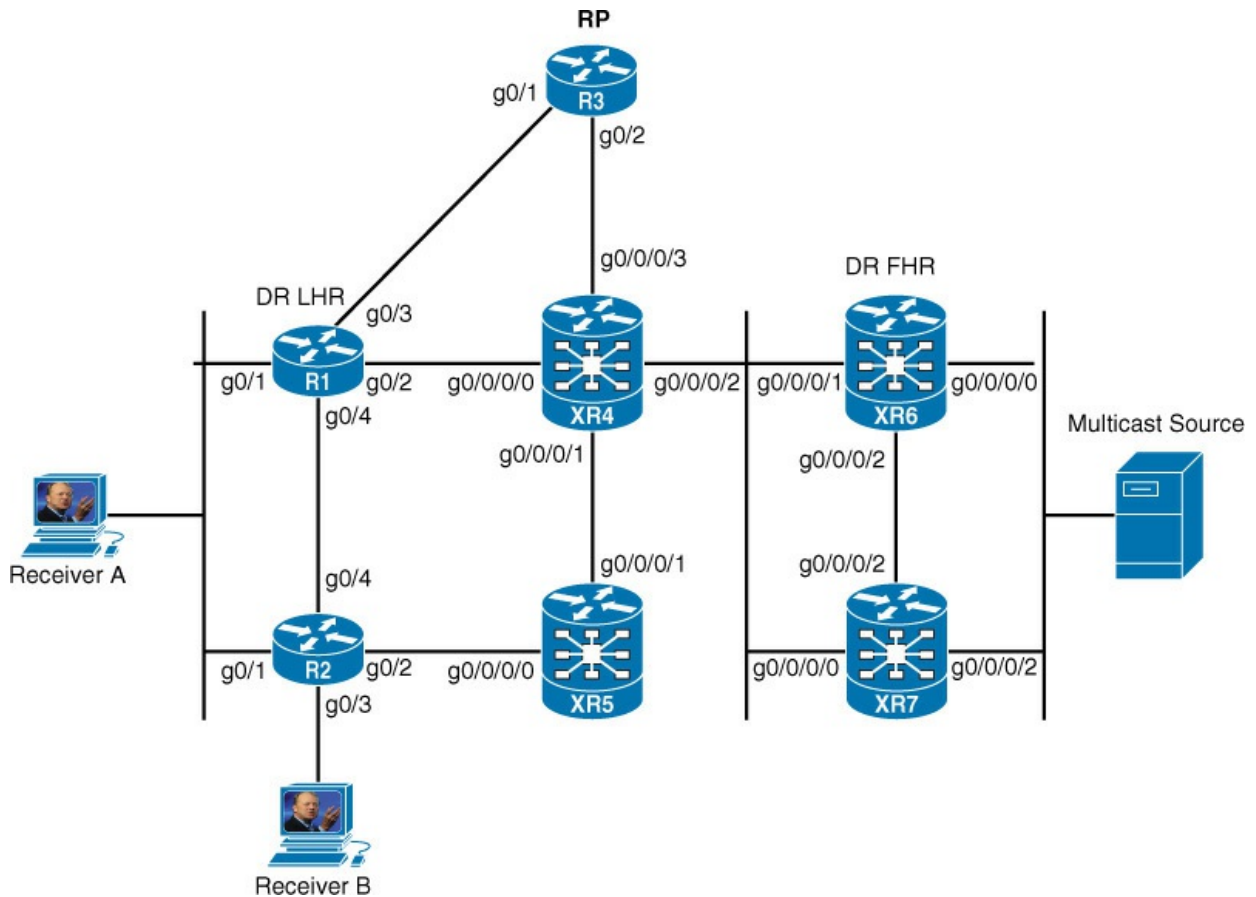


Figure 16-24 Multicast Topology Example

All routers have loopback IP addresses corresponding to their hostname number (R1 = 192.168.1.1/32, R2 = 192.168.2.2/32, and so on). Table 16-4 provides the physical interfaces and associated IP addresses for all hosts in the topology. All hosts shown in the same row in the table are part of the same network segment in the topology.

Host	Interface	IP Address	Host	Interface	IP Address	Host	Interface	IP Address
R1	g0/1	10.12.2.1	R2	g0/1	10.12.2.2	Receiver A	g0/0	10.12.2.3
R1	g0/2	10.14.1.1	XR4	g0/0/0/0	10.14.1.4			
R1	g0/3	10.13.1.1	R3	g0/1	10.13.1.3			
R1	g0/4	10.12.1.1	R2	g0/4	10.12.1.2			
R2	g0/2	10.25.1.2	XR5	g0/0/0/0	10.25.1.5			
R2	g0/3	10.2.2.2	Receiver B	g0/0	10.2.2.1			
R3	g0/2	10.34.1.3	XR4	g0/0/0/3	10.34.1.4			
XR4	g0/0/0/1	10.45.1.4	XR5	g0/0/0/1	10.45.1.5			
XR4	g0/0/0/2	10.46.7.4	XR6	g0/0/0/1	10.46.7.6	XR7	g0/0/0/0	10.46.7.7
XR5	g0/0/0/2	10.5.5.5						
XR6	g0/0/0/0	10.67.1.6	XR7	g0/0/0/1	10.67.1.7	Source	g0/0	10.67.1.10
XR6	g0/0/0/2	10.67.2.6	XR7	g0/0/0/2	10.67.2.7			

**Table 16-4** Interfaces and IP Addresses Used for the Topology Illustrated in [Figure 16-24](#)

Note

Static RPs and C-RPs for BSR and Auto-RP must have PIM-SM enabled on their loopback interfaces.

### Static RP

In IOS, static RPs can be configured with the command **ip pim rp-address** *ip-address* [**group-list** *acl-name*] [**priority** *priority-value*] [**interval** *seconds*] [**scope** *scope-value*] [**bidir**] in global configuration mode, and in IOS XR with the command **rp-address** *ip-address* [*acl-name*] [**override**] [**bidir**] under PIM configuration mode. This configuration is required in all routers including the RP.

Static RP can coexist with Auto-RP or BSR, but Auto-RP or BSR RPs take precedence over statically configured RPs. For example, if a router receives Auto-RP or BSR information for a multicast group that has statically configured RP information, the Auto-RP or BSR information is used.

To override this behavior, and use only the static RP mapping, use the **override** keyword in IOS and IOS XR.

The configuration in [Example 16-4](#) shows how to configure a static RP. For IOS XR, the static RP is configured under PIM configuration mode. In IOS, the static RP is configured under global configuration mode. This needs to be configured in all routers in the multicast domain, including the RP. The interface associated with the RP's IP address needs to be PIM-SM enabled.

### Example 16-4 Static RP Configuration

[Click here to view code image](#)

#### IOS

```
interface Loopback0
  ip address 192.168.3.3
  ip pim sparse-mode
!
ip pim rp-address 192.168.3.3
```

#### IOS XR

```
multicast-routing
  address-family ipv4
    interface all enable
!
router pim
  address-family ipv4
    rp-address 192.168.3.3
```

### Auto-RP

In IOS, C-RPs are configured with the command **ip pim send-rp-announce** [*interface-type interface-number*] **scope** *ttl-value* [**group-list** *acl-name* [**interval** *seconds*] [**bidir**]] under global configuration mode, and in IOS XR with the command **auto-rp candidate-rp** [*interface-type interface-number*] **scope** *ttl-value* [**group-list** *acl-name*] [**interval** *seconds*] [**bidir**] under PIM configuration mode.

In IOS, MAs are configured with the command **ip pim send-rp-discovery** [*interface-type interface-number*] **scope** *ttl-value*, and in IOS XR they are configured with the command **auto-rp mapping-agent** [*interface-type interface-number*] **scope** *ttl-value* [**interval** *seconds*] under PIM configuration mode.

Cisco-RP-Announce (224.0.1.39) and Cisco-RP-Discovery (224.0.1.40) are multicast messages generated by the MAs and the C-RPs respectively that need to be flooded to all PIM-enabled routers in the network. There are three ways to accomplish this:

- Configure one or more static RPs for the groups 224.0.1.39 and 224.0.1.40 in all PIM-enabled routers.
- Enable Auto-RP listener. In IOS XR, the Auto-RP listener is enabled by default, and in IOS it is configured with the command **ip pim autorp listener** under global configuration mode. This feature allows for these messages to be flooded across the entire PIM domain.
- Use the command **ip pim sparse-dense-mode** under interface configuration mode instead of the command **ip pim sparse-mode**. With this command, if a particular multicast group maps to an entry in the router's group-to-RP mapping cache, that group is treated as a sparse mode group on that interface; otherwise, it is treated as a dense mode group on that interface. This will allow the messages to be flooded using dense mode and the groups being advertised by the MA will use sparse mode. This method applies only to IOS and is generally not recommended in favor of the Auto-RP listener feature.

The configuration in [Example 16-5](#) shows how to configure Auto-RP using the Auto-RP listener feature. R3 and XR5 are C-RPs, and XR4 is the MA.

## Example 16-5 Auto-RP Configuration

[Click here to view code image](#)

```
R3 - C-RP
interface Loopback0
 ip pim sparse-mode
 !
 ip pim autorp listener
 ip pim send-rp-announce Loopback0 scope 32
```

```
XR5 - C-RP
router pim
 address-family ipv4
  auto-rp candidate-rp Loopback0 scope 32 interval 60
```

```
XR4 - MA
router pim
 address-family ipv4
  auto-rp mapping-agent Loopback0 scope 32 interval 60
```

### BSR

In IOS, C-BSRs are configured with the command **ip pim bsr-candidate** *[interface-type interface-number] [hash-mask-length] [priority-value]* and in IOS XR with the command **bsr candidate-bsr** *ip-address [hash-mask-len length] [priority value]* under PIM configuration mode.

In IOS, C-RPs are configured with the command **ip pim rp-candidate** *[interface-type interface-number] [bidir] [group-list acl-name] [interval seconds] [priority value]* and in IOS XR with the command **bsr candidate-rp** *ip-address [bidir] [group-list acl-name] [interval seconds] [priority value]* under PIM configuration mode.

#### Note

IOS PIM BSR uses the value 0 as the default priority for C-RPs and C-BSRs, and IOS XR uses the value 192 for C-RPs and 1 for C-BSRs. This implementation predates RFC 5059, where the default priority for C-BSRs is 64 and for C-RPs is 192. The IOS implementation deviates from the RFC and requires the explicit configuration to the appropriate values to be compliant.

The configuration in [Example 16-6](#) shows how to configure BSR. R3 and XR5 are C-RPs, and XR4 is configured as the BSR.

## Example 16-6 BSR Configuration

[Click here to view code image](#)

```
R3 - C-RP
interface Loopback0
 ip pim sparse-mode
 !
 ip pim rp-candidate Loopback0
```

**XR5 - C-RP**

```

router pim
  address-family ipv4
    bsr candidate-rp 192.168.5.5 priority 192 interval 60

```

**XR4 - BSR**

```

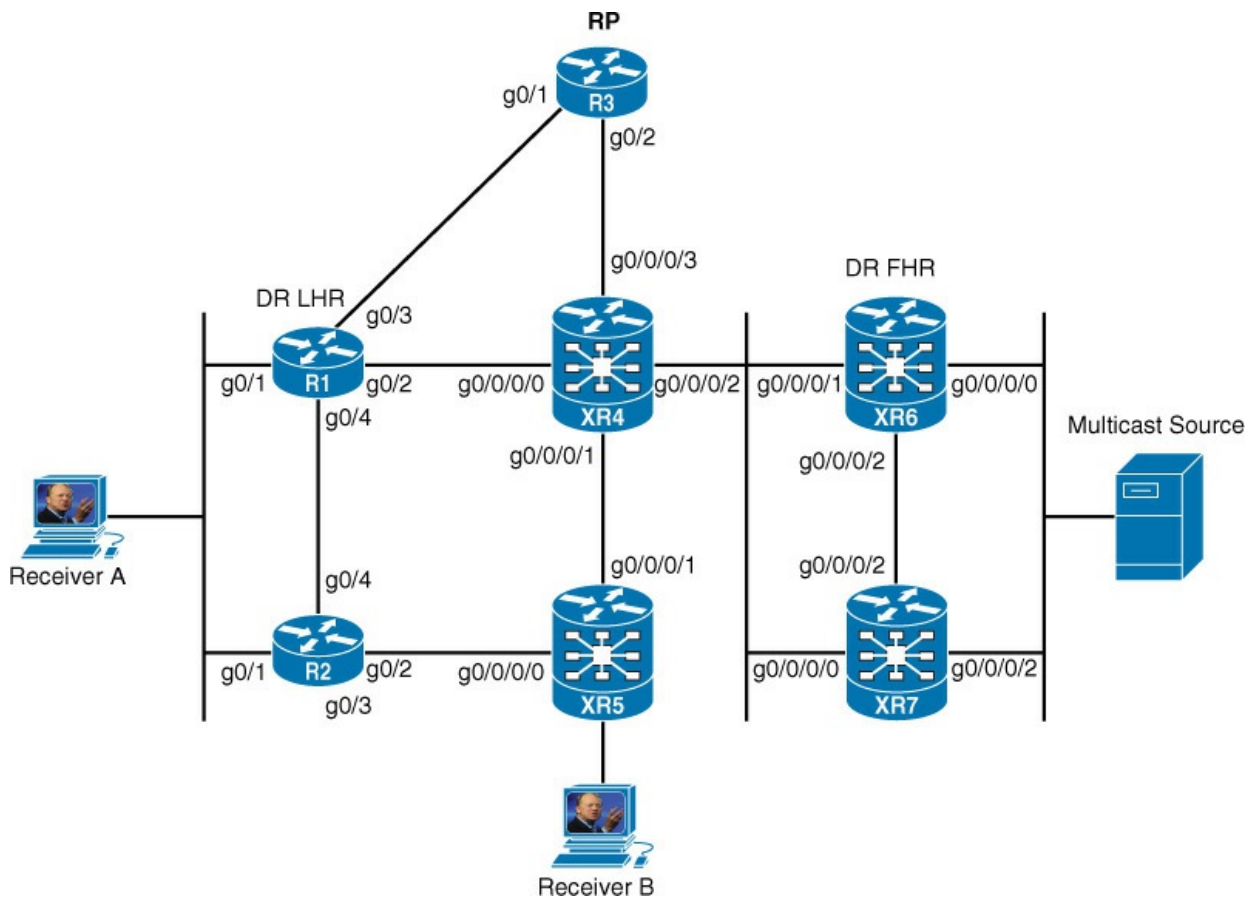
router pim
  address-family ipv4!
    bsr candidate-bsr 192.168.4.4 hash-mask-len 30 priority 1

```

**MULTICAST VERIFICATION**

This section covers the monitoring and verification of multicast in IOS and IOS XR routers by using **show** commands.

**Figure 16-25** illustrates a small PIM-SM multicast network with static RP. A media-streaming server is connected to XR6 and XR7's local LAN, and it is configured to stream multicast traffic using the group address 239.255.1.1. Receiver A is connected to R1's and R2's local LAN and has already signaled via an IGMP join that it would like to receive the multicast stream.



**Figure 16-25** Multicast Domain

**Example 16-7** displays the key multicast configuration components for the example network shown in **Figure 16-25**.

**Example 16-7 Multicast Configurations**

[Click here to view code image](#)

**R1**

```
ip multicast-routing
!
interface GigabitEthernet0/1
 ip pim sparse
!
interface GigabitEthernet0/2
 ip pim sparse
!
interface GigabitEthernet0/3
 ip pim sparse
!
interface GigabitEthernet0/4
 ip pim sparse
!
ip pim rp-address 192.168.3.3
```

**R2**

```
ip multicast-routing
!
interface GigabitEthernet0/1
 ip pim sparse-
!
interface GigabitEthernet0/2
 ip pim sparse
!
interface GigabitEthernet0/3
 ip pim sparse
interface GigabitEthernet0/4
 ip pim sparse
!
ip pim rp-address 192.168.3.3
```

**R3 - RP**

```
ip multicast-routing
!
interface Loopback0
 ip pim sparse-mode
!
interface GigabitEthernet0/1
 ip pim sparse-mode
!
interface GigabitEthernet0/2
 ip pim sparse-mode
!
ip pim rp-address 192.168.3.3
```



**XR4**

```
multicast-routing
  address-family ipv4
    interface all enable
  !
router pim
  address-family ipv4
    rp-address 192.168.3.3
```

**XR5**

```
multicast-routing
  address-family ipv4
    interface all enable
  !
router pim
  address-family ipv4
    rp-address 192.168.3.3
```

**XR6**

```
multicast-routing
  address-family ipv4
    dr-priority 255
    rate-per-route
    interface all enable
  !
  !
router pim
  address-family ipv4
    rp-address 192.168.3.3
    interface GigabitEthernet0/0/0/0
      dr-priority 255
```

**XR7**

```
multicast-routing
  address-family ipv4
    interface all enable
  !
router pim
  address-family ipv4
    rp-address 192.168.3.3
```

Information such as which router is the IGMP querier, IGMP version, and so on can be verified with the IOS command **show ip igmp interface** [*interface-type interface-number*] and the IOS XR **show igmp interface** [*interface-type interface-number*], as shown in [Example 16-8](#).

**Example 16-8 IGMP Interface Settings**

[Click here to view code image](#)

```
R2#show ip igmp interface g0/1
GigabitEthernet0/1 is up, line protocol is up
Internet address is 10.12.2.2/24
IGMP is enabled on interface
Current IGMP host version is 2
Current IGMP router version is 2
IGMP query interval is 60 seconds
IGMP configured query interval is 60 seconds
IGMP querier timeout is 120 seconds
IGMP configured querier timeout is 120 seconds
IGMP max query response time is 10 seconds
Last member query count is 2
Last member query response interval is 1000 ms
Inbound IGMP access group is not set
IGMP activity: 2 joins, 0 leaves
Multicast routing is enabled on interface
Multicast TTL threshold is 0
Multicast designated router (DR) is 10.12.2.2 (this system)
IGMP querying router is 10.12.2.1
Multicast groups joined by this system (number of users):
    224.0.1.40(1)
```

```
RP/0/0/CPU0:XR5#show igmp interface g0/0/0/2

GigabitEthernet0/0/0/2 is up, line protocol is up
Internet address is 10.5.5.5/24
IGMP is enabled on interface
Current IGMP version is 3
IGMP query interval is 60 seconds
IGMP querier timeout is 125 seconds
IGMP max query response time is 10 seconds
Last member query response interval is 1 seconds
IGMP activity: 6 joins, 0 leaves
IGMP querying router is 10.5.5.5 (this system)
Time elapsed since last query sent 00:00:02
Time elapsed since IGMP router enabled 05:37:14
Time elapsed since last report received 00:00:49
```

R1 and R2 both receive an IGMP join from Receiver A for group 239.255.1.1. The IGMP joins can be verified with the IOS command **show ip igmp groups** [*interface-type interface-number*], as shown in [Example 16-9](#). The equivalent command in IOS XR is **show igmp groups** [*interface-type interface-number*].

### **Example 16-9** IGMP Entry for Receiver A

[Click here to view code image](#)

```
R1#show ip igmp groups gigabitEthernet 0/1
IGMP Connected Group Membership
Group Address      Interface          Uptime    Expires    Last Reporter    Group
Accounted
239.255.1.1       GigabitEthernet0/1 00:48:53  00:02:58  10.12.2.3
224.0.1.40        GigabitEthernet0/1 00:52:05  00:02:59  10.12.2.2
```

```
R2#show ip igmp groups gigabitEthernet 0/1
IGMP Connected Group Membership
Group Address      Interface          Uptime    Expires    Last Reporter    Group
Accounted
239.255.1.1       GigabitEthernet0/1 00:49:27  00:02:24  10.12.2.3
224.0.1.40        GigabitEthernet0/1 00:52:14  00:02:26  10.12.2.2
```

To see which interfaces are participating in PIM and what their DR priority is and so on, you can use the IOS **show ip pim interface** command. The equivalent command in XR is **show pim interface**, as shown in [Example 16-10](#).

### Example 16-10 PIM Interfaces

[Click here to view code image](#)

```
R1#show ip pim interface

Address          Interface          Ver/  Nbr   Query  DR    DR
                  Mode             Count Intvl Prior
10.12.2.1        GigabitEthernet0/1 v2/S  1     30     1     10.12.2.2
10.14.1.1        GigabitEthernet0/2 v2/S  1     30     1     10.14.1.4
10.13.1.1        GigabitEthernet0/3 v2/S  0     30     1     10.13.1.1
10.12.1.1        GigabitEthernet0/4 v2/S  1     30     1     10.12.1.2
```

```
RP/0/0/CPU0:XR6#show pim interface

PIM interfaces in VRF default
Address          Interface          PIM  Nbr   Hello  DR    DR
                  Count Intvl  Prior
192.168.6.6     Loopback0         on   1     30     255  this
system
10.67.1.6       GigabitEthernet0/0/0/0 on   2     30     255  this
system
10.46.7.6       GigabitEthernet0/0/0/1 on   3     30     255  this
system
10.67.2.6       GigabitEthernet0/0/0/2 on   2     30     255  this
system
```

To avoid duplicate traffic into the LAN, only the DR is allowed to send PIM joins upstream toward the RP (before the SPT switchover) or the source (after the SPT switchover). R2 is elected as the DR for the LAN because it has the highest IP address. This can be verified by using the IOS command **show ip pim neighbor [interface-type interface-number]**, as shown in [Example 16-](#)

11.

### Example 16-11 R2 Elected as the DR

[Click here to view code image](#)

```
R1#show ip pim neighbor gigabitEthernet 0/1
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
      P - Proxy Capable, S - State Refresh Capable, G - GenID Capable
Neighbor      Interface      Uptime/Expires    Ver  DR
Address
10.12.2.2     GigabitEthernet0/1  00:56:21/00:01:29 v2   1 / DR S P G
```

```
R2#show ip pim neighbor gigabitEthernet 0/1
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
      P - Proxy Capable, S - State Refresh Capable, G - GenID Capable
Neighbor      Interface      Uptime/Expires    Ver  DR
Address
10.12.2.1     GigabitEthernet0/1  00:56:45/00:01:34 v2   1 / S P G
```

The IOS command to check which router is the active RP for the multicast group is **show ip pim rp group-address**, and the IOS XR command is **show pim group-map group-address** and **show pim rp [mapping]**.

Example 16-12 displays the active PIM group mapping for the multicast address 239.255.1.1. The assigned RP address is 192.168.3.3 (R3), operating in PIM-SM, and the RP is configured using static RP.

Notice that XR4 is not showing any multicast state for the 239.255.1.1 group address, but is instead showing the default 224.0.0.0/4 multicast range. The reason for this is that the multicast source is not sending any multicast traffic yet.

### Example 16-12 PIM Group Mapping

[Click here to view code image](#)

```
R2#show ip pim rp 239.255.1.1
Group: 239.255.1.1, RP: 192.168.3.3, uptime 01:07:58, expires never
```

```

RP/0/0/CPU0:XR4#show pim group-map 239.255.1.1
IP PIM Group Mapping Table
(* indicates group mappings being used)
(+ indicates BSR group mappings active in MRIB)

Group Range          Proto Client  Groups RP address      Info
-----
224.0.0.0/4*        SM    config  0          192.168.3.3      RPF: Gi0/0/0/3,10.34.1.3

RP/0/0/CPU0:XR4#show pim rp mapping
tPIM Group-to-RP Mappings
Group(s) 224.0.0.0/4
  RP 192.168.3.3 (?), v2
    Info source: 0.0.0.0 (?), elected via config
    Uptime: 02:22:59, expires: never

```

In IOS, to view the multicast distribution tree, use the commands **show ip mroute src-ip-address/group-address** and **show ip mrrib route src-ip-address/group-address**. In IOS XR, the equivalent commands are **show pim topology src-ip-address/group-address** and **show mrrib route src-ip-address/group-address**.

Example 16-13 displays the PIM topology table for 239.1.1.1 on R2 and R3. R2 shows Gigabit Ethernet 0/1 as the OIF, and it is in forwarding state, which indicates that it is ready to forward multicast traffic. The (\*,239.255.1.1) entry indicates that this is a shared tree, and the SJC flags set at the same time indicate that the group is ready to do a PIM switchover. The IIF is Gigabit Ethernet 0/4, and the RPF neighbor is R1. The IIF on R3 is Gigabit Ethernet 0/1 and the IIF is Null because the source has not yet started transmitting its multicast stream. From the outputs of these commands, it can be concluded that the shared distribution tree was formed between R1, R2, and R3.

### **Example 16-13** PIM Topology Table

[Click here to view code image](#)

```

R2#show ip mroute 239.255.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
      L - Local, P - Pruned, R - RP-bit set, F - Register flag,
      T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
      X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
      U - URD, I - Received Source Specific Host Report,
      Z - Multicast Tunnel, z - MDT-data group sender,
      Y - Joined MDT-data group, y - Sending to MDT-data group,
      G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
      N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
      Q - Received BGP S-A Route, q - Sent BGP S-A Route,
      V - RD & Vector, v - Vector, p - PIM Joins on route
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.255.1.1), 01:44:14/00:02:42, RP 192.168.3.3, flags: SJC
Incoming interface: GigabitEthernet0/4, RPF nbr 10.12.1.1
Outgoing interface list:
  GigabitEthernet0/1, Forward/Sparse, 01:37:29/00:02:42

```

```

R3#show ip mroute 239.255.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
      L - Local, P - Pruned, R - RP-bit set, F - Register flag,
      T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
      X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
      U - URD, I - Received Source Specific Host Report,
      Z - Multicast Tunnel, z - MDT-data group sender,
      Y - Joined MDT-data group, y - Sending to MDT-data group,
      G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
      N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
      Q - Received BGP S-A Route, q - Sent BGP S-A Route,
      V - RD & Vector, v - Vector, p - PIM Joins on route
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.255.1.1), 01:45:54/00:02:54, RP 192.168.3.3, flags: S
Incoming interface: Null, RPF nbr 0.0.0.0
Outgoing interface list:
  GigabitEthernet0/1, Forward/Sparse, 01:45:54/00:02:54

```

**Table 16-5** describes the flags that appear in the output in **Example 16-13** and some other useful flags.

Flag	Description
D = Dense	Entry is operating in dense mode.
S = Sparse	Entry is operating in sparse mode.
J – Join SPT (*,G)	When the J flag is set for (*,G) entries, the next (S,G) multicast packet received will trigger a PIM SPT switchover. For the switchover to take place, the C flag needs to be set as well.
C = Connected	A member of the multicast group is present on the directly connected interface.
L = Local	The router itself is a member of the multicast group.
P = Pruned	Route has been pruned. IOS keeps this information in case a downstream member wants to join the source.
R = RP-bit set	Indicates that the (S,G) entry is pointing toward the RP.
F = Register flag	Indicates that register messages are being sent to the RP for a multicast source.
T = SPT-bit set	Indicates that packets have been received on the shortest path source tree.

Table 16-5 IOS Most Useful show ip mroute Flags

On the source side, the DR is responsible for sending encapsulated multicast data packets into PIM register messages destined to the RP. The DR priority was set to 255 on R6, as shown in [Example 16-7](#), so that it would be elected as the DR. This can be verified by using the IOS command **show pim neighbor** [interface-type interface-number], as shown in [Example 16-14](#).

### Example 16-14 XR6 Elected as DR

[Click here to view code image](#)

```
RP/0/0/CPU0:XR6#show pim neighbor gigabitEthernet 0/0/0/0

PIM neighbors in VRF default
Flag: B - Bidir capable, P - Proxy capable, DR - Designated Router,
      E - ECMP Redirect capable
      * indicates the neighbor created for this router

Neighbor Address      Interface                Uptime    Expires    DR  pri  Flags
10.67.1.6*           GigabitEthernet0/0/0/0  02:09:05  00:01:17  255 (DR) B P
10.67.1.7             GigabitEthernet0/0/0/0  02:08:45  00:01:44  1   B P
```

Once the source starts transmitting, multicast state is created for the group in XR6, as shown in [Example 16-15](#). In the output of the **show mrrib route** and **show mfib route** commands, Encapstunnel0 indicates that the multicast traffic is being encapsulated in register messages and is being unicasted to the RP. This is why XR4 is not showing any state yet. The A flag on XR6 indicates that it can accept traffic from the source on Gigabit Ethernet 0/0/0/0, and the F flag indicates that multicast traffic can be forwarded out Encapstunnel0.

The source and destination of Encapstunnel0 can be seen with the command **show pim ipv4 tunnel info all**. The source IP address 10.46.7.6 is the IP of R6's RPF interface, and the

destination IP address 192.168.3.3 is the IP of the RP.

In the output of the **show pim topology** command, the RA flag indicates the source is alive and transmitting traffic. The KAT flag tracks whether traffic is flowing for the (S,G) route on which it is set. A route does not time out while the KAT is running. The KAT runs for 3.5 minutes, and the route goes into KAT probing mode for as long as 65 seconds. The route is deleted if no traffic is seen during the probing interval. Because this multicast route is for register messages, as indicated by the SR flag, it will time out eventually when the RP switches over to the SPT. This will happen the moment R3 receives the register messages because it will check the source IP address and build an SPT between R3 and XR7, as indicated by the T flag on R3. From R3, traffic will continue to flow down the shared tree to R2 and finally to Receiver A.

#### Note

The reason why the SPT was not built between R3 and XR6 is explained in an example later in this section.

### Example 16-15 Source Registration and SPT Between RP and FHR Tree Creation

[Click here to view code image](#)

```
RP/0/0/CPU0:XR6#show mrib route 239.255.1.1
IP Multicast Routing Information Base
Entry flags: L - Domain-Local Source, E - External Source to the Domain,
             C - Directly-Connected Check, S - Signal, IA - Inherit Accept,
             IF - Inherit From, D - Drop, ME - MDT Encap, EID - Encap ID,
             MD - MDT Decap, MT - MDT Threshold Crossed, MH - MDT interface handle
             CD - Conditional Decap, MPLS - MPLS Decap, MF - MPLS Encap, EX - Extranet
             MoFE - MoFRR Enabled, MoFS - MoFRR State, MoFP - MoFRR Primary
             MoFB - MoFRR Backup, RPFID - RPF ID Set
Interface flags: F - Forward, A - Accept, IC - Internal Copy,
                NS - Negate Signal, DP - Don't Preserve, SP - Signal Present,
                II - Internal Interest, ID - Internal Disinterest, LI - Local Interest,
                LD - Local Disinterest, DI - Decapsulation Interface
                EI - Encapsulation Interface, MI - MDT Interface, LVIF - MPLS Encap,
                EX - Extranet, A2 - Secondary Accept, MT - MDT Threshold Crossed,
                MA - Data MDT Assigned, LMI - mLDP MDT Interface, TMI - P2MP-TE MDT Interface
                IRMI - IR MDT Interface
```

```
(10.67.1.10,239.255.1.1) RPF nbr: 10.67.1.10 Flags:
```

```
Up: 00:00:16
```

```
Incoming Interface List
```

```
GigabitEthernet0/0/0/0 Flags: A, Up: 00:00:16
```

```
Outgoing Interface List
```

```
Encapstunnel0 Flags: F NS EI, Up: 00:00:16
```

```
RP/0/0/CPU0:XR6#show mfib route 239.255.1.1
```

```
IP Multicast Forwarding Information Base
```

```
Entry flags: C - Directly-Connected Check, S - Signal, D - Drop,
```

```
IA - Inherit Accept, IF - Inherit From, EID - Encap ID,
```

```
ME - MDT Encap, MD - MDT Decap, MT - MDT Threshold Crossed,
```

```
MH - MDT interface handle, CD - Conditional Decap,
```

```
DT - MDT Decap True, EX - Extranet, RPFID - RPF ID Set,
```

```
MoFE - MoFRR Enabled, MoFS - MoFRR State
```



```

Interface flags: F - Forward, A - Accept, IC - Internal Copy,
  NS - Negate Signal, DP - Don't Preserve, SP - Signal Present,
  EG - Egress, EI - Encapsulation Interface, MI - MDT Interface,
  EX - Extranet, A2 - Secondary Accept
Forwarding/Replication Counts: Packets in/Packets out/Bytes out
Failure Counts: RPF / TTL / Empty Olist / Encap RL / Other

```

```
(10.67.1.10,239.255.1.1),  Flags:
```

```

Up: 00:00:19
Last Used: never
SW Forwarding Counts: 0/0/0
SW Replication Counts: 0/0/0
SW Failure Counts: 0/0/0/0/0
Encapstunnel0 Flags:  F NS EI, Up:00:00:19
GigabitEthernet0/0/0/0 Flags:  A, Up:00:00:19

```

```
RP/0/0/CPU0:XR6#show pim topology 239.255.1.1
```

```
IP PIM Multicast Topology Table
```

```
Entry state: (*S,G) [RPT/SPT] Protocol Uptime Info
```

```
Entry flags: KAT - Keep Alive Timer, AA - Assume Alive, PA - Probe Alive
```

```
  RA - Really Alive, IA - Inherit Alive, LH - Last Hop
```

```
  DSS - Don't Signal Sources,  RR - Register Received
```

```
  SR - Sending Registers, SNR - Sending Null Registers
```

```
  E - MSDP External, EX - Extranet
```

```
  MFA - Mofrr Active, MFP - Mofrr Primary, MFB - Mofrr Backup
```

```
  DCC - Don't Check Connected, ME - MDT Encap, MD - MDT Decap
```

```
  MT - Crossed Data MDT threshold, MA - Data MDT Assigned
```

```
  SAJ - BGP Source Active Joined, SAR - BGP Source Active Received,
```

```
  SAS - BGP Source Active Sent, IM - Inband mLDP
```

```
Interface state: Name, Uptime, Fwd, Info
```

```
Interface flags: LI - Local Interest, LD - Local Dissinterest,
```

```
  II - Internal Interest, ID - Internal Dissinterest,
```

```
  LH - Last Hop, AS - Assert, AB - Admin Boundary, EX - Extranet,
```

```
  BGP - BGP C-Multicast Join, BP - BGP Source Active Prune,
```

```
  MVS - MVPN Safi Learned, MV6S - MVPN IPv6 Safi Learned
```

```
(10.67.1.10,239.255.1.1)SPT SM Up: 00:00:20
```

```
JP: Join(never) RPF: GigabitEthernet0/0/0/0,10.67.1.10* Flags: KAT(00:03:10) RA SR
```

```
Encapstunnel0          00:00:20 fwd
```

```
RP/0/0/CPU0:XR6#show pim ipv4 tunnel info all
```

Interface	RP Address	Source Address
Encapstunnel0	192.168.3.3	10.46.7.6

```
RP/0/0/CPU0:XR4#show mrrib route 239.255.1.1
```

```
No matching routes in MRIB route-DB
```

```

R3#show ip mroute 239.255.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
      L - Local, P - Pruned, R - RP-bit set, F - Register flag,
      T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
      X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
      U - URD, I - Received Source Specific Host Report,
      Z - Multicast Tunnel, z - MDT-data group sender,
      Y - Joined MDT-data group, y - Sending to MDT-data group,
      G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
      N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
      Q - Received BGP S-A Route, q - Sent BGP S-A Route,
      V - RD & Vector, v - Vector, p - PIM Joins on route
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.255.1.1), 00:02:07/00:03:12, RP 192.168.3.3, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    GigabitEthernet0/1, Forward/Sparse, 00:00:17/00:03:12

(10.67.1.10, 239.255.1.1), 00:02:07/00:02:22, flags: T
  Incoming interface: GigabitEthernet0/2, RPF nbr 10.34.1.4
  Outgoing interface list:
    GigabitEthernet0/1, Forward/Sparse, 00:00:17/00:03:12

```

Table 16-6 and Table 16-7 explain the flags that appear in the output in Example 16-15 and some other useful flags

Flag	Description
S = Signal	Indicates the MFIB will signal PIM when traffic is received on any interface for this entry that does not have the NS flag set.
A = Accept	Indicates that multicast data can be accepted on this interface. For example, for PIM-SM, the A flag would appear on the RPF interface set in the mroute table.
F = Forward	Indicates that multicast data can be forwarded out this interface. For example, the interfaces that are in the outgoing interface list in the mroute table will have this flag set.
NS = Negate signal	Indicates that the MFIB will notify the multicast control plane when traffic is received on the specified interface, if the S flag is not set. The NS flag is used for <ul style="list-style-type: none"> <li>■ <b>SPT switchover in PIM-SM:</b> The NS flag is set on the (*,G) accept interface toward the RP to trigger SPT switchover.</li> <li>■ <b>Asserts:</b> The NS-flag is set on (*,G) and (S,G) forward interfaces to trigger PIM asserts.</li> <li>■ <b>Liveness checking for active sources in PIM-SM:</b> The NS flag is set on the (S,G) accept interface toward the source to check for active sources.</li> </ul>

Table 16-6 IOS XR Most Useful show mrrib route and show mfib route Flags

<b>Flag</b>	<b>Description</b>
KAT = Keepalive timer	The keepalive timer tracks whether traffic is flowing for the (S,G) route on which it is set. A route does not time out while the KAT is running. The KAT runs for 3.5 minutes, and the route goes into KAT probing mode for as long as 65 seconds. The route is deleted if no traffic is seen during the probing interval and if there is no longer any reason to keep the route; for example, registers and (S,G) joins.
AA = Assume alive	Indicates that the source was alive, but recent confirmation of traffic flow was not received.
RA = Really alive	Indicates that the source is confirmed to be sending traffic for the route.
RR = Register received	Indicates that the RP has received and answered PIM register messages for this (S,G) route.
SR = Sending registers	Indicates that the FHR DR has begun sending registers for this (S,G) route, but has not yet received a register stop message.
LI = Local interest	Indicates that there are local receivers for this entry on this interface, as reported by IGMP.
AS = Assert	Indicates that a PIM assert message was seen on this interface and the active PIM assert state exists.

**Table 16-7** *IOS XR Most Useful show pim topology Flags*

The moment R2 receives multicast traffic from the RP (R3), it checks the source IP address and performs an SPT switchover, as indicated by the JT flags shown in [Example 16-16](#).

### **Example 16-16** *SPT Switchover*

[Click here to view code image](#)

```

R2#show ip mroute 239.255.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
      L - Local, P - Pruned, R - RP-bit set, F - Register flag,
      T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
      X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
      U - URD, I - Received Source Specific Host Report,
      Z - Multicast Tunnel, z - MDT-data group sender,
      Y - Joined MDT-data group, y - Sending to MDT-data group,
      G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
      N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
      Q - Received BGP S-A Route, q - Sent BGP S-A Route,
      V - RD & Vector, v - Vector, p - PIM Joins on route
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.255.1.1), 00:19:39/stopped, RP 192.168.3.3, flags: SJC
  Incoming interface: GigabitEthernet0/4, RPF nbr 10.12.1.1
  Outgoing interface list:
    GigabitEthernet0/1, Forward/Sparse, 00:07:02/00:02:29

(10.67.1.10, 239.255.1.1), 00:00:10/00:02:49, flags: JT
  Incoming interface: GigabitEthernet0/4, RPF nbr 10.12.1.1
  Outgoing interface list:
    GigabitEthernet0/1, Forward/Sparse, 00:00:10/00:02:49

```

In [Example 16-17](#), XR4's IIF is Gigabit Ethernet 0/0/0/2, and the OIF is Gigabit Ethernet 0/0/0/0. From this, it can be concluded that the SPT was formed between R1, R2, XR4, and XR7. The reason it is XR7 and not XR6 is because they have the same AD and route metric back to the source, so the highest IP is used as the tiebreaker and XR7 is elected as the RPF neighbor. You can verify the RPF neighbor with the IOS command **show ip rpf ip-address** and the IOS XR command **show pim rpf ip-address**.

### Example 16-17 SPT Path

[Click here to view code image](#)

```

RP/0/0/CPU0:XR4#show pim rpf 10.67.1.10
Table: IPv4-Unicast-default
* 10.67.1.10/32 [110/20]
  via GigabitEthernet0/0/0/2 with rpf neighbor 10.46.7.7

```

```

R1#show ip rpf 10.67.1.10
RPF information for ? (10.67.1.10)
  RPF interface: GigabitEthernet0/2
  RPF neighbor: ? (10.14.1.4)
  RPF route/mask: 10.67.1.0/24
  RPF type: unicast (ospf 1)
Doing distance-preferred lookups across tables
RPF topology: ipv4 multicast base, originated from ipv4 unicast base

```

The IOS command to view active multicast streams is **show ip mroute active**. The most similar command in IOS XR is **show mfib ipv4 route rate group-address**.

IOS XR routers require the command **rate-per-route** under the address family in multicast routing configuration mode in order to gather statistics for the command **show mfib ipv4 route rate**.

[Example 16-18](#) demonstrates how to enable multicast stream statistics.

### Example 16-18 Enabling (S,G) Rate Calculations

[Click here to view code image](#)

```
multicast-routing
address-family ipv4
  rate-per-route
```

[Example 16-19](#) displays the active multicast flows for R1 and XR4. The media server multicast server is streaming to group 239.255.1.1 and is consuming ~3290 Kbps of bandwidth.

### Example 16-19 Active Traffic Flows

[Click here to view code image](#)

```
R1#show ip mroute active
Use "show ip mfib active" to get better response time for a large number of mroutes.

Active IP Multicast Sources - sending >= 4 kbps

Group: 239.255.1.1, (?)
Source: 10.67.1.10 (?)
Rate: 4115 pps/3292 kbps(1sec), 3292 kbps(last 40 secs), 3141 kbps(life avg)
```

```
RP/0/RSP0/CPU0:XR4#show mfib route rate
IP Multicast Forwarding Rates
(Source Address, Group Address)
Incoming rate: pps (Incoming node)
Outgoing rate:
Node: (Outgoing node) : pps

(10.67.1.10,239.255.1.1)
Incoming rate : 4110 pps (0/0/CPU0)
Outgoing rate : 4110 pps (0/0/CPU0)
```

## BIDIRECTIONAL PIM

Bidir-PIM was designed for many-to-many applications within individual PIM domains. These types of applications result in an (S,G) entry for the many sources; therefore, the number of (S,G) entries in the mroute table in the routers can increase dramatically. As the number of entries increase, the routers need to work harder and harder trying to maintain this state.

The SPT switchover can be disabled on LHRs for PIM-SM to always use the shared tree to the

RP, which can result in some state reduction, but it is generally insufficient. This is because, as discussed in previous sections, in PIM-SM (\*,G) and (S,G) state is always created along the path from the FHR to the RP.

Bidir-PIM eliminates the need for FHR PIM register messages to be sent to the RP and an SPT between the FHR and the RP by allowing multicast packets to natively flow from the source to the RP using shared-tree state only. This ensures that only (\*,G) entries will appear in the mroute tables and that the data path for the multicast packets flowing from the source to the RP and vice versa will be the same.

Bidirectional shared trees violate the normal (\*,G) RPF rules because traffic is allowed to flow upstream along the shared tree. This means that in certain situations, traffic will be accepted on an outgoing interface. This can lead to multicast routing loops, which can melt down a network.

To overcome this problem, Bidir-PIM uses designated forwarders (DFs). The DF is the router on each subnet that is responsible for forwarding traffic up the shared tree and hence is allowed to violate the normal RPF rules and accept traffic on an outgoing interface.

In Bidir-PIM, multicast traffic is routed only along a bidirectional shared tree that is rooted at the RP for the group. The RPs function is to act as a routing vector in which all the traffic converges, and unlike PIM-SM, the RP address (RPA) only needs to be a routable address and need not exist on a physical device, making the RP a phantom RP. This means that the RP IP address does not need to be configured on a router but can just be an IP address in a link. This link is known as the *rendezvous point link* (RPL). In Bidir-PIM, all multicast traffic to groups mapping to a specific RPA is forwarded to the RPL of that RPA. The RPL is special within a Bidir-PIM domain because it is the only link on which a DF election does not take place.

The RP can also be a physical router, but it is not necessary. Using a phantom RP, also known as *RPA*, is the preferred technique for establishing a redundant RP configuration for Bidir-PIM.

Just as PIM-SM, membership to a bidirectional group is signaled via explicit join messages. Traffic from multicast sources is unconditionally sent up the shared tree toward the RP and passed down the tree toward the receivers on each branch of the tree.

Figure 16-26 and Figure 16-27 illustrate the difference in multicast state created per router for a PIM-SM shared tree and source tree versus a bidirectional shared tree. In Figure 16-26, the topology on the left illustrates what the PIM topology looks like before an SPT switchover or what it looks like if the SPT switchover is disabled. Notice that (\*,G) and (S,G) state is required between the RP and the FHR R1 because there is no way in PIM-SM to use shared trees between the RP and the FHR. With just two sources, the state of the RP for PIM-SM is higher as compared to the state of the RP shown in Figure 16-27 for Bidir-PIM, even though Bidir-PIM has more sources and receivers. The reason for this is that Bidir-PIM only uses (\*,G) shared trees from the source to the RP and down to the receivers, which allows it to scale beyond PIM-SM's capabilities.

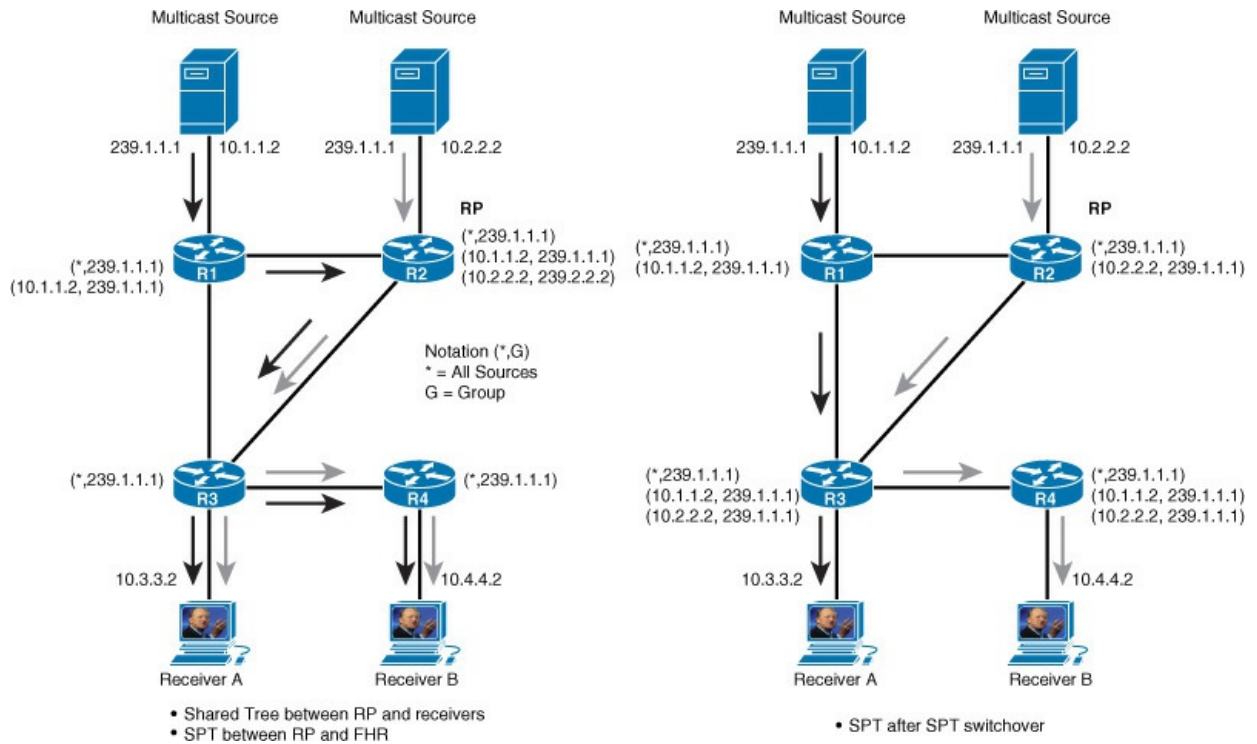


Figure 16-26 PIM-SM Shared Tree and SPT

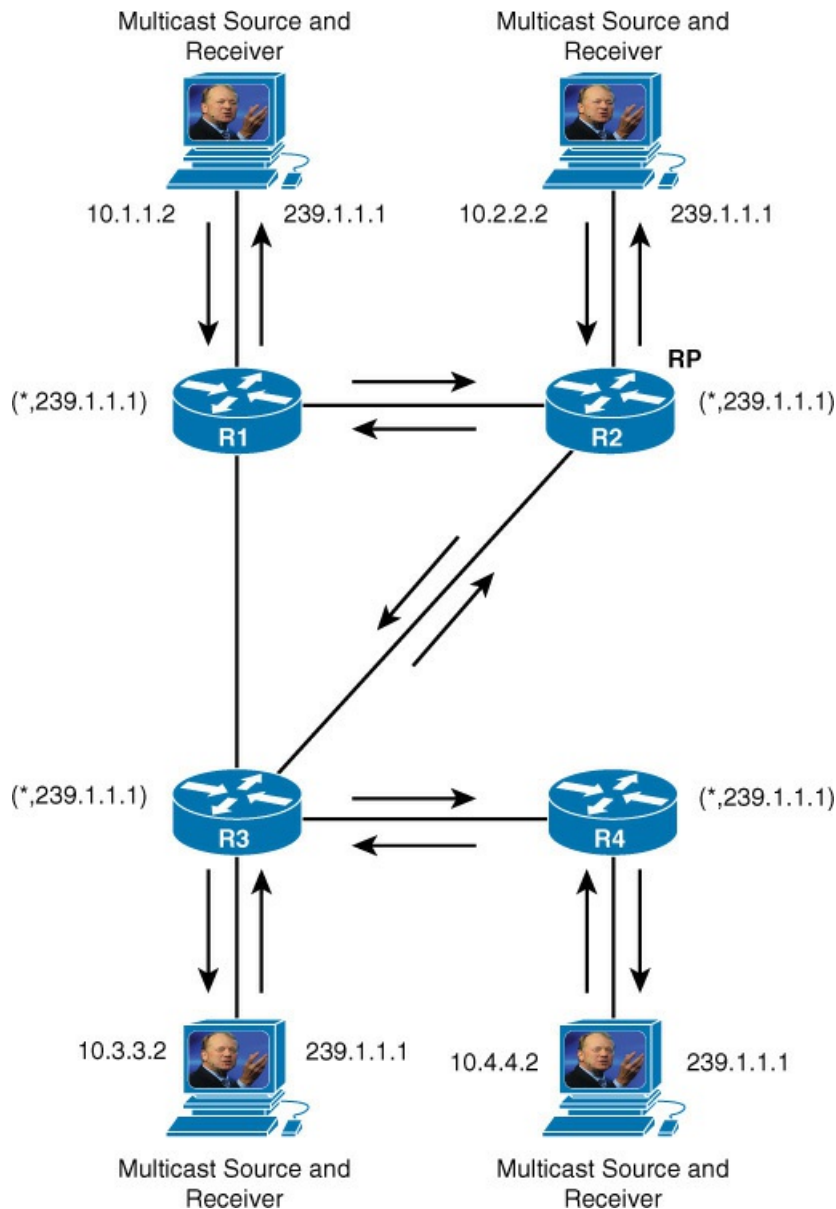


Figure 16-27 Bidir-PIM Shared Tree

### Bidir-PIM Designated Forwarder

On every network segment, including point-to-point links, PIM routers participate in a designated forwarder (DF) election. The DF election process selects one router as the DF for every RP of bidirectional groups. The DF is responsible for forwarding multicast packets received on that network upstream to the RP and for forwarding multicast packets downstream to the receivers.

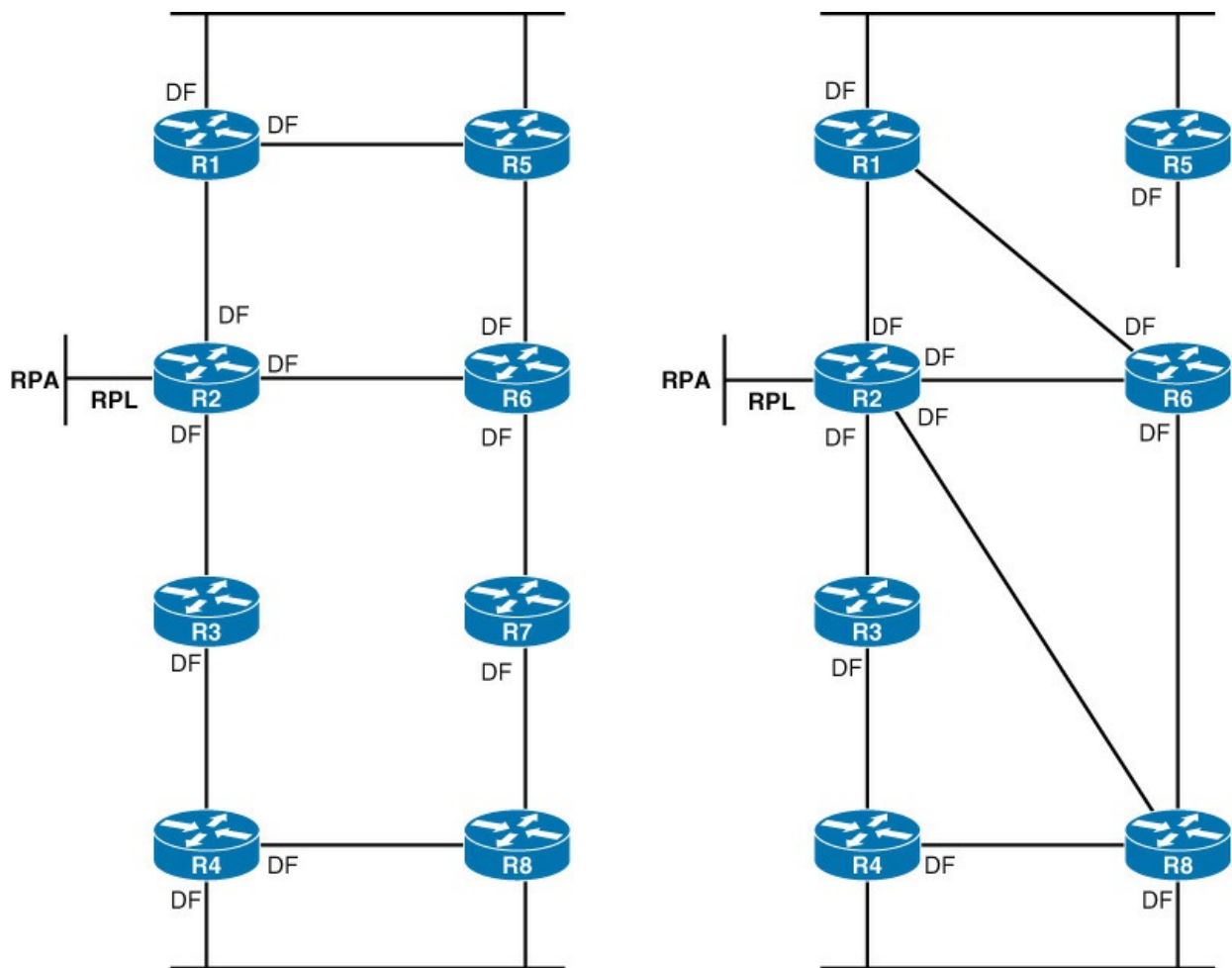
The DF election is based on unicast routing metrics and uses the same tiebreak rules employed by PIM assert processes. The router with the most preferred unicast routing metric to the RP becomes the DF. Use of this method ensures that only one copy of every packet will be sent to the RP, even if there are parallel equal-cost paths to the RP.

Because a DF is selected for every RP of bidirectional groups, multiple PIM routers may be elected as DFs on any network segment, one for each RP. In addition, any particular router may be elected as DF on more than one interface.

The DF also assumes DR responsibilities of sending PIM join and prune messages toward the RP to inform it about host group membership.



Assuming all links have the same cost, the topology in [Figure 16-28](#) illustrates the router links that would be elected as DFs.



**Figure 16-28** Bidir-PIM DF Election

For groups mapping to a given RP, the following responsibilities are uniquely assigned to the DF for that RP on each link:

1. The DF is the only router that forwards packets traveling downstream (toward the receivers) on to the link.
2. The DF is the only router that picks up upstream-traveling packets (away from the source) off the link to forward toward the RPL.

When the DF receives traffic on an interface (incoming or outgoing), it forwards the traffic out all interfaces (besides the one on which the traffic was received) in the outgoing interface list (OIL) and the incoming interface.

Non-DF routers on a link, which use that link as their RPF interface to reach the RP, may perform the following forwarding actions for bidirectional groups:

1. Forward packets from the link toward downstream receivers.
2. Forward packets from downstream sources upstream on to the link (provided they are the DF

for the downstream link from which the packet was picked up).

Figure 16-29 illustrates how the multicast distribution tree is created and the forwarding rules for DF and non-DF routers are applied.

1. Gamer 1 and 2 send an IGMPv2 join multicast group 239.255.1.1.
2. The IGMP join is processed by the DF of each segment.
3. The DF LHRs then send PIM join messages toward the RP.
4. All routers from the RP to the LHRs R1 and R4 generate a single multicast entry (\*,239.255.1.1) in their mroute table.
5. At this point, a shared tree is built between R1 and R4 and multicast data traffic is ready to start flowing bidirectionally.

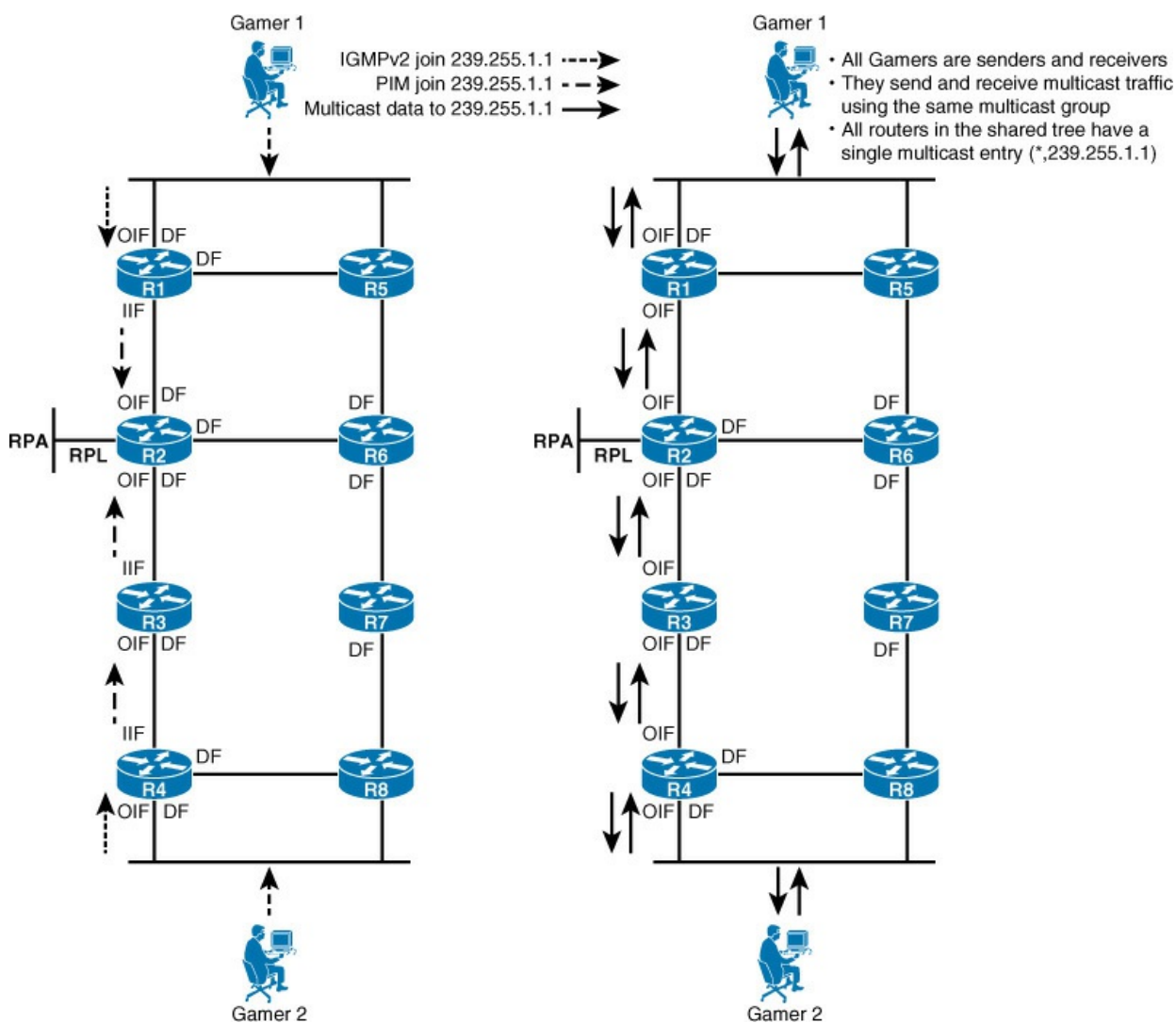


Figure 16-29 Bidir-PIM Forwarding Rules Example

The rule of the DF is the only router that picks up upstream traveling packets (away from the source) off the link to forward toward the RPL is fulfilled when Gamer 1 starts sending multicast traffic to group 239.255.1.1 and as this traffic flows up the tree, the IIF interface in R1 changes

into an OIF interface.

The rule of the DF is the only router that forwards packets traveling downstream (toward the receivers) on to the link is fulfilled once R2 forwards the packets downstream toward Gamer 2. These packets are forwarded from R2 downstream using the DF of R2, R3, and R4 until it reaches Gamer 2.

The rule of non-DFs forward packets from the link toward downstream receivers is fulfilled when R4 forwards multicast traffic coming from R3 into the Gamer 2 link.

The rule of non-DFs forward packets from downstream sources on to the link (provided they are the DF for the downstream link from which the packet was picked up) is fulfilled when Gamer 2 starts sending multicast traffic back to Gamer 1 on R3 and R4.

Figure 16-30 shows an example of what the same network would look like if all routers except for the RP had gamers on them. This shows how even though Bidir-PIM breaks the RPF forwarding rules, thanks to the DF mechanism, multicast route loops are avoided.

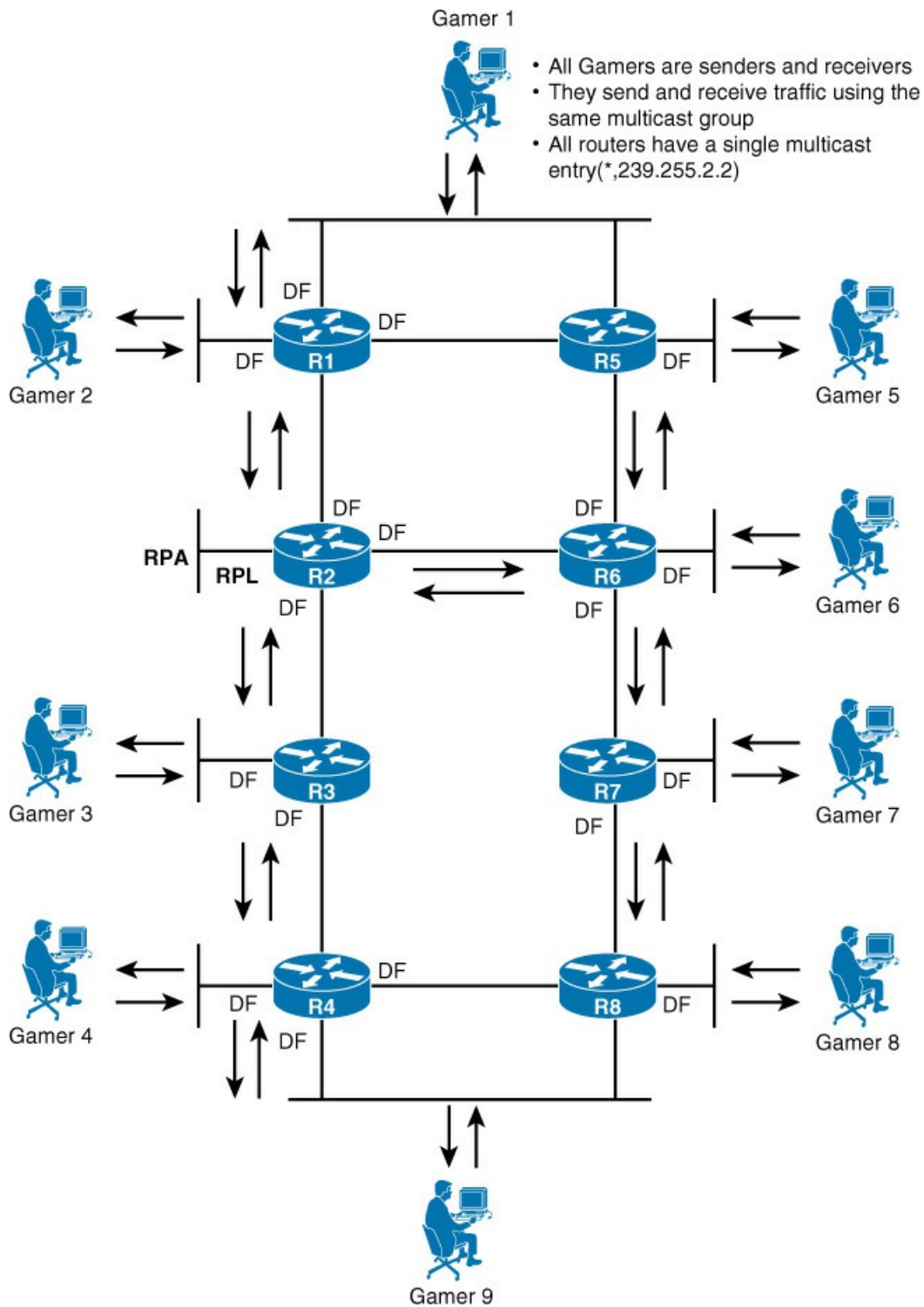


Figure 16-30 Bidir-PIM DF Route-Loop Avoidance

In IOS and IOS XR, the RP configuration for Bidir-PIM can be enabled via static RP, Auto-RP, or the BSR mechanism using the same commands as for PIM-SM. The only difference is that the commands for all three mechanisms need to include the keyword **bidir**.

In addition, in IOS, Bidir-PIM needs to be enabled with the command **ip pim bidir-enable** under global configuration mode.

## SUMMARY

Multicast communication is a one-to-many technology that optimizes network bandwidth utilization and conserves server load. This chapter presented an overview of the fundamentals of multicast, IGMP, PIM-DM, PIM-SM, and Bidir-PIM. It also includes basic multicast configuration and verification examples.

LAN multicast key concepts covered in this chapter include the following:

- The IGMP querier is responsible for periodically sending general membership query messages into the LAN segment to the *all-hosts group* address 224.0.0.1 to see whether there are any members remaining in the LAN segment.
- The DR on a LHR (router connected to the receivers) is responsible for sending PIM join and prune messages toward the RP to inform it about host group membership.
- The DR on a FHR (router connected to the source) is responsible for sending PIM register messages to the RP. Without DRs, multiple routers in the LAN segment could send either joins or register messages, resulting in duplicate multicast packets arriving into the LAN segment.
- The PIM forwarder is the only router in a LAN segment allowed to forward multicast traffic for a specific (S,G); otherwise, duplicate packets could flow into the LAN segment. The PIM assert mechanism determines who the PIM forwarder is.

PIM-DM key concepts covered in this chapter include the following:

- Uses a flood and prune mechanism to build multicast distribution trees
- Suitable for very small networks or lab environments
- Not supported by IOS XR

PIM-SM key concepts covered in this chapter include the following:

- A receiver sends explicit joins to build a shared tree, also known as RPT, which is made of (\*,G) routes from the RP to the receiver. The RPF interface for the (,G) entries is the interface pointing toward the RP.
- When a source starts transmitting, the multicast packets are sent by the FHR DR to the RP via PIM register messages (unicast messages to the RP address).
- An SPT (S,G) is built from the RP to the source.
- When the LHR receives multicast traffic from the RP, it performs an SPT switchover if there is a better path to the source and bypasses the RP.
- The LHR lets the RP know that there is no need to send multicast traffic down the shared tree.

PIM-Bidir key concepts covered in this chapter include the following:

- DF election based on routing protocol metric to the RP is one of the major differences between

Bidir-PIM and PIM-SM.

- There are no (S,G) entries.
- The RP does not need to be a physical router; it just needs to be a routable address known as the RPA or phantom RP.

## REFERENCES IN THIS CHAPTER

Williamson, Beau. *Developing IP Multicast Networks, Volume I*, Indianapolis: Cisco Press, 1999.

Cisco. Cisco IOS Software Configuration Guides, <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides, <http://www.cisco.com>

RFC 1112, *Host Extensions for IP Multicasting*, S. Deering, IETF, <http://www.ietf.org/rfc/rfc1112.txt>, August 1989

RFC 4601, *Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)*, B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, IETF, <http://www.ietf.org/rfc/rfc4601.txt>, August 2006

RFC 3973, *Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)*, A. Adams, J. Nicholas, W. Siadak, IETF, <http://www.ietf.org/rfc/rfc3973.txt>, January 2005

RFC 5015, *Bidirectional Protocol Independent Multicast (BIDIR-PIM)*, M. Handley, I. Kouvelas, T. Speakman, L. Vicisano, IETF, <http://www.ietf.org/rfc/rfc5015.txt>, October 2007

RFC 5059, *Bootstrap Router (BSR) Mechanism for Protocol Independent Multicast (PIM)*, N. Bhaskar, A. Gall, J. Lingard, S. Venaas, IETF, <http://www.ietf.org/rfc/rfc5059.txt>, January 2008

RFC 5771, *IANA Guidelines for IPv4 Multicast Address Assignments*, M. Cotton, L. Vegoda, D. Meyer, IETF, <http://www.ietf.org/rfc/rfc5771.txt>, March 2010

RFC 2236, *Internet Group Management Protocol, Version 2*, W. Fenner, IETF, <http://www.ietf.org/rfc/rfc2236.txt>, November 1997

RFC 3376, *Internet Group Management Protocol, Version 3*, B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan, <http://www.ietf.org/rfc/rfc3376.txt>, October 2002

RFC 4541, *Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches*, M. Christensen, K. Kimball, F. Solensky, <http://www.ietf.org/rfc/rfc4541>, May 2006

IPv4 Multicast Address Space Registry, Stig Venaas, <http://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml>, May 2014

## Chapter 17. Advanced IPv4 Multicast Routing

This chapter covers the following topics:

- Interdomain multicast routing
- Rendezvous point (RP) redundancy
- Source Specific Multicast (SSM)
- Multicast security
- Multicast traffic engineering
- Multicast troubleshooting

This chapter expands upon [Chapter 16, “IPv4 Multicast Routing,”](#) and covers advanced multicast technologies, strategies, and scenarios that you can apply to large complex networks to maintain performance, security, availability, stability, and scalability. Large multicast networks require additional features to provide scalability and reachability between routing domains and autonomous systems.

### INTERDOMAIN MULTICAST ROUTING

Protocol Independent Multicast (PIM) Sparse Mode (PIM-SM) was designed to operate in a single PIM domain (that is, intradomain routing). However, some IP multicast applications require the capability to operate in multiple PIM domains. To meet this requirement, features such as Multicast Source Discovery Protocol and Multiprotocol BGP (MBGP) provide native interdomain multicast service by allowing separate PIM-SM domains to route multicast traffic between them.

#### Multiprotocol BGP

RFC 4760 defines MBGP by specifying BGP address family extensions that extend the capabilities of BGP beyond IPv4 routing. BGP can distribute routing information for multiple address families, like IPv6 and L3VPN, along with the appropriate address family modifiers, like multicast and unicast. These extensions are backward compatible with BGP routers that do not support them. The multicast extension is the only extension covered in this chapter, so MBGP can be interpreted as multicast BGP and (M)BGP can be interpreted as multicast BGP or unicast BGP for the remainder of this chapter. MBGP does not carry multicast groups; MBGP carries unicast prefixes to perform multicast reverse path forwarding (RPF) check calculations.

MBGP maintains separate BGP tables for each protocol (address family + address family modifier). This implies that a separate unicast BGP table and a separate multicast BGP table exist within BGP. The unicast BGP table contains unicast prefixes used for unicast traffic forwarding, and the multicast BGP table contains the unicast prefixes that are used to perform RPF checking on multicast traffic.

MBGP achieves this separation by using the BGP path attributes (PAs) `MP_REACH_NLRI` and `MP_UNREACH_NLRI`. These attributes are carried inside BGP update messages and are used to carry network reachability information for different address families.

The Address Family Identifier (AFI) and Subsequent Address Family Identifier (SAFI) are two

fields that identify the specific address family within the network layer reachability information (NLRI). For example, an AFI with a value of 1 identifies the address family is IPv4, and the SAFI values that can be used in combination with this AFI are shown in [Table 17-1](#).

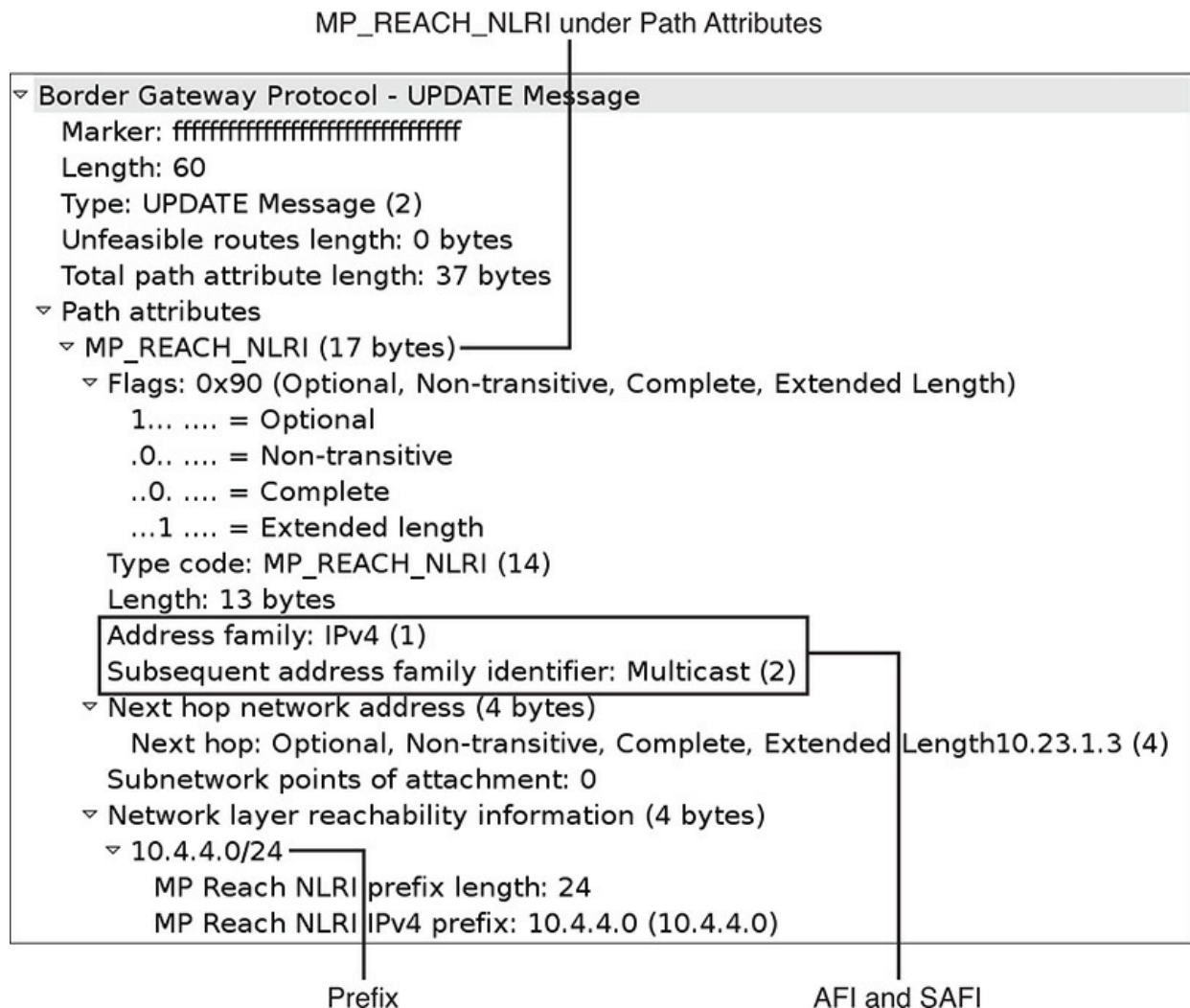
AFI	SAFI	Network Layer Information
1	1	IPv4 unicast
1	2	IPv4 multicast

**Table 17-1** IPv4 AFI = 1 Sub-Address Family Identifiers

Note

The full list of AFI numbers is maintained by the IANA and can be found online at <http://iana.org>.

[Figure 17-1](#) displays a packet capture of an MBGP update message. The MBGP update message includes an AFI with a value of 1 and a SAFI with a value of 2, which indicates the NLRI contained is for multicast. If the SAFI were 1, the NLRI would be for unicast. This is how MBGP can carry unicast and multicast RPF information in the same BGP updates.



**Figure 17-1** Packet Capture of MBGP Update Message



MBGP configuration for multicast in IOS and IOS XR uses the same address family CLI syntax as unicast except that the **multicast** SAFI keyword is used in the command **address-family ipv4** [address-family-modifier].

Figure 17-2 demonstrates a simple BGP topology using unicast BGP and multicast BGP with further details in the list that follows.

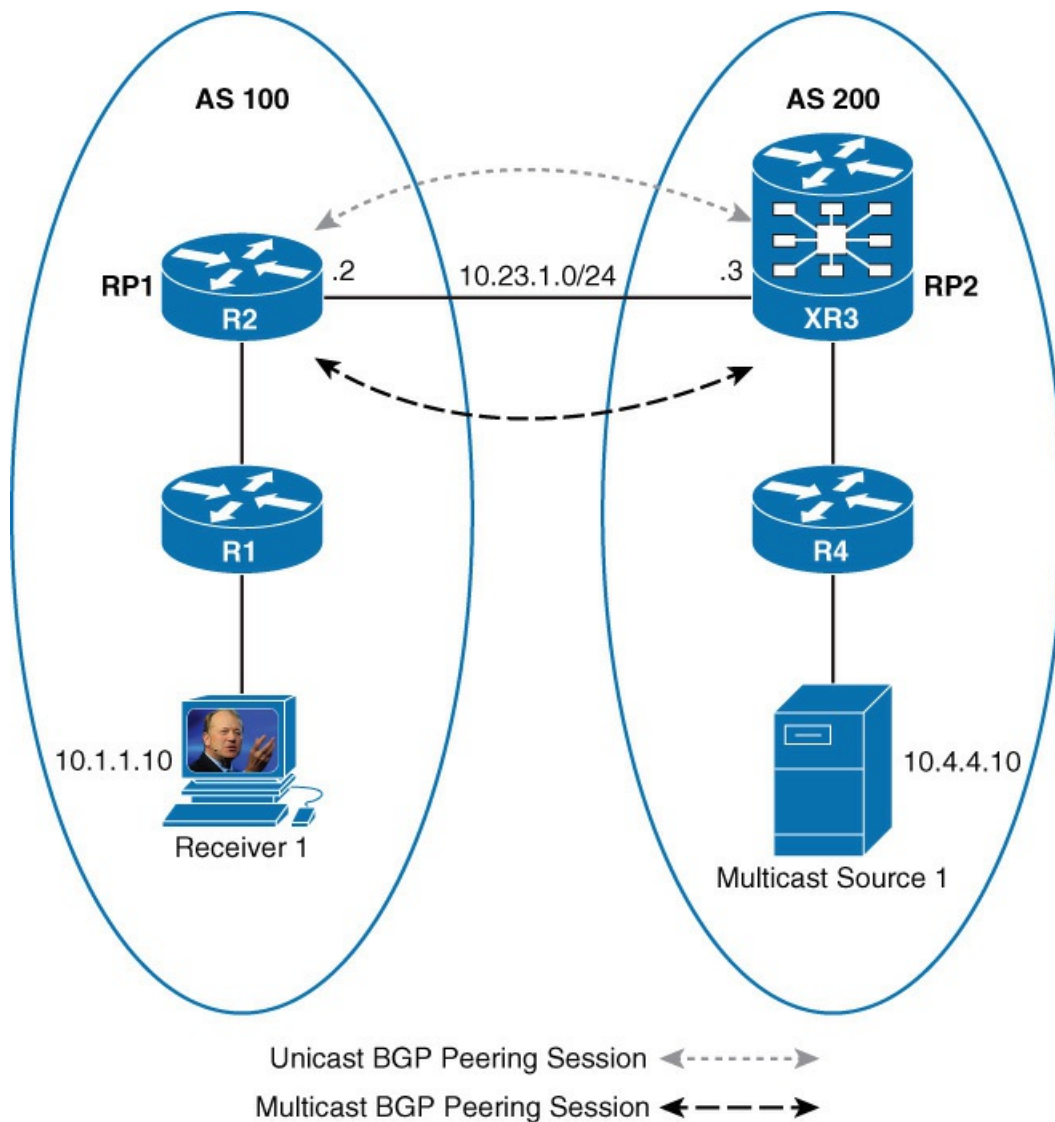


Figure 17-2 BGP Topology

- R2 (AS100) is peering with XR3 (AS200).
- Interdomain multicast routing needs to be performed in order for Receiver 1 to be able to join multicast feeds coming from Multicast Source 1.
- For interdomain multicast routing to work, AS100 needs to know how to reach the source subnet 10.4.4.0/24 to perform RPF checks.
- The source subnet could be learned via unicast BGP to perform RPF checks, but there is a requirement that dictates the source subnet should not be advertised into unicast BGP, so it is advertised into multicast BGP instead.

■ XR3 is advertising R4's loopback interface 192.168.4.4/32 into BGP's unicast table and the multicast source 2 subnet 10.4.4.0/24 into BGP's multicast table.

■ XR3 is advertising R4's loopback interface into unicast BGP just to demonstrate how the unicast BGP and multicast BGP tables are independent of each other in the coming examples.

Example 17-1 demonstrates the (M)BGP configuration for the topology shown in Figure 17-2. Multicast routing and PIM-SM need to be enabled in IOS in order for the (M)BGP routes to be accepted into the BGP multicast table. In IOS XR this is not a requirement, but it is necessary to enable it because without multicast routing, MBGP would not be needed.

### **Example 17-1** MBGP Configuration

[Click here to view code image](#)

```
R2
ip multicast-routing
!
interface GigabitEthernet0/1
 ip pim sparse-mode
!
interface GigabitEthernet0/2
 ip pim sparse-mode
!
router bgp 100
 bgp log-neighbor-changes
 neighbor 10.23.1.3 remote-as 200
!
 address-family ipv4
  neighbor 10.23.1.3 activate
 exit-address-family
!
 address-family ipv4 multicast
  neighbor 10.23.1.3 activate
 exit-address-family
```

**XR3**

```
route-policy PASSALL
  pass
end-policy
!
router bgp 200
  address-family ipv4 unicast
    network 192.168.4.4/32
  !
  address-family ipv4 multicast
    network 10.4.4.0/24
  !
  neighbor 10.23.1.2
    remote-as 100
    address-family ipv4 unicast
      route-policy PASSALL in
      route-policy PASSALL out
    !
    address-family ipv4 multicast
      route-policy PASSALL in
      route-policy PASSALL out
  !
multicast-routing
  address-family ipv4
    interface GigabitEthernet0/0/0/0
      enable
    !
    interface GigabitEthernet0/0/0/1
      enable
```

**Example 17-2** displays the unicast BGP table for R2 and XR3. The loopback interface from R4 192.168.4.4/32 is advertised into BGP's unicast table.

**Example 17-2 Unicast BGP Session Summary Verification**

**Click here to view code image**

```
R2#show ip bgp ipv4 unicast
! Output omitted for brevity
BGP table version is 3, local router ID is 192.168.2.2

   Network          Next Hop           Metric LocPrf Weight Path
* > 192.168.4.4/32  10.23.1.3          2             0 200 i

Processed 1 prefixes, 1 paths
```

```
RP/0/0/CPU0:XR3#show bgp ipv4 unicast
```

```
! Output omitted for brevity
```

```
BGP router identifier 192.168.3.3, local AS number 200
```

```
BGP main routing table version 6
```

```
Status codes: s suppressed, d damped, h history, * valid, > best
```

```
          i - internal, r RIB-failure, S stale, N Nexthop-discard
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 192.168.4.4/32	10.34.1.4	2		32768	i

```
Processed 1 prefixes, 1 paths
```

**Example 17-3** displays the multicast BGP table for R2 and XR3. The source subnet 10.4.4.0/24 is advertised into BGP's multicast table. Compare this to **Example 17-2** and notice how the unicast and multicast address family prefixes are kept in separate BGP tables (Loc-RIB databases).

### **Example 17-3** Multicast BGP Session Summary Verification

[Click here to view code image](#)

```
R2#show ip bgp ipv4 multicast
```

```
! Output omitted for brevity
```

```
BGP table version is 3, local router ID is 192.168.2.2
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 10.4.4.0/24	10.23.1.3	2		0	200 i

```
Processed 1 prefixes, 1 paths
```

```
RP/0/0/CPU0:XR3#show bgp ipv4 multicast
```

```
! Output omitted for brevity
```

```
BGP router identifier 192.168.3.3, local AS number 200
```

```
BGP main routing table version 6
```

```
Status codes: s suppressed, d damped, h history, * valid, > best
```

```
          i - internal, r RIB-failure, S stale, N Nexthop-discard
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 10.4.4.0/24	10.34.1.4	2		32768	i

```
Processed 1 prefixes, 1 paths
```

MBGP can be used for RPF discovery for multicast in cases where the unicast BGP table should not be modified. Implementing MBGP for multicast does not affect unicast BGP routing in any way.

## Multicast Source Discovery Protocol

Multicast Source Discovery Protocol (MSDP) allows RPs in the same or different PIM-SM domain to share information about active sources. For example, Figure 17-3 illustrates an interdomain PIM topology where Receiver 1 is in AS100 and the multicast source is in AS200. Multicast Source 1 is actively transmitting traffic for group 239.1.1.1 and XR3 (RP2) is aware there is an active source in the domain. R2 (RP1) needs to be aware of the active source in AS200, as well, so that if Receiver 1 decides to join the 239.1.1.1 group, it can build a tree to the source and send traffic to Receiver 1. This can be accomplished with MSDP.

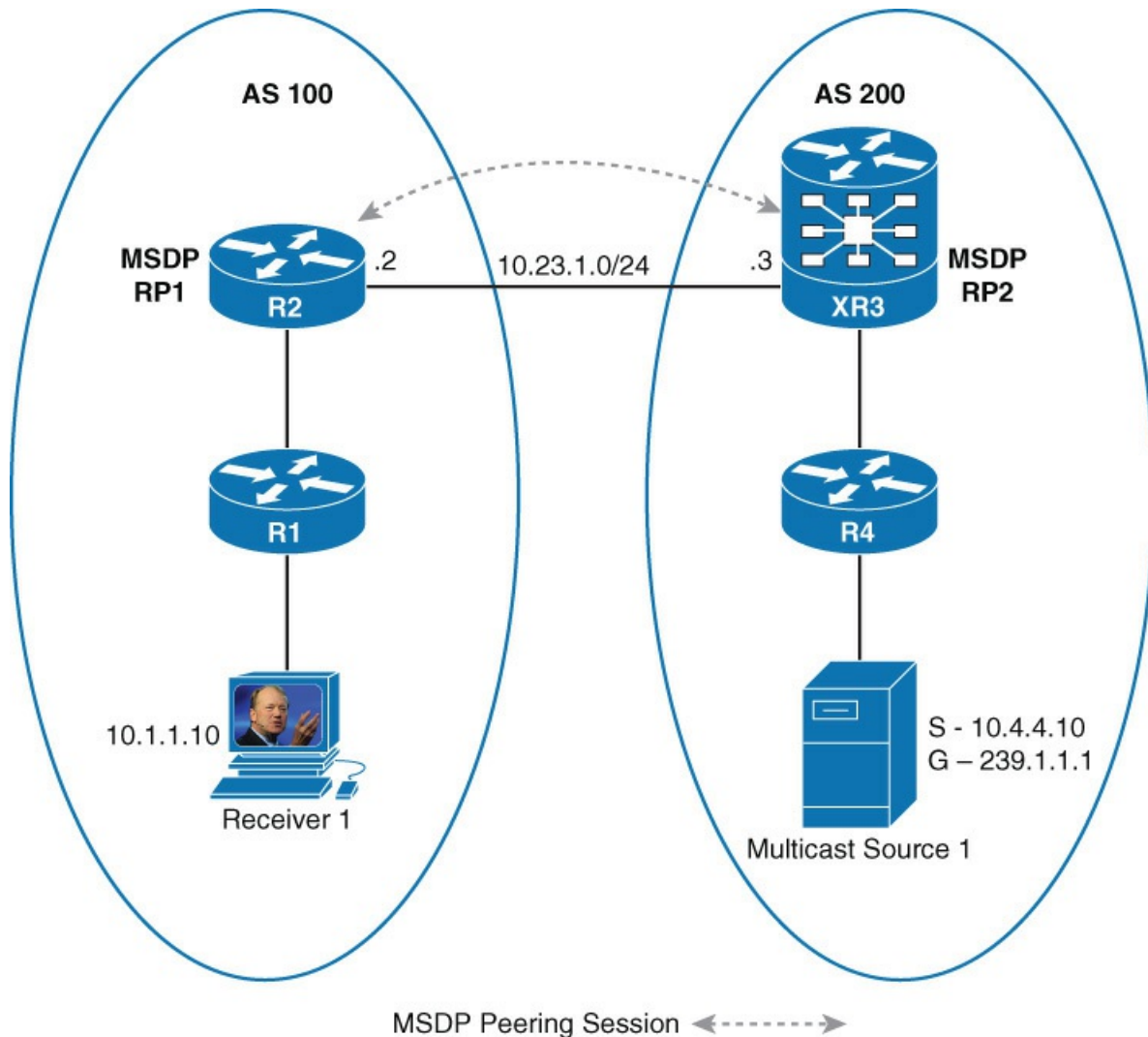


Figure 17-3 Interdomain PIM Topology with MSDP

### Note

MSDP relies on the RPs to know only about active sources; it works only with PIM-SM, and is not supported by the other PIM variations.

### Note

MSDP can also be used within a PIM-SM domain between anycast RPs, as discussed in the section, "Rendezvous Point Redundancy," later in this chapter.

## MSDP Source Active Message Types

MSDP source active (SA) messages are encoded in Type, Length, and Value (TLV) format. RFC 3618 defines four MSDP message types with their own TLV format.

- SA messages (Type 1)
- SA request messages (Type 2)
- SA response messages (Type 3)
- Keepalive messages (Type 4)

Out of these four messages, SA request messages and SA response messages are no longer used; they have not been deprecated by the RFC. SA request messages were used by MSDP speakers with SA caching disabled to request a list of active sources from MSDP peers with MSDP caching enabled, and SA response messages were sent in response to this request. Because RFC 3618 indicates that all MSDP peers must cache SA messages, these TLVs are not necessary for routers compliant with the RFC and so are not discussed further in this chapter.

### **SA Messages**

SA messages advertise active sources in a PIM-SM domain. SA messages contain the IP address of the originating RP and one or more of the (S,G) pairs being advertised.

### **Keepalive Messages**

Keepalive messages are sent every 60 seconds to keep the MSDP session active. Data arrival performs the same function as keepalives and keeps session from timing out. By default, if no keepalive or SA messages are received within 75 seconds, the MSDP session is reset and reopened.

Figure 17-4 demonstrates how MSDP exchanges active source information between PIM service provider (SP) domains. The topology illustrates four different PIM domains, where each RP is an MSDP speaker. The dashed lines between the RPs represent TCP MSDP sessions between them. MSDP SA exchanges between the PIM domains take place as follows. The list that follows provides additional information related to the figure.

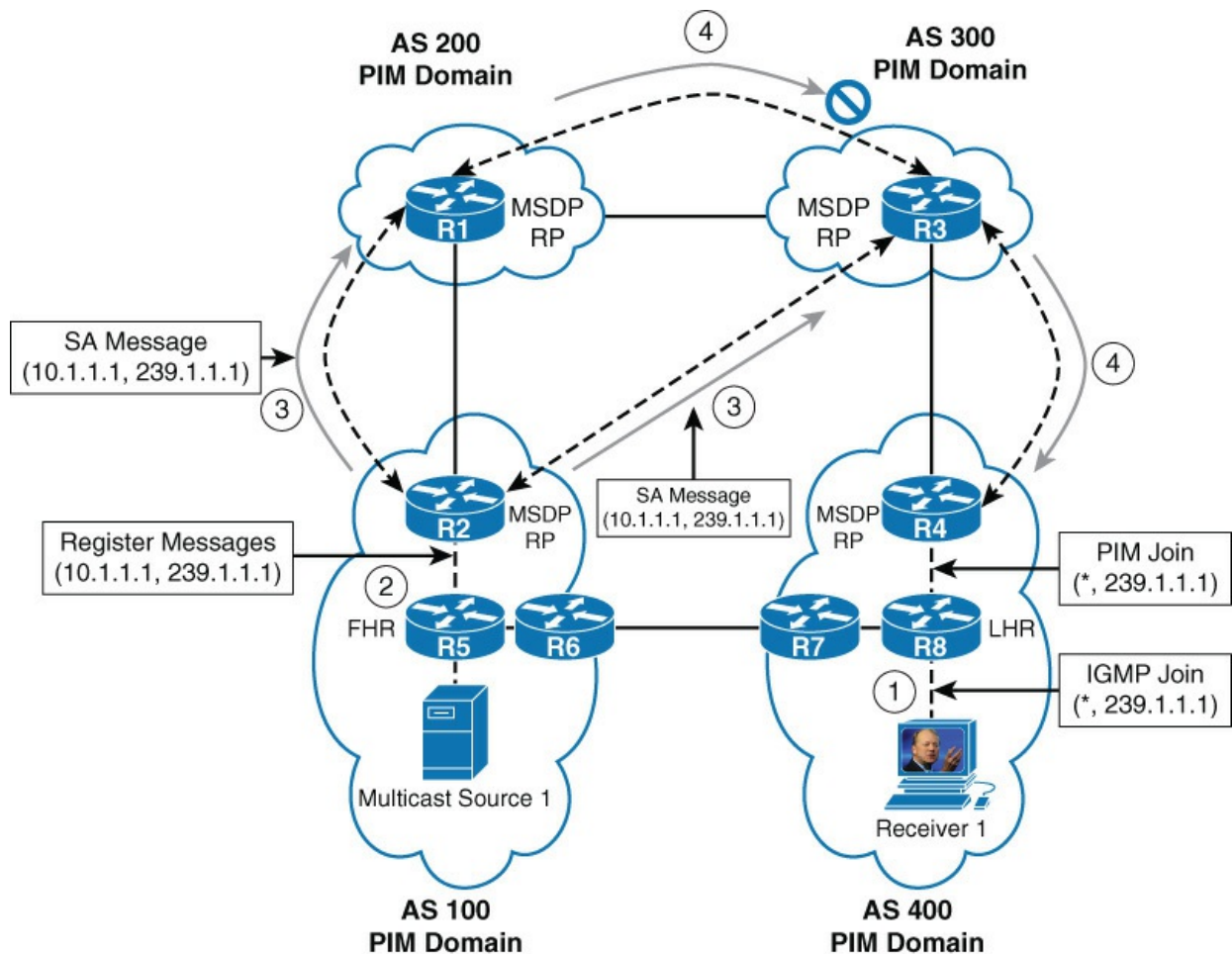


Figure 17-4 MSDP SA Exchange

**Step 1.** Receiver 1 sends an IGMP join to (\*,239.1.1.1), which causes the last-hop router (LHR) (R8) to send a PIM join for this group to the RP (R4), and a shared tree is built between the R4 and R8.

**Step 2.** When the multicast source in AS100 starts sending multicast traffic for group 239.1.1.1, the first-hop router (FHR) (R5) starts sending register messages to the RP (R2). At this point, R2 is aware that a source is active in the local PIM domain.

**Step 3.** R2 then starts to send MSDP source active (SA) messages to its MSDP peers R1 and R3 in the AS200 and AS300 PIM domains, respectively, and it will continue to do this periodically every 60 seconds by default as long as the source is active.

**Step 4.** The SA messages received by R1 and R3 are RPF checked and then forwarded downstream to their own MSDP peers. For example, R3 will send SA messages to R4, and R1 will send SA messages to R3. The SA messages from R1 to R3 fail the RPF check and are dropped because there is a shortest AS path to AS100 through R2. The SA messages from R2 to R3 pass the RPF check and are accepted and forwarded to R4.

The RPs perform the RPF checks by consulting the BGP next hop in the BGP table to determine the next hop toward the originator of the SA message. If both MBGP and unicast BGP are configured, MBGP is checked first, followed by unicast BGP. The next-hop neighbor is the RPF peer for the originator. SA messages that are received from the originator on any interface other

than the interface to the RPF peer are dropped. For this reason, the SA message flooding process is referred to as *peer-RPF forwarding*. Because of the peer-RPF forwarding mechanism, BGP or MBGP must be running along with MSDP.

Note

Default MSDP peers and MSDP mesh groups are examples of scenarios that do not require the use of (M)BGP. These scenarios are covered later in this chapter.

Figure 17-5 demonstrates what happens after MSDP exchanges active source information between PIM-SM domains. The list that follows goes into more detail.

- On the topology on the left side, the RP (R4) in AS400 receives the SA message from R3, it checks to see whether there are any interested receivers for the advertised group. If there are none, the R4 does nothing; but because it knows it has an interested receiver (Receiver 1) attached to it, it sends an (S,G) PIM join to the FHR (R5) in AS100. The (S,G) join is sent out of the RP's RPF interface to the source.

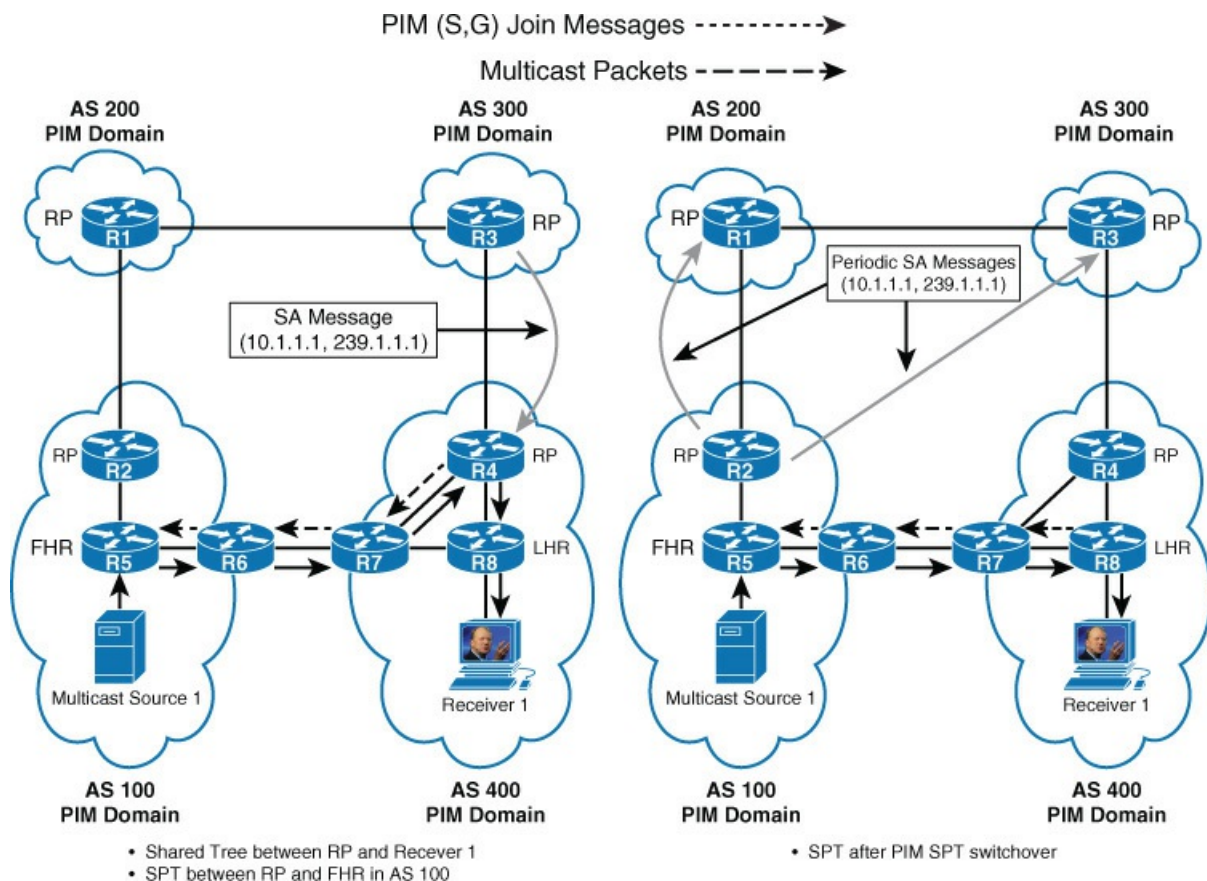


Figure 17-5 MSDP Topology MDT Creation

Note

PIM and multicast data follow RPF routing information and do not follow the MSDP peering path.

- When the (S,G) join arrives at the FHR (R5) in AS100, the multicast traffic begins to flow from R5 to the RP (R4) in AS400 and from R4 to Receiver 1.



- When the multicast traffic arrives at the LHR (R8) in AS400, R8 performs an SPT switchover, as illustrated in the topology on the right side of [Figure 17-5](#). At this point, multicast traffic starts flowing between the FHR (R5) and the LHR (R8), bypassing the RP (R4) in AS400.

- The originating RP (R2) continues to send periodic SA messages for the (S,G) every 60 seconds for as long as the source is actively sending packets to the group.

When an RP receives an SA message, it caches it. SA caching reduces join latency for new receivers of a group. For example, if the RP in AS300 has a receiver that wants to join the same group that receiver 1 in AS400 joined, the RP checks its SA cache to see whether there are active sources; if there are, it can join the source tree as soon as it receives the request from the receiver. If SA caching is disabled, it must wait for the next periodic SA message, which can take up to 60 seconds by default.

Note

SA caching is enabled by default in IOS and IOS XR and cannot be disabled

### MSDP Peers

MSDP peers establish neighbor relationships in a very similar fashion to BGP by forming sessions using TCP (port 639). Using TCP eliminates the need to implement fragmentation, retransmission acknowledgment and sequencing into MSDP. The lowest IP address peer is the one that initiates the connection while the highest IP address peer waits in the listen state for the other peer to make the connection.

In IOS, the following is required for MSDP peer configuration to work:

- Enable multicast routing
- Enable PIM on all relevant interfaces including the interdomain links
- Configure RPs (static or dynamic)

After these requirements have been implemented, you configure MSDP peers with the command **ip msdp peer** {*peer-name*| *peer-address*} [**connect-source** *interface-type interface-number*] [**remote-as** *as-number*] under global configuration mode.

MSDP must be enabled on all RPs that participate in interdomain routing. The keyword **connect-source** performs the same function as the keyword **update-source** for BGP. It should be configured using a PIM-enabled loopback interface as the source because a loopback interface is always up and because the likelihood of the MSDP peering session going down is much lower during a physical link failure. If this command is not used, the IP address of the physical interface toward the peer is used as the source address. The **remote-as** keyword is optional. If it is not configured, the remote autonomous system value is derived from (M)BGP (if configured) or set to 0, when only Interior Gateway Protocol (IGP) is present.

The MSDP configuration for IOS XR occurs in the protocol-specific configuration mode to provide an easy way for enabling, disabling, and configuring multicast features on a large number of interfaces.

Just as with IOS, the following is required for MSDP peer configuration to work:

- Enable multicast routing and PIM on all relevant interfaces including the interdomain links.
- Configure RPs (static or dynamic).

After implementing the preceding requirements, perform the following steps to enable MSDP:

### **Step 1. Enter MSDP configuration mode.**

Enter MSDP configuration mode with the command **router msdp**.

### **Step 2. Configure MSDP peer(s).**

MSDP peers are configured with the command **peer peer-address**.

### **Step 3. Configure an MSDP source interface.**

The source interface for the MSDP peering is set with the command **connect-source [interface-type interface-number]** within the MSDP configuration mode. The interface specified should be a loopback interface.

The remote autonomous system for the MSDP session is set with the command **remote-as as-number** under MSDP configuration mode.

In addition, MSDP relies on either autonomous system path (AS\_Path) or next hop information toward the originating RP to perform RPF checks on SA messages. This can be achieved by enabling either unicast BGP, multicast BGP, or by using an IGP for routers compliant with the latest RPF rules in RFC 3618 that allow for an MSDP peer-RPF check using an IG.P (IOS and IOS XR are compliant.)

In some scenarios, MSDP SA RPF checks are not necessary:

- When there is only a single MSDP peer connection configured on the router
- When a default peer (default MSDP route) is configured
- When the originating RP is directly connected
- When MSDP mesh groups are used

Some examples of these scenarios are covered in the “[MSDP Stub Networks](#)” and “[Anycast RP](#)” sections in this chapter.

In IOS, a feature called *MSDP Compliance with IETF RFC 3618* enables MSDP to comply with the peer-RPF forwarding rules defined in RFC 3618 specifications. Enabling this feature prevents SA message loops. In addition, it eliminates the requirement for BGP route reflectors (RRs) to

run MSDP, enables the use of an IGP for the RPF check, and allows MSDP peerings between routers in non-directly connected autonomous systems. This feature can be enabled with the global configuration command **ip msdp rpf rfc3618**.

Figure 17-6 demonstrates the usage of MBGP and MSDP to provide multicast connectivity between three PIM domains, which is further detailed in the list that follows.

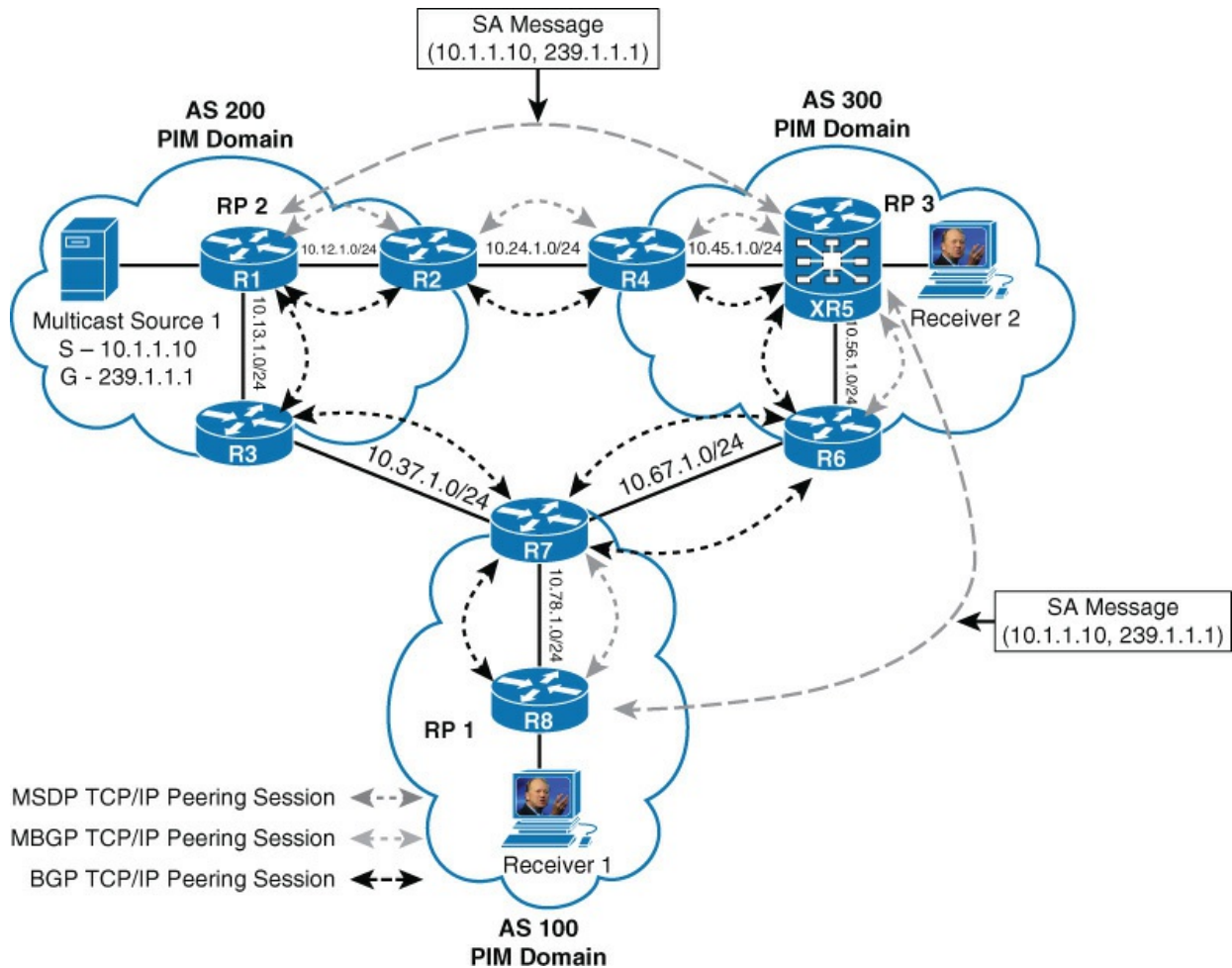


Figure 17-6 Topology Demonstrating MSDP and MBGP Sessions Between PIM Domains

- There are MSDP peering connections between the route reflectors XR5 and R1, and XR5 and R8.
- There are unicast BGP sessions between all routers, and MBGP is configured throughout all three domains except between R1 and R3, and R3 and R7.
- R2 is advertising reachability to R1's loopback via BGP and MBGP, the unicast BGP prefix is used for MSDP peering, and the MBGP prefix is used by XR5 to perform RPF checks on SA messages coming from R1.
- XR5's loopback is being advertised into BGP and MBGP, the BGP prefix is used for MSDP peering, and the MBGP prefix is used by R8 to perform RPF checks on SA messages coming from XR5.
- R2 is advertising the IP address of the multicast source via MBGP, and R3 is advertising the IP

address of the multicast source via BGP. This will result in R7 receiving reachability information to the 10.1.1.0/24 prefix (source subnet) via unicast BGP from R3 and MBGP via R6.

■ Because RPF checks are performed by checking the MBGP table first, R6 will be elected as R7's RPF peer. This means that the source tree will be created in the direction of AS300. This was done to demonstrate how MBGP could be used to manipulate the multicast data path.

**Example 17-4** demonstrates the MSDP and BGP configuration for the topology illustrated in **Figure 17-6**.

### **Example 17-4** MSDP and MBGP Configuration Example

[Click here to view code image](#)

#### **R1**

```
router bgp 200
!
address-family ipv4
  neighbor 192.168.2.2 activate
  neighbor 192.168.2.2 route-reflector-client
  neighbor 192.168.3.3 activate
  neighbor 192.168.3.3 route-reflector-client
exit-address-family
!
address-family ipv4 multicast
  neighbor 192.168.2.2 activate
exit-address-family
!
ip pim rp-address 192.168.1.1
!
ip msdp peer 192.168.5.5 connect-source Loopback0 remote-as 300
```

#### **R2**

```
router bgp 200
!
address-family ipv4
  network 192.168.1.1 mask 255.255.255.255
  neighbor 10.24.1.4 activate
  neighbor 192.168.1.1 activate
  neighbor 192.168.1.1 next-hop-self
exit-address-family
!
address-family ipv4 multicast
  network 192.168.1.1 mask 255.255.255.255
  neighbor 10.24.1.4 activate
  neighbor 192.168.1.1 activate
  neighbor 192.168.1.1 next-hop-self
exit-address-family
```

**R3**

```
router bgp 200
  bgp log-neighbor-changes
  neighbor 10.37.1.7 remote-as 100
  neighbor 192.168.1.1 remote-as 200
  neighbor 192.168.1.1 update-source Loopback0
  !
  address-family ipv4
    network 10.1.1.0 mask 255.255.255.0
    network 192.168.1.1 mask 255.255.255.255
  neighbor 10.37.1.7 activate
  neighbor 192.168.1.1 activate
  neighbor 192.168.1.1 next-hop-self
  exit-address-family
```

**XR5**

```
router bgp 300
  address-family ipv4 unicast
  !
  address-family ipv4 multicast
  !
  neighbor 192.168.4.4
    remote-as 300
    update-source Loopback0
  address-family ipv4 unicast
    route-reflector-client
  !
  address-family ipv4 multicast
    route-reflector-client
  !
  !
  neighbor 192.168.6.6
    remote-as 300
    update-source Loopback0
  address-family ipv4 unicast
    route-reflector-client
  !
  address-family ipv4 multicast
    route-reflector-client
  !
  router msdp
  peer 192.168.1.1
    connect-source Loopback0
    remote-as 200
  !
  peer 192.168.8.8
    connect-source Loopback0
    remote-as 100
  !
  !
  router pim
  address-family ipv4
    rp-address 192.168.5.5
```

**R6**

```
router bgp 300
  bgp log-neighbor-changes
  neighbor 10.67.1.7 remote-as 100
  neighbor 192.168.5.5 remote-as 300
  neighbor 192.168.5.5 update-source Loopback0
  !
  address-family ipv4
    network 192.168.5.5 mask 255.255.255.255
    neighbor 10.67.1.7 activate
    neighbor 192.168.5.5 activate
    neighbor 192.168.5.5 next-hop-self
  exit-address-family
  !
  address-family ipv4 multicast
    network 192.168.5.5 mask 255.255.255.255
    neighbor 10.67.1.7 activate
    neighbor 192.168.5.5 activate
    neighbor 192.168.5.5 next-hop-self
  exit-address-family
```

**R8**

```
router bgp 100
  !
  address-family ipv4
    neighbor 192.168.7.7 activate
  exit-address-family
  !
  address-family ipv4 multicast
    neighbor 192.168.7.7 activate
  exit-address-family

ip pim rp-address 192.168.8.8

ip msdp peer 192.168.5.5 connect-source Loopback0 remote-as 300
```

Once the source becomes active, MSDP SA messages are sent from R1 to XR5 and from XR5 to R8. All of these routers will store the SA information in their SA cache.

[Figure 17-7](#) shows a packet capture of the SA message sent from R1 to XR5. The packet capture indicates MSDP is using TCP destination port 639 and that it is a Type 1 SA message. The capture also indicates the group for which it is sending SA messages for, in addition to the multicast source for the group. The RP address is the originating RP's address. [Figure 17-8](#) shows a packet capture of the SA message sent from XR5 to R8. Notice that the only difference between this SA message and the one shown in [Figure 17-7](#) is the destination IP address and the TCP source port; the rest of the information in the SA message is all the same.

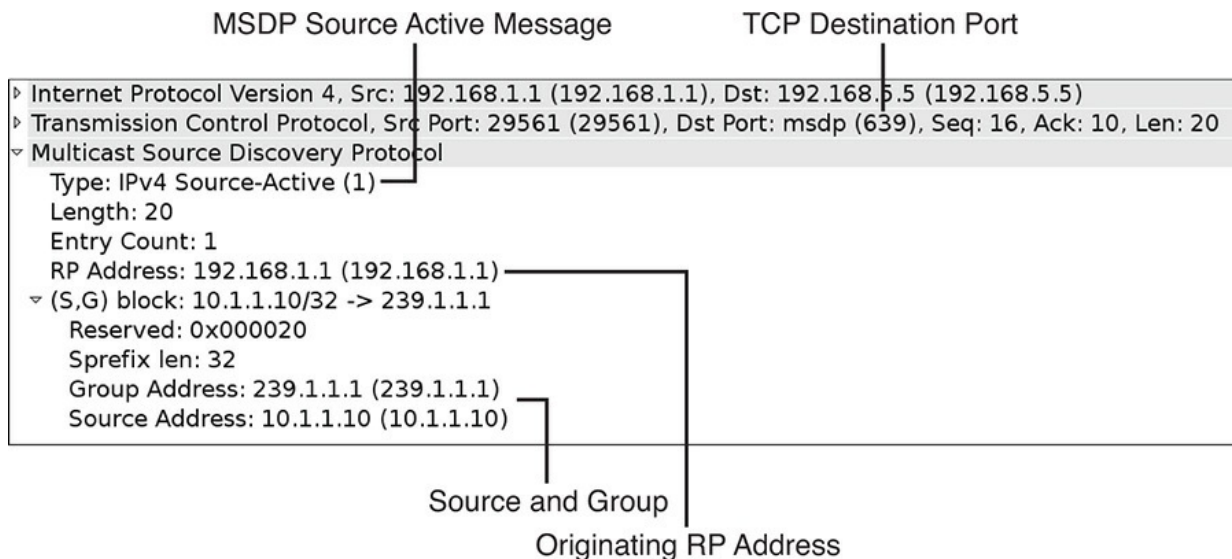


Figure 17-7 Packet Capture for MSDP SA Message from R1 to XR5

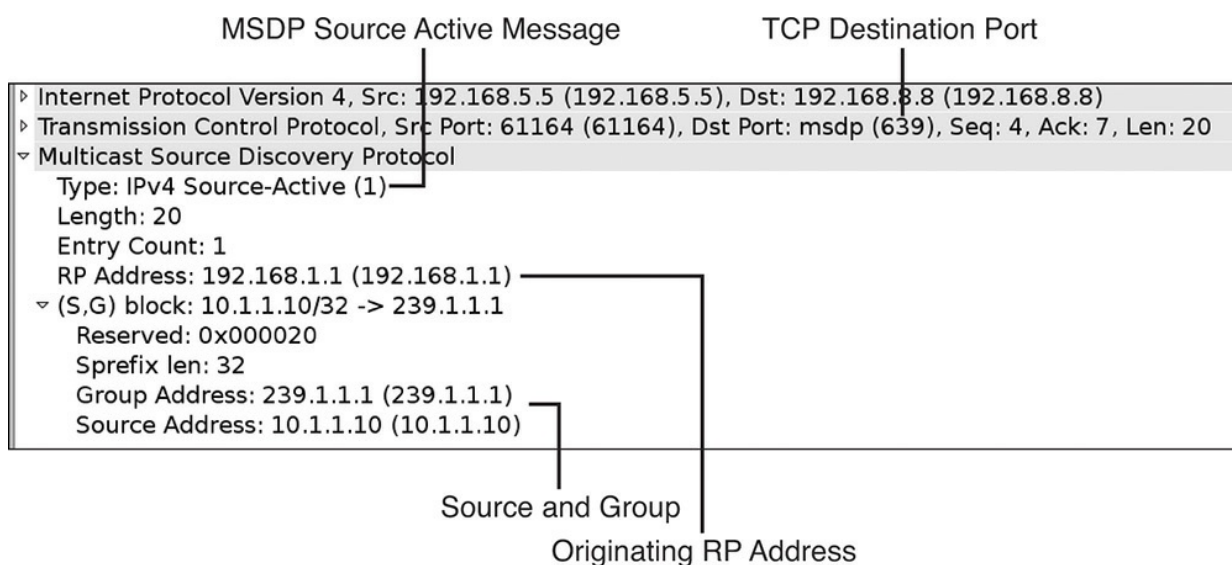


Figure 17-8 Packet Capture for MSDP SA Message from XR5 to R8

## MSDP Verification

This section covers the monitoring and verification of MSDP in IOS and IOS XR routers by using **show** commands.

MSDP peer status can be displayed with the IOS command **show ip msdp summary** and the IOS XR command **show msdp summary**. [Example 17-5](#) displays the MSDP session information between R1 and XR5 and XR5 and R8. Highlighted in all routers are the MSDP peer IP address, MSDP peer autonomous system, and MSDP session state.

### Example 17-5 MSDP Peer Status on R1 and XR5

[Click here to view code image](#)

```
R1#show ip msdp summary
MSDP Peer Status Summary
Peer Address      AS      State    Uptime/  Reset SA  Peer Name
                  Downtime Count Count
192.168.5.5      300    Up       00:19:41 0      0      ?
```

```
RP/0/0/CPU0:XR5#show msdp summary
Out of Resource Handling Enabled
Maximum External SA's Global : 20000
Current External Active SAs : 0
```

#### MSDP Peer Status Summary

Peer Address	AS	State	Uptime/ Downtime	Reset Count	Peer Name	Active SA Cnt	Cfg.Max Ext.SAs	TLV recv/sent
192.168.1.1	200	Up	00:29:30	0	?	0	0	240/59
192.168.8.8	100	Up	00:31:08	0	?	0	0	265/63

#### R8#show ip msdp summary

##### MSDP Peer Status Summary

Peer Address	AS	State	Uptime/ Downtime	Reset Count	SA Count	Peer Name
192.168.5.5	300	Up	00:22:24	0	0	?

Receiver 2 in AS300 sends an Internet Group Message Protocol (IGMP) request for 239.1.1.1. Because XR5 has information in its SA cache about this group, it sends a PIM join message upstream and it starts receiving multicast traffic, as shown in [Example 17-6](#).

### Example 17-6 XR5 Mroute Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR5#show mrib route 239.1.1.1
! Output omitted for brevity

(*,239.1.1.1) RPF nbr: 192.168.5.5 Flags: C
Up: 03:40:40
Incoming Interface List
  Decapstunnel0 Flags: A NS, Up: 03:40:40
Outgoing Interface List
  GigabitEthernet0/0/0/2 Flags: F NS LI, Up: 03:40:40

(10.1.1.10,239.1.1.1) RPF nbr: 10.45.1.4 Flags:
Up: 00:09:35
Incoming Interface List
  GigabitEthernet0/0/0/0 Flags: A, Up: 00:02:57
Outgoing Interface List
  GigabitEthernet0/0/0/2 Flags: F NS, Up: 00:09:35
```

The MSDP SA cache information can be displayed with the IOS command **show ip msdp sa-cache** and the IOS XR command **show msdp sa-cache**. [Example 17-7](#) shows XR5's MSDP SA cache after receiving an MSDP SA message from R1 for group 239.1.1.1 and source 10.1.1.10.

### Example 17-7 XR5 MSDP SA Cache

[Click here to view code image](#)



```
RP/0/0/CPU0:XR5#show msdp sa-cache
```

```
MSDP Flags:
```

```
E - set MRIB E flag , L - domain local source is active,  
EA - externally active source, PI - PIM is interested in the group,  
DE - SAs have been denied. Timers age/expiration,
```

```
Cache Entry:
```

```
(10.1.1.10, 239.1.1.1), RP 192.168.1.1, MBGP/AS 200, 00:01:02/00:01:32  
Learned from peer 192.168.1.1, RPF peer 192.168.1.1  
SAs recvd 2, Encapsulated data received: 0  
grp flags: PI, src flags: E, EA, PI
```

[Example 17-8](#) shows how R8 stores the SA message it received from XR5 in its MSDP SA cache for group 239.1.1.1. With this information in the cache, as soon as Receiver 1 sends an IGMP join to group 239.1.1.1, R8 can immediately send a PIM join upstream to the source and does not have to wait for the next periodic SA message to be received.

### Example 17-8 R8 MSDP SA Cache

[Click here to view code image](#)

```
R8#show ip msdp sa-cache
```

```
MSDP Source-Active Cache - 1 entries
```

```
(10.1.1.10, 239.1.1.1), RP 192.168.1.1, MBGP/AS 200, 00:05:19/00:05:21, Peer  
192.168.5.5
```

[Example 17-9](#) shows how once Receiver 1 joins group 239.1.1.1, R8 joins the SPT to the source and starts receiving multicast traffic. The output also indicates that the RPF neighbor information was discovered via MBGP updates.

### Example 17-9 R8 Mroute Table

[Click here to view code image](#)

```
R8#show ip mroute 239.1.1.1
```

```
! Output omitted for brevity
```

```
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
```

```
Timers: Uptime/Expires
```

```
Interface state: Interface, Next-Hop or VCD, State/Mode
```

```
(* , 239.1.1.1), 04:22:56/stopped, RP 192.168.8.8, flags: SJC
```

```
Incoming interface: Null, RPF nbr 0.0.0.0
```

```
Outgoing interface list:
```

```
GigabitEthernet0/2, Forward/Sparse, 00:00:08/00:02:51
```

```
(10.1.1.10, 239.1.1.1), 00:31:14/00:02:51, flags: T
```

```
Incoming interface: GigabitEthernet0/1, RPF nbr 10.78.1.7, Mbgp
```

```
Outgoing interface list:
```

```
GigabitEthernet0/2, Forward/Sparse, 00:00:08/00:02:51
```

[Example 17-10](#) shows that the RPF neighbor on R7 is R6's interface (10.67.1.6) and that this

information was received via MBGP. Because R6 is the RPF neighbor, PIM joins will be sent to it and the MDT will be formed in this direction.

### Example 17-10 RPF Check for Source Subnet

[Click here to view code image](#)

```
R7#show ip rpf 10.1.1.0
RPF information for ? (10.1.1.0)
  RPF interface: GigabitEthernet0/2
  RPF neighbor: ? (10.67.1.6)
  RPF route/mask: 10.1.1.0/24
  RPF type: multicast (bgp 100)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base
```

Example 17-11 shows that if MBGP were not configured between R6 and R7, the RPF neighbor on R7 would be R3's interface (10.37.1.3) instead of R6's interface (10.67.1.6) and PIM joins would be sent in that direction to form the MDT. The output also indicates that the RPF information was learned via unicast BGP.

### Example 17-11 RPF Check for Source Subnet Without MBGP Between R6 and R7

[Click here to view code image](#)

```
R7#show ip rpf 10.1.1.0
RPF information for ? (10.1.1.0)
  RPF interface: GigabitEthernet0/1
  RPF neighbor: ? (10.37.1.3)
  RPF route/mask: 10.1.1.0/24
  RPF type: unicast (bgp 100)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

## MSDP Stub Networks

Stub networks are typically not configured with BGP and rely on static default routes to reach external networks. As mentioned earlier in this section, MSDP relies on BGP path information to learn the MSDP topology for the SA peer-RPF check, but there are certain special cases, such as stub networks, in which MSDP can be deployed without BGP while allowing the RPF checks to be skipped.

Figure 17-9 illustrates a network with a stub autonomous system network (AS100). To configure MSDP on a stub network in IOS, the command **ip msdp default-peer** {ip-address | peer-name} [**prefix-list** list] is used and in IOS XR, the command **default-peer** ip-address [**prefix-list** list] under **router msdp** configuration mode is used. When this command is used, the RPF checks are skipped.

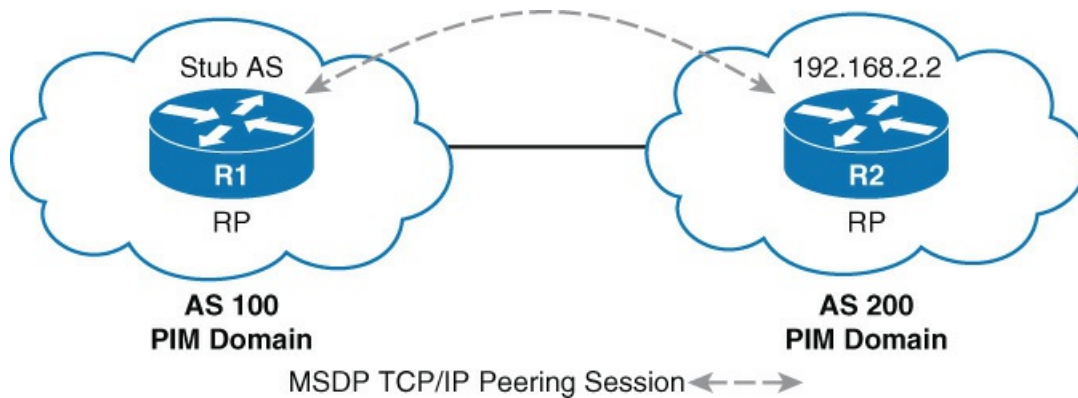


Figure 17-9 MSDP Default Peering for Stub Networks

Example 17-12 demonstrates the MSDP configuration for peering on stub networks.

### Example 17-12 MSDP Default Peering Configuration Example for Stub Networks

[Click here to view code image](#)

#### IOS

```
ip msdp default-peer 192.168.2.2
```

#### XR

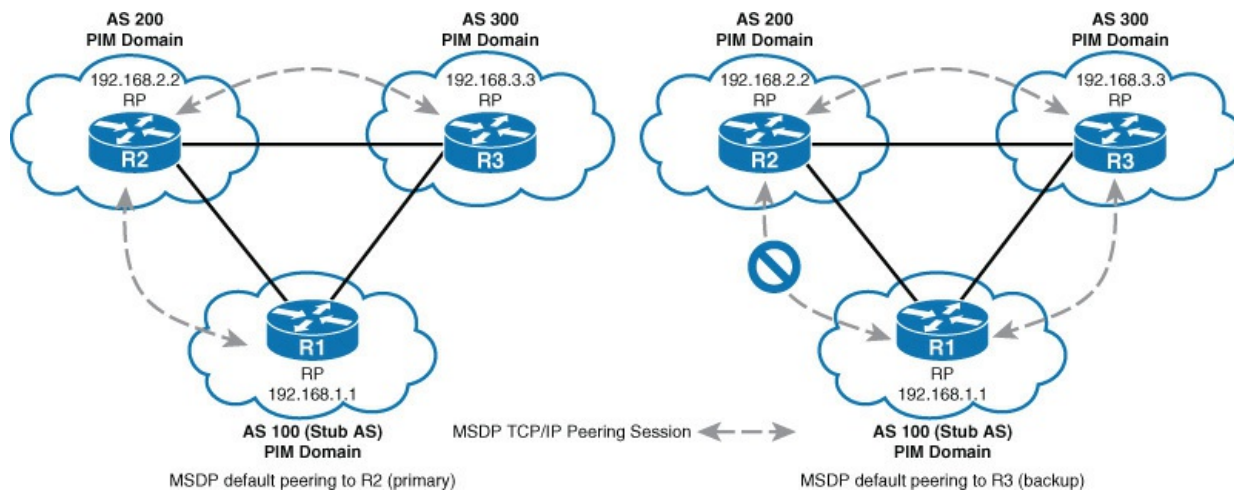
```
router msdp
 default-peer 192.168.2.2
```

#### Note

Stub networks may also be configured using the normal IOS command `ip msdp peer` or the IOS XR command `msdp peer` when there is only a single MSDP peer configured. When this is the case, it is treated exactly in the same manner as a default peering and RPF checks will be skipped.

Stub networks can use additional secondary MSDP default peer connections to provide some redundancy in case the primary MSDP default peer goes down. Only the primary MSDP peer connection is active while the backup is in standby. The backup MSDP peer connection only becomes active when the primary MSDP peering goes down.

Figure 17-10 illustrates the primary MSDP default peer connection is to R2 (192.168.2.2). The secondary default peer connection is to R3 (192.168.3.3). The secondary peering is not activated unless the peering to R2 is lost.



**Figure 17-10** Multiple Default MSDP Peers on Stub Networks

**Example 17-13** shows a sample configuration for multiple default MSDP peerings on stub networks. The first line of the command to appear in the configuration acts as the primary MSDP default peer, and the second line is the backup.

**Example 17-13** MSDP Default Peering with Backup Configuration Example for Stub Networks

[Click here to view code image](#)

**IOS**

```
ip msdp default-peer 192.168.2.2
ip msdp default-peer 192.168.3.3
```

**XR**

```
router msdp
 default-peer 192.168.2.2
 default-peer 192.168.3.3
```

## RENDEZVOUS POINT REDUNDANCY

A PIM domain should be designed with resiliency, high availability, performance, and security in mind. This section describes the most common practices that can be implemented to achieve these design principles.

When there is a single RP deployed in a PIM domain, it might become inaccessible for any number of reasons, such as high CPU or because its links are congested and so forth. If this happens, this creates a single point of failure for all multicast traffic. The most common best practice to avoid this unpleasant outcome is to deploy multiple RPs in the PIM domain.

There are multiple ways to deploy redundant RPs:

- Auto-RP with multiple RPs
- Bootstrap Router (BSR) with multiple RPs
- Static RP with multiple RPs
- Anycast RP

Figure 17-11 provides a reference topology to describe the various techniques to provide RP redundancy.

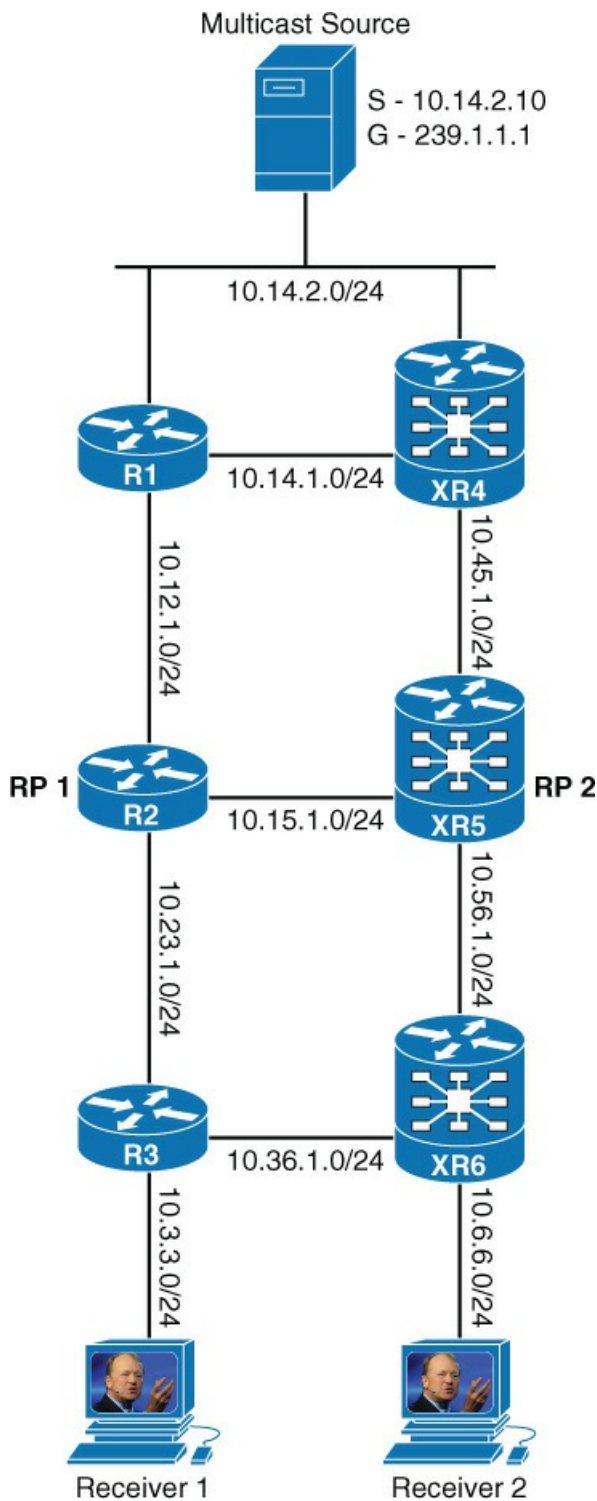


Figure 17-11 PIM-SM Topology with Redundant RPs

### Auto-RP with Multiple RPs

As discussed in Chapter 16, the candidate rendezvous points (C-RPs) send RP-announce messages every 60 seconds to the group address 224.0.1.39. Mapping agents (MAs) listen to these messages and store the information contained in these messages in a group-to-RP mapping cache along with hold times. The MAs then advertise the group-to-RP mapping information to all routers in the PIM domain using RP-discovery messages sent every 60 seconds to the group address 224.0.1.40. All PIM-enabled routers join 224.0.1.40 and store the group-to-RP mapping information in their private cache.

Multiple RP MAs can be configured in the same network to provide redundancy in case of a failure. There is no election mechanism between them, and they act independently of each other. They all advertise identical group-to-RP mapping information to all routers in the PIM domain.

As with MAs, multiple C-RPs can also be configured for load balancing, redundancy, or both:

- For redundancy, every C-RP should advertise the same group range. By doing this, the MA selects the C-RP with the highest IP address.
- For load balancing, every C-RP should service a different range of group addresses; for example, if there are two C-RPs and the multicast range is 232.0.0.0-239.255.255.255 (232.0.0.0/5), one could serve 232.0.0.0-235.255.255.255 (232.0.0.0/6) and the other 236.0.0.0-239.255.255.255 (236.0.0.0/6).
- For both load balancing and redundancy, if there are two C-RPs, one could serve the specific range 232.0.0.0-235.255.255.255 (232.0.0.0/6) and the multicast range 232.0.0.0-239.255.255.255 (232.0.0.0/5), and the other could serve the specific range 236.0.0.0-239.255.255.255 (236.0.0.0/6) and 232.0.0.0/5. This way, traffic is load balanced between the two based on the longest match selection criteria. If the C-RP servicing 232.0.0.0/6 fails, the other C-RP covering the whole 232.0.0.0/5 range would take over and if the C-RP servicing 236.0.0.0/6 fails, the other C-RP covering the whole 232.0.0.0/5 range would take over.

In [Figure 17-11](#), Auto-RP is used for the group-to-RP information to be distributed to all routers in the PIM domain. R2 and XR5 are configured as C-RPs and as MAs.

The C-RPs R2 and XR5 are advertising the default group range 224.0.0.0/4, and they use their loopbacks as the source for the messages, which are 192.168.2.2 and 192.168.5.5, respectively. [Example 17-14](#) shows the relevant configuration for this scenario.

### **Example 17-14** *Auto-RP with Multiple RPs Configuration*

[Click here to view code image](#)

```
R2
interface Loopback0
 ip address 192.168.2.2 255.255.255.255
 ip pim sparse-mode
!
ip pim autorp listener
ip pim send-rp-announce Loopback0 scope 10
ip pim send-rp-discovery Loopback0 scope 10
```

```
XR5
router pim
 address-family ipv4
  auto-rp mapping-agent Loopback0 scope 10 interval 60
  auto-rp candidate-rp Loopback0 scope 10 group-list 224-4 interval 60
```

Because the C-RP with the highest IP address is always elected as the RP, XR5 is elected as the RP for the domain. This can be seen in IOS with the command **show ip pim rp mapping** and in IOS XR with the command **show pim rp mapping**, as shown in [Example 17-15](#).

### Example 17-15 Auto-RP RP Mapping

[Click here to view code image](#)

```
R3#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 224.0.0.0/4
  RP 192.168.5.5 (?), v2
    Info source: 192.168.2.2 (?), elected via Auto-RP
    Uptime: 00:02:52, expires: 00:02:02
```

```
RP/0/0/CPU0:XR6#show pim rp mapping
PIM Group-to-RP Mappings
Group(s) 224.0.0.0/4
  RP 192.168.5.5 (?), v2
    Info source: 192.168.5.5 (?), elected via autorp
    Uptime: 00:00:04, expires: 00:02:57
```

### Auto-RP Group Filtering

As the number of groups in the PIM domain grows, dividing the groups between multiple RPs to balance the load between them will decrease the CPU and memory resources consumed by maintaining a large amount of multicast state.

Splitting group assignment to different RPs is possible with the optional **group-list** keyword in the IOS command **ip pim send-rp-announce** under global configuration mode and in IOS XR with the command **auto-rp candidate-rp**. The **group-list** keyword references an access list that includes the groups for which the router can be the RP.

[Example 17-6](#) shows the configuration necessary to map all groups from 232.0.0.0 to 235.255.255.255 (232.0.0.0/6) to R2 and groups 236.0.0.0 to 239.255.255.255 (236.0.0.0/6) to XR5.

### Example 17-16 Auto-RP Load-Balancing Configuration

[Click here to view code image](#)

```
R2
access-list 1 permit 232.0.0.0 3.255.255.255
!
ip pim send-rp-announce Loopback0 scope 10 group-list 1
```

**XR5**

```
ipv4 access-list 1
 10 permit ipv4 236.0.0.0 3.255.255.255 any
!
router pim
 address-family ipv4
  auto-rp candidate-rp Loopback0 scope 10 group-list 1 interval 60
```

**Example 17-17** shows the new group-to-RP mappings in the PIM domain. R2 was elected as the RP for group range 232.0.0.0/6, and XR5 was elected as the RP for group range 236.0.0.0/6.

**Example 17-17 Load Balancing Auto-RP Group-to-RP Mapping**

[Click here to view code image](#)

```
R3#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 232.0.0.0/6
  RP 192.168.2.2 (?), v2v1
    Info source: 192.168.5.5 (?), elected via Auto-RP
    Uptime: 00:00:08, expires: 00:02:51
Group(s) 236.0.0.0/6
  RP 192.168.5.5 (?), v2
    Info source: 192.168.5.5 (?), elected via Auto-RP
    Uptime: 00:00:08, expires: 00:02:47
```

```
RP/0/0/CPU0:XR6#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 232.0.0.0/6
  RP 192.168.2.2 (?), v2
    Info source: 192.168.5.5 (?), elected via autorp
    Uptime: 00:02:55, expires: 00:02:57
Group(s) 236.0.0.0/6
  RP 192.168.5.5 (?), v2
    Info source: 192.168.5.5 (?), elected via autorp
    Uptime: 00:02:02, expires: 00:02:57
```

**Example 17-18** demonstrates the configuration necessary to achieve both load balancing and redundancy. The configuration is the same as in **Example 17-16** but with the addition of a less-specific range 232.0.0.0/5 on both C-RPs.

**Example 17-18 Auto-RP Load-Balancing and Redundancy Configuration**

[Click here to view code image](#)

```
R2
access-list 1 permit 232.0.0.0 3.255.255.255
access-list 1 permit 232.0.0.0 7.255.255.255
!
ip pim send-rp-announce Loopback0 scope 10 group-list 1
```



**XR5**

```
ipv4 access-list 1
 10 permit ipv4 236.0.0.0 3.255.255.255 any
 20 permit ipv4 236.0.0.0 7.255.255.255 any
!
router pim
 address-family ipv4
   auto-rp candidate-rp Loopback0 scope 10 group-list 1 interval 60
```

**Example 17-19** shows the results after the MA advertises the new group-to-RP mappings to all the routers in the PIM domain. The group range 232.0.0.0/6 is advertised by R2, and group 236.0.0.0/6 is advertised by R2. Notice that for the less-specific range 232.0.0.0/5 XR5 was elected as the RP because it has a higher IP address than R2.

**Example 17-19 Load Balancing and Redundancy Auto-RP Group-to-RP Mapping**

[Click here to view code image](#)

```
R3#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 232.0.0.0/6
  RP 192.168.2.2 (?), v2v1
    Info source: 192.168.5.5 (?), elected via Auto-RP
    Uptime: 00:04:55, expires: 00:02:31
Group(s) 232.0.0.0/5
  RP 192.168.5.5 (?), v2
    Info source: 192.168.5.5 (?), elected via Auto-RP
    Uptime: 00:02:57, expires: 00:02:32
Group(s) 236.0.0.0/6
  RP 192.168.5.5 (?), v2
    Info source: 192.168.2.2 (?), elected via Auto-RP
    Uptime: 00:02:55, expires: 00:02:03
R3#
```

```
RP/0/0/CPU0:XR6#show ip pim rp mapping
PIM Group-to-RP Mappings
Group(s) 232.0.0.0/6
  RP 192.168.2.2 (?), v2
    Info source: 192.168.2.2 (?), elected via autorp
    Uptime: 00:04:50, expires: 00:02:54
Group(s) 236.0.0.0/6
  RP 192.168.5.5 (?), v2
    Info source: 192.168.2.2 (?), elected via autorp
    Uptime: 00:04:47, expires: 00:02:54
Group(s) 232.0.0.0/5
  RP 192.168.5.5 (?), v2
    Info source: 192.168.2.2 (?), elected via autorp
    Uptime: 00:04:49, expires: 00:02:54
RP/0/0/CPU0:XR6#
```

**Example 17-20** shows what the group-to-RP mappings would look like if R2 failed. If R2 fails, its

group range 232.0.0.0/6 would still be covered by the less-specific group range 232.0.0.0/5 serviced by XR5.

### Example 17-20 Auto-RP RP Mapping After R2 Failure

[Click here to view code image](#)

```
R3#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 232.0.0.0/5
  RP 192.168.5.5 (?), v2
    Info source: 192.168.5.5 (?), elected via Auto-RP
    Uptime: 00:04:43, expires: 00:02:35
Group(s) 236.0.0.0/6
  RP 192.168.5.5 (?), v2
    Info source: 192.168.5.5 (?), elected via Auto-RP
    Uptime: 00:00:21, expires: 00:02:36
```

```
RP/0/0/CPU0:XR6#show ip pim rp mapping
PIM Group-to-RP Mappings
Group(s) 236.0.0.0/6
  RP 192.168.5.5 (?), v2
    Info source: 192.168.5.5 (?), elected via autorp
    Uptime: 00:01:49, expires: 00:02:20
Group(s) 232.0.0.0/5
  RP 192.168.5.5 (?), v2
    Info source: 192.168.5.5 (?), elected via autorp
    Uptime: 00:06:55, expires: 00:02:20
RP/0/0/CPU0:XR6#
```

Example 17-21 shows what the group-to-RP mappings would look like if XR5 had failed instead of R2. If XR5 fails, its group range 236.0.0.0/6 would be covered by the less-specific group range 232.0.0.0/5 serviced by R2.

### Example 17-21 Auto-RP RP Mapping After XR5 Failure

[Click here to view code image](#)

```
R3#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 232.0.0.0/6
  RP 192.168.2.2 (?), v2v1
    Info source: 192.168.2.2 (?), elected via Auto-RP
    Uptime: 00:06:10, expires: 00:02:56
Group(s) 232.0.0.0/5
  RP 192.168.2.2 (?), v2v1
    Info source: 192.168.2.2 (?), elected via Auto-RP
    Uptime: 00:05:39, expires: 00:02:56
```

```

RP/0/0/CPU0:XR6#show pim rp mapping
PIM Group-to-RP Mappings
Group(s) 232.0.0.0/6
  RP 192.168.2.2 (?), v2
    Info source: 192.168.2.2 (?), elected via autorp
    Uptime: 00:07:11, expires: 00:01:54
Group(s) 232.0.0.0/5
  RP 192.168.2.2 (?), v2
    Info source: 192.168.2.2 (?), elected via autorp
    Uptime: 00:03:26, expires: 00:01:54

```

## BSR with Multiple RPs

PIM uses the BSR mechanism to discover and announce RP-set information for each group prefix to all the routers in a PIM domain just as an MA does in Auto-RP, but BSR is part of the PIMv2 specification.

To avoid a single point of failure, multiple candidate BSRs (C-BSRs) can be deployed in a PIM domain, where the C-BSR with the highest priority is elected as the BSR. If the C-BSR priorities are equal or if the C-BSR priority is not configured, the C-BSR with the highest IP address is elected as the BSR and starts sending BSR messages to all PIM routers in the PIM domain. In IOS, the BSR priority is 0 by default, and in IOS XR it is 1.

As with C-BSRs, multiple C-RPs can be deployed throughout the PIM domain for redundancy. These C-RPs receive the BSR messages, which include the IP address of the currently active BSR, and because they know the IP address of the BSR, they can unicast candidate-RP-advertisement messages directly to it, which carry a list of group address and group mask field pairs.

The BSR forms a group-to-RP mapping cache and then sends it in BSR messages every 60 seconds by default to all PIM routers in the entire network. Unlike the MA in Auto-RP, the BSR does not elect the RP itself; it leaves this task to each individual router in the network.

Each router in the network uses a well-known hashing algorithm to elect the currently active RP for a particular group range. Because each router is running the same algorithm against the same list of C-RPs, they will all select the same RP for a particular group range. The default hash mask length advertised by the BSR router in IOS XR is 32 and in IOS is 0.

C-RPs with a lower priority are preferred. If the priorities are the same, the C-RP with the highest IP address is elected as the RP for the particular group range. In IOS XR, the default C-RP priority is 0 and in IOS XR is 192.

In [Figure 17-11](#), the BSR is used for the group-to-RP information to be distributed to all routers in the PIM domain. R2 and XR5 are configured as C-BSR and as C-RPs with default priorities. The C-RPs R2 and XR5 are advertising the default group range 224.0.0.0/4, and they are using their loopbacks as the source of the messages, which are 192.168.2.2 and 192.168.5.5, respectively.

[Example 17-22](#) shows the BSR configuration for this scenario. The priority for the C-RP configuration on R2 is 0 by default, and it was set to 192 to match the default value of 192 of XR5. The hash mask length for the BSR configuration on XR5 is 30 by default, and it was set to 0 to

match the default value of R2.

### Example 17-22 BSR with Multiple RPs Configuration

[Click here to view code image](#)

```
R2
ip pim bsr-candidate Loopback0 0
ip pim rp-candidate Loopback0 priority 192
```

```
XR5
router pim
address-family ipv4
  bsr candidate-bsr 192.168.5.5 hash-mask-len 0 priority 1
  bsr candidate-rp 192.168.5.5 priority 192 interval 60
```

Example 17-23 shows XR5 was chosen as the BSR for the domain because its default priority is 1.

### Example 17-23 XR5 is the BSR for the PIM Domain

[Click here to view code image](#)

```
R3#show ip pim bsr-router
PIMv2 Bootstrap information
  BSR address: 192.168.5.5 (?)
  Uptime:      00:00:12, BSR Priority: 1, Hash mask length: 0
  Expires:     00:02:09
```

```
RP/0/0/CPU0:XR6#show pim bsr election
PIM BSR Election State

Cand/Elect-State      Uptime      BS-Timer      BSR              C-BSR
No-Info/Accept-Pref  00:00:07    00:02:03     192.168.5.5 [1, 0]  0.0.0.0 [0, 0]
```

In Auto-RP, the MA selects the RP for a group range and advertises it to all the routers in the PIM domain. In BSR, the BSR router advertises the RP-set information to all routers in the PIM domain and lets them elect the RP by running an RP hash algorithm. Example 17-24 shows how R3 and XR6 received information about all C-RPs in the PIM domain from the BSR router.

### Example 17-24 BSR RP Mapping Information on LHRs

[Click here to view code image](#)

```
R3#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 224.0.0.0/4
  RP 192.168.5.5 (?), v2
    Info source: 192.168.5.5 (?), via bootstrap, priority 192, holdtime 150
    Uptime: 00:06:50, expires: 00:02:08
  RP 192.168.2.2 (?), v2
    Info source: 192.168.5.5 (?), via bootstrap, priority 192, holdtime 150
    Uptime: 00:06:50, expires: 00:02:09
```

```
RP/0/0/CPU0:XR6#show ip pim rp mapping
PIM Group-to-RP Mappings
Group(s) 224.0.0.0/4
  RP 192.168.5.5 (?), v2
    Info source: 10.56.1.5 (?), elected via bsr, priority 192, holdtime 150
    Uptime: 00:08:15, expires: 00:02:10
Group(s) 224.0.0.0/4
  RP 192.168.2.2 (?), v2
    Info source: 10.56.1.5 (?), elected via bsr, priority 192, holdtime 150
    Uptime: 00:08:15, expires: 00:02:10
```

The hash value is based on G, the RP address, and the hash mask included in the bootstrap message. The RP with the highest hash value is elected as the RP for the group range, which in this case is XR5, as you can see in [Example 17-25](#).

### **Example 17-25** *RP Hash Shows XR5 Was Chosen as the RP for the PIM Domain*

[Click here to view code image](#)

```
R3#show ip pim rp-hash 239.1.1.1
RP 192.168.5.5 (?), v2
  Info source: 192.168.5.5 (?), via bootstrap, priority 192, holdtime 150
  Uptime: 00:14:31, expires: 00:02:05
PIMv2 Hash Value (mask 0.0.0.0)
  RP 192.168.5.5, via bootstrap, priority 192, hash value 809444037
  RP 192.168.2.2, via bootstrap, priority 192, hash value 462261592
```

```
R3#show ip pim rp-hash 239.1.1.5
RP 192.168.5.5 (?), v2
  Info source: 192.168.5.5 (?), via bootstrap, priority 192, holdtime 150
  Uptime: 00:14:34, expires: 00:02:02
PIMv2 Hash Value (mask 0.0.0.0)
  RP 192.168.5.5, via bootstrap, priority 192, hash value 809444037
  RP 192.168.2.2, via bootstrap, priority 192, hash value 462261592
```

[Example 17-26](#) shows that XR5 is chosen as the RP for all multicast groups.

### **Example 17-26** *RP Hash Shows XR5 Is Chosen as the RP for All Groups*

[Click here to view code image](#)

```
R2#show ip pim rp-hash 238.1.1.1
RP 192.168.5.5 (?), v2
  Info source: 192.168.5.5 (?), via bootstrap, priority 192, holdtime 150
  Uptime: 07:56:50, expires: 00:02:23
PIMv2 Hash Value (mask 0.0.0.0)
  RP 192.168.5.5, via bootstrap, priority 192, hash value 809444037
  RP 192.168.2.2, via bootstrap, priority 192, hash value 462261592
```

```
R2#show ip pim rp-hash 239.1.1.1
RP 192.168.5.5 (?), v2
  Info source: 192.168.5.5 (?), via bootstrap, priority 192, holdtime 150
  Uptime: 07:56:54, expires: 00:02:19
PIMv2 Hash Value (mask 0.0.0.0)
  RP 192.168.5.5, via bootstrap, priority 192, hash value 809444037
  RP 192.168.2.2, via bootstrap, priority 192, hash value 462261592
```

```
R2#show ip pim rp-hash 239.1.1.6
RP 192.168.5.5 (?), v2
  Info source: 192.168.5.5 (?), via bootstrap, priority 192, holdtime 150
  Uptime: 07:56:55, expires: 00:02:17
PIMv2 Hash Value (mask 0.0.0.0)
  RP 192.168.5.5, via bootstrap, priority 192, hash value 809444037
  RP 192.168.2.2, via bootstrap, priority 192, hash value 462261592
```

## BSR Group Filtering

BSR supports access lists to divide the groups between the RPs to distribute the consumption of router resources used for maintaining a large amount of multicast state.

Assigning specific groups to different RPs uses the optional **group-list** keyword in the IOS command **ip pim rp-candidate** and the IOS XR command **bsr candidate-rp**. The **group-list** keyword references an access list that includes the groups for which the router can be the RP.

Example 17-27 demonstrates the concept where all groups from 239.0.0.0 to 239.255.255.255 are directed toward R2 and groups 238.0.0.0 to 238.255.255.255 are directed to XR5.

### Example 17-27 BSR Group Filtering Configuration

[Click here to view code image](#)

```
R2
access-list 1 permit 239.0.0.0 0.255.255.255
!
ip pim rp-candidate Loopback0 group-list 1 priority 192
```

**XR5**

```
ipv4 access-list 1
 10 permit ipv4 238.0.0.0 0.255.255.255 any
!
router pim
 address-family ipv4
  bsr candidate-rp 192.168.5.5 group-list 1 priority 192 interval 60
```

**Example 17-28** displays the RP mappings based on the new BSR advertisements using the new configuration. Group range 238.0.0.0/8 is being advertised by XR5 and group 239.0.0.0/8 by R2.

**Example 17-28 RP Mapping After Configuring Group Filtering**

[Click here to view code image](#)

```
R3#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 238.0.0.0/8
  RP 192.168.5.5 (?), v2
    Info source: 192.168.5.5 (?), via bootstrap, priority 192, holdtime 150
    Uptime: 00:00:18, expires: 00:02:07
Group(s) 239.0.0.0/8
  RP 192.168.2.2 (?), v2
    Info source: 192.168.5.5 (?), via bootstrap, priority 192, holdtime 150
    Uptime: 00:36:56, expires: 00:02:10
```

```
RP/0/0/CPU0:XR6#show ip pim rp mapping
PIM Group-to-RP Mappings
Group(s) 238.0.0.0/8
  RP 192.168.5.5 (?), v2
    Info source: 10.56.1.5 (?), elected via bsr, priority 192, holdtime 150
    Uptime: 00:00:02, expires: 00:02:27
Group(s) 239.0.0.0/8
  RP 192.168.2.2 (?), v2
    Info source: 10.56.1.5 (?), elected via bsr, priority 192, holdtime 150
    Uptime: 00:42:47, expires: 00:02:27
RP/0/0/CPU0:XR6#
```

Using the RP hashing algorithm is a better way to distribute the load on the RPs, and this is described in the next section.

**BSR RP Hash Algorithm**

In Auto-RP, if two C-RPs advertise the same group range, only one C-RP is chosen as the RP for that range. In BSR, by using the hash mask, the two C-RPs can split the load between them in half.

The hash mask can be specified in IOS with the command **ip pim bsr-candidate** and in IOS XR with the command **bsr candidate-bsr**. As mentioned earlier in this section, the default hash mask length advertised by an IOS XR BSR router is 32, and 0 for an IOS router, so setting this to a different value is completely optional.

The BSR advertises the hash mask along with the C-RP and group addresses in its bootstrap messages, and the receiving routers run the hash algorithm using these variables. This algorithm then assigns a consecutive number of multicast group addresses to one C-RP and then another group to another C-RP and so on.

For example, in [Figure 17-11](#), R2 and XR5 are the C-RPs, the hash mask is set to 30 (default value in IOS XR), and the group range is 239.0.0.0 to 239.255.255.255:

- The last 2 bits of the hash mask are used to indicate the number of consecutive groups that will be assigned to an RP, which in this case would be four (2 bits = 4 group addresses).
- This translates to R2 being assigned group range 239.0.0.0, 239.0.0.1, 239.0.0.2, 239.0.0.3.
- XR5 is then assigned the group range 239.0.0.4, 239.0.0.5, 239.0.0.6, 239.0.0.7.
- This continues for the entire multicast range, and in theory, both RPs should end up with an even number of group assignments.

The hash mask allows you to specify how many consecutive addresses are bundled into a single range. If the hash mask is set to 28, there are 4 bits (32 – Hash Mask Length) used to indicate the number of consecutive groups; this means that 16 consecutive group addresses are assigned to each RP.

[Example 17-29](#) demonstrates the BSR hash configuration on R2 and XR5.

### **Example 17-29** BSR Hash Mask Configuration

[Click here to view code image](#)

```
R2
access-list 1 permit 239.0.0.0 0.255.255.255
ip pim bsr-candidate Loopback0 30
ip pim rp-candidate Loopback0 group-list 1 priority 192
```

```
XR5
ipv4 access-list 1
 10 permit ipv4 239.0.0.0 0.255.255.255 any
!
router pim
 address-family ipv4
  bsr candidate-bsr 192.168.5.5 hash-mask-len 30 priority 1
  bsr candidate-rp 192.168.5.5 group-list 1 priority 192 interval 60
```

### **Static RP with Multiple RPs**

By default, only one static RP can be configured on a router because there can be only one RP active for a group at any one time. The default static RP configuration serves the whole 224.0.0.0/4 multicast range; however, using group lists, just as with Auto-RP and BSR, allows for dividing the multicast groups among multiple static RPs. This will help decrease the router resources that can be consumed by maintaining a large amount of multicast state.



You can assign different groups to different RPs in IOS with the command **ip pim rp-address ip-address [group-list acl-name]** in global configuration mode and in IOS XR with the command **rp-address ip-address [acl-name]** under PIM configuration mode. The **group-list** keyword in IOS or the **acl-name** argument in IOS XR reference an access list that includes the groups for which the router can be the RP.

Example 17-30 shows how to map all groups from 239.0.0.0 to 239.255.255.255 to R2 and groups 238.0.0.0 to 232.255.255.255 to XR5.

### **Example 17-30** *Static RP with Multiple RPs and Group Filtering Configuration*

[Click here to view code image](#)

#### **R2**

```
access-list 1 permit 238.0.0.0 0.255.255.255
access-list 2 permit 239.0.0.0 0.255.255.255
ip pim rp-address 192.168.2.2 2
ip pim rp-address 192.168.5.5 1
```

#### **XR5**

```
ipv4 access-list 1
 10 permit ipv4 238.0.0.0 0.255.255.255 any
!
ipv4 access-list 2
 10 permit ipv4 239.0.0.0 0.255.255.255 any
!
router pim
 address-family ipv4
  rp-address 192.168.2.2 2
  rp-address 192.168.5.5 1
```

### **Anycast RP**

A much better way to achieve RP load sharing and redundancy is to use the anycast RP mechanism, described in RFC 3446. With anycast RP, all RPs within the PIM-SM domain are assigned the same IP address, also known as the *anycast RP address*, which is a /32 host route configured on a loopback interface. All routers within the PIM-SM domain learn of the anycast RP address through Auto-RP, BSR, or static RP assignment.

Each designated router (DR) in one portion of the PIM-SM domain will send PIM register and join messages to the closest anycast RP address based on unicast routing metrics. If a particular anycast RP router fails, unicast routing will direct subsequent PIM register and join messages to the nearest anycast RP. Assume that in Figure 17-11, R2 and XR5 add a Loopback 1 interface with the same IP address 192.168.25.25. The Loopback 1 interfaces will be used as anycast RPs, and they advertise the same anycast RP address 192.168.25.25 into the PIM domain. From R3's perspective, R2 is the nearest RP, so it will send PIM join messages to it. From XR6's perspective, XR5 is the nearest RP, so it will send PIM join messages to it. If R2 fails, XR5 will be the nearest RP for R3, so it will send PIM messages to it as long as R2 continues to be down.

On the source side, XR4 is by default the DR router because it has the highest IP address. When the source becomes active, XR4 will start sending PIM registers to XR5, which is the nearest RP.

The problem here is that R2 will not receive these PIM registers and will be completely unaware that the source is active. The solution is for R2 and XR5 to MSDP peer with each other; this way, when XR5 receives the register messages, it will send SAs to R2 to let it know there is an active source in the network.

When MSDP is configured within a PIM domain, it is recommended to use MSDP mesh groups. MSDP mesh groups are useful when two or more peers are in a group where all MSDP speakers peer with each other (fully meshed). SA flooding is reduced when MSDP mesh groups are configured because when an MSDP peer in the group receives an SA message from another MSDP peer in the group, it assumes that this SA message was sent to all the other MSDP peers in the group. As a result, it is not necessary for the receiving MSDP peer to flood the SA message to the other MSDP peers in the group.

MSDP mesh groups also eliminate the need to run (M)BGP to do RPF checks on arriving SA messages. This is because SA messages are never flooded to other MSDP peers in the mesh group. As a result, it is not necessary to perform the RPF check on arriving SA messages. For this reason, it is recommended to use MSDP mesh groups only within a PIM domain, or intradomain, and not on interdomain deployments because the absence of RPF checks could create SA loops.

The IOS command to configure MSDP Mesh-Groups is **ip msdp mesh-group** *mesh-name* {*peer-address* | *peer-name*} under global configuration mode and the IOS XR command **peer** *peer-address* **mesh-group** *mesh-name* under MSDP configuration mode.

MSDP uses the RP address as the source address of its SA messages, but the SA source address must be unique (just as the router ID [RID] needs to be unique for OSPF). To ensure the source of the SA messages is unique, an additional loopback needs to be configured to be used as the anycast RP address and should be advertised into the domain.

**Example 17-31** demonstrates the anycast configuration for R2 and XR5, which are the RPs for the domain. The RP address is configured statically in all routers in the domain, but BSR and Auto-RP could be used. The Loopback 1 interface is created specifically to be used as the anycast RP address (192.168.25.25/32). The reason for this is that because MSDP uses the RP address as the source of its SA messages, it will try to use 192.168.25.25/32, which still does not solve the problem of SA messages having a unique source address. To force MSDP to use a different source address, the IOS command **ip msdp originator-id** *interface-type interface-number* under global configuration mode or the IOS XR command **originator-id** *interface-type interface-number* under MSDP configuration mode should be used. Because Lo1 was created to be used as the anycast RP, Lo0 can be used as MSDP's originator ID.

Note

Because duplicate loopback addresses exist in the network for the deployment of anycast RP, problems may occur in routing protocols due to duplicate RIDs. Most routing protocols require the RID to be unique. For this reason, it is strongly recommended to explicitly configure a unique RID for all routing protocols.

### **Example 17-31** Anycast RP Configuration

[Click here to view code image](#)

```
R2
ip multicast-routing
!
interface Loopback0
 ip address 192.168.2.2 255.255.255.255
!
interface Loopback1
 ip address 192.168.25.25 255.255.255.255
 ip pim sparse-mode
!
router ospf 1
 router-id 192.168.2.2
 network 192.168.25.25 0.0.0.0 area 0
!
ip pim rp-address 192.168.25.25
!
ip msdp peer 192.168.5.5 connect-source Loopback0
ip msdp cache-sa-state
ip msdp originator-id Loopback0
ip msdp mesh-group ANYCAST 192.168.5.5
```

```
XR5
interface Loopback0
 ipv4 address 192.168.5.5 255.255.255.255
!
interface Loopback1
 ipv4 address 192.168.25.25 255.255.255.255
!
router ospf 1
 router-id 192.168.5.5
 area 0
 interface Loopback0
 !
 interface Loopback1
 !
!
multicast-routing
 address-family ipv4
 interface all enable
!
!
router msdp
 originator-id Loopback0
 peer 192.168.2.2
 connect-source Loopback0
 mesh-group ANYCAST
!
!
router pim
 address-family ipv4
 rp-address 192.168.25.25
```

[Example 17-32](#) verifies that the MSDP session between R2 and XR5 is up.

## Example 17-32 MSDP Session Information Between R2 and XR5

[Click here to view code image](#)

```
R2#show ip msdp summary
MSDP Peer Status Summary
Peer Address      AS      State    Uptime/  Reset SA    Peer Name
                  AS      State    Downtime Count Count
192.168.5.5       ?      Up       00:08:43 1      0      ?
```

```
RP/0/0/CPU0:XR5#show msdp summary
Out of Resource Handling Enabled
Maximum External SA's Global : 20000
Current External Active SAs : 0
```

```
MSDP Peer Status Summary
Peer Address      AS      State    Uptime/  Reset Peer  Active Cfg.Max  TLV
                  AS      State    Downtime Count Name   SA Cnt Ext.SAs recv/sent
192.168.2.2       0      Up       00:10:55 1      ?      0      0      10/22
```

## SOURCE SPECIFIC MULTICAST

In traditional PIM-SM/DM (Dense Mode) networks, receivers use Internet Group Management Protocol Version 2 (IGMPv2) joins to signal that they would like to receive multicast traffic from a specific multicast group. The IGMPv2 joins include only the multicast group (G) the receiver wants to join but it does not specify the source (S) for the multicast traffic. Because the source is unknown to the receiver, the receiver can accept traffic from any source transmitting to the group. This type of multicast service model is known as *any source multicast* (ASM).

One of the problems with ASM is that the possibility exists that a receiver has the potential of receiving multicast traffic from different sources transmitting to the same group. Even though the application on the receivers typically can filter out the unwanted traffic, network bandwidth and resources are wasted.

RFC 4607 defines PIM Source Specific Multicast (SSM) as one of the operating modes of PIM. The Internet Assigned Number Authority (IANA) assigned the 232.0.0.0/8 multicast range to SSM, so routers support SSM by default on this range. SSM is allowed to use any other multicast group in the 224.0.0.0/4 multicast range so long as it is not reserved (like the Local Network Control Block range 224.0.0.0 – 224.0.0.255 that was defined early in [Chapter 16's "Multicast Addressing"](#) section). SSM operates with IGMPv3, and requires IGMPv3 support on the multicast routers, the receiver where the application is running, and the application itself.

IGMPv3 membership reports (joins) are sent to the multicast group address 224.0.0.22 and allow a receiver to specify the source and the group it would like to receive multicast traffic from. Because the IGMPv3 join includes the (S,G), referred to as a *channel* in SSM, the DR builds a source tree (shortest path tree [SPT]) by sending an (S,G) PIM join directly to the source. Because SSM is SPT based, RPs are no longer required, and there is no need to implement dynamic RP discovery mechanisms such as Auto-RP and BSR. SSM supports interdomain routing, and because it does not require the use of RPs, it does not need MSDP to support it.

Typical multicast deployments that benefit from PIM-SSM consist of entertainment-type

solutions offered by service providers such as IPTV over GPON, ETTH, cable, and xDSL. In many multicast deployments where the source is known, PIM-SSM is the obvious multicast routing protocol of choice to use because of its simplicity.

IGMPv3 source discovery is normally achieved via some type of directory services, Session Description Protocol (SDP), file transfer (FTP, SFTP, HTTP, XML, and so on), or they can also be manually configured on the multicast device or application.

Figure 17-12 illustrates a cable customer with a set-top box (STB) requesting an STB management server to provide it with a list of the sources and channels that it is entitled to receive based on its serial number.

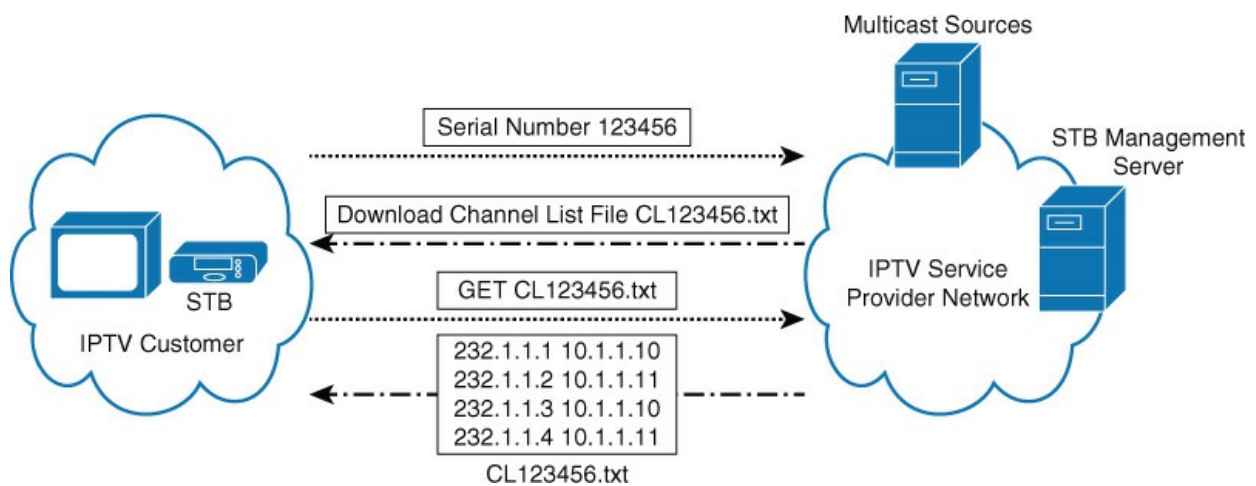


Figure 17-12 STB Requesting File with Available Sources

Figure 17-13 demonstrates how an MDT is created in SSM. Receiver 1 sends an IGMPv3 join for source S 10.13.2.10 and multicast group G 232.1.1.1. R2 then sends an (S,G) PIM join directly to its RPF neighbor R1. At this point, the MDT is built, and multicast traffic starts flowing down the source tree to the receiver.

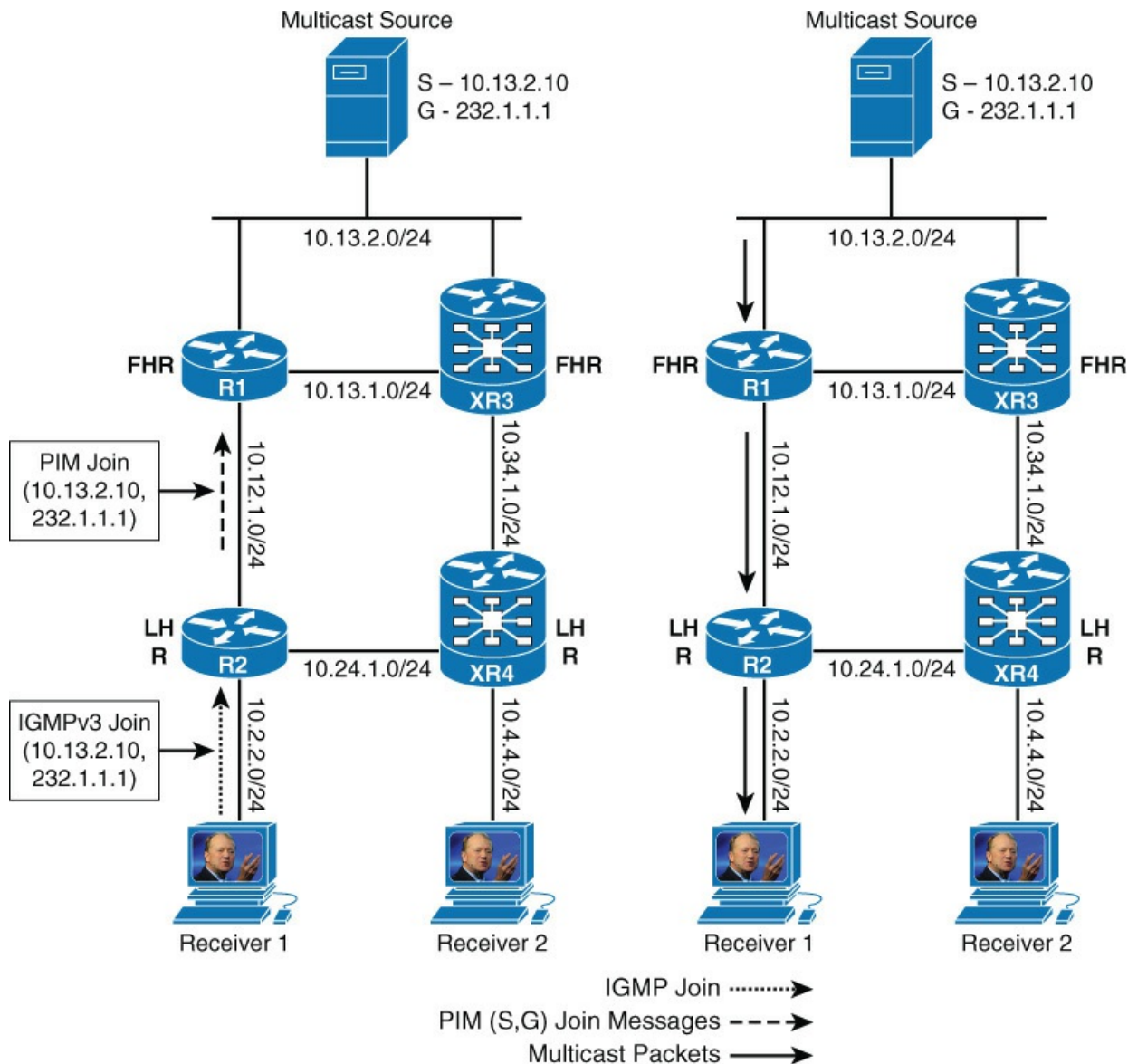


Figure 17-13 SSM Topology MDT Creation

In IOS, apart from enabling multicast routing and PIM on relevant interfaces, the following steps are also required to enable SSM:

**Step 1.** Enable IGMPv3 on receiver facing interfaces (required). IGMPv3 is enabled with the interface parameter command **ip igmp version 3**.

**Step 2.** Enable SSM (required). SSM is enabled with the command **ip pim ssm {default | range access-list}** under global configuration mode. The **default** keyword enables SSM for the default 232.0.0.0/8 range. The **range** keyword enables SSM for other ranges in the 224.0.0.0/4 multicast range.

In IOS XR, enabling multicast routing is all that is required to enable IGMPv3 and SSM for the default 232.0.0.0/8 range. Use the command **ssm range access-list** to enable SSM for other ranges in the 224.0.0.0/4 multicast range.

Example 17-33 demonstrates a sample SSM configuration for the default 232.0.0.0/8 SSM multicast range on IOS and IOS XR.

### Example 17-33 SSM Configuration for Default 232.0.0.0/8 Range

[Click here to view code image](#)

```
IOS
ip multicast-routing
!
ip pim ssm default
!
interface GigabitEthernet0/4
ip pim sparse-mode
ip igmp version 3
```

```
IOS XR
multicast-routing
address-family ipv4
interface all enable
```

Example 17-34 demonstrates the SSM configuration using the default 232.0.0.0/8 range and the additional 239.0.0.0/8 range.

### Example 17-34 SSM Configuration Using the range Option

[Click here to view code image](#)

```
IOS
ip access-list standard SSM-Range
permit 232.0.0.0 0.255.255.255
permit 239.0.0.0 0.255.255.255
!
ip pim ssm range SSM-Range
```

```
IOS XR
ipv4 access-list SSM-Range
10 permit ipv4 232.0.0.0 0.255.255.255 any
20 permit ipv4 239.0.0.0 0.255.255.255 any
!
multicast-routing
address-family ipv4
ssm range SSM-Range
interface all enable
```

IOS XR has the capability to display the group-to-PIM mode mapping with the command **show pim group-map**. Example 17-35 demonstrates the group-to-PIM mode mapping on XR4 after applying the configuration in Example 17-34.

### Example 17-35 PIM Group Mapping Showing 232.0.0.0/8 and 239.0.0.0/8 in SSM Range

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show pim group-map
```

```
IP PIM Group Mapping Table
```

```
(* indicates group mappings being used)
```

```
(+ indicates BSR group mappings active in MRIB)
```

Group Range	Proto	Client	Groups	RP address	Info
224.0.1.39/32*	DM	perm	0	0.0.0.0	
224.0.1.40/32*	DM	perm	1	0.0.0.0	
224.0.0.0/24*	NO	perm	0	0.0.0.0	
232.0.0.0/8*	SSM	config	1	0.0.0.0	
239.0.0.0/8*	SSM	config	0	0.0.0.0	
224.0.0.0/4*	SM	static	0	0.0.0.0	RPF: Null,0.0.0.0

Figure 17-14 illustrates a topology with SSM enabled on R1, R2, XR3, and XR4. The source is transmitting to multicast groups 232.1.1.1 and 239.1.1.1. Receiver 1 has sent an IGMPv3 join to R2 for group 232.1.1.1, and Receiver 2 has sent an IGMPv3 join for group 239.1.1.1. The configurations used for all routers in the topology are those shown in Example 17-34.



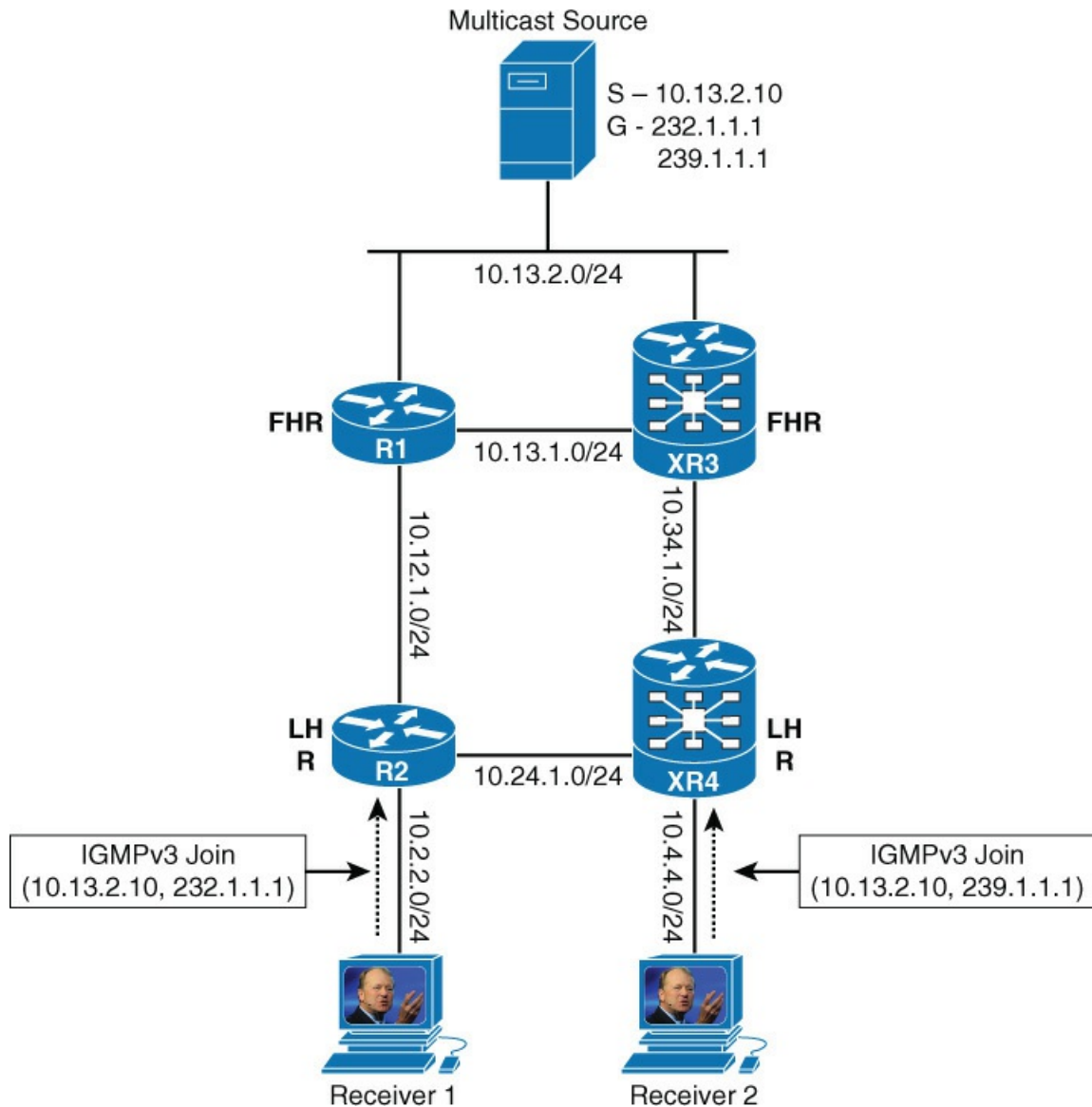


Figure 17-14 SSM Topology

R2 and XR4 will do a reverse path forwarding (RPF) lookup to identify the RPF neighbors to send them (S,G) PIM joins for the groups they received IGMP joins for. The RPF neighbors can be viewed by executing the commands shown in Example 17-36.

### Example 17-36 RPF Neighbor for Multicast Source

[Click here to view code image](#)

```
R2#show ip rpf 10.13.2.10
RPF information for ? (10.13.2.10)
  RPF interface: GigabitEthernet0/1
  RPF neighbor: ? (10.12.1.1)
  RPF route/mask: 10.13.2.0/24
  RPF type: unicast (ospf 1)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

```
RP/0/0/CPU0:XR4#show pim rpf 10.13.2.10
```

```
Table: IPv4-Unicast-default
```

```
* 10.13.2.10/32 [110/2]
```

```
via GigabitEthernet0/0/0/2 with rpf neighbor 10.34.1.3
```

**Example 17-37** shows the mroute table on R1 and R2. The multicast flags TI in R2 are indicating 232.1.1.1 is an SSM Group (s) that this is an SPT (T) and that it received an IGMPv3 join (I). The multicast flags sT in R1 are indicating 232.1.1.1 is an SSM group(s) that this is an SPT (T).

**Example 17-37** *Mroute table on R1 and R2*

[Click here to view code image](#)

```

R2#show ip mroute 232.1.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
      L - Local, P - Pruned, R - RP-bit set, F - Register flag,
      T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
      X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
      U - URD, I - Received Source Specific Host Report,
      Z - Multicast Tunnel, z - MDT-data group sender,
      Y - Joined MDT-data group, y - Sending to MDT-data group,
      G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
      N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
      Q - Received BGP S-A Route, q - Sent BGP S-A Route,
      V - RD & Vector, v - Vector, p - PIM Joins on route,
      x - VxLAN group

Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(10.13.2.10, 232.1.1.1), 10:10:53/00:02:29, flags: sTI
  Incoming interface: GigabitEthernet0/1, RPF nbr 10.12.1.1
  Outgoing interface list:
    GigabitEthernet0/4, Forward/Sparse, 00:17:35/00:02:29

```

```

R1#show ip mroute 232.1.1.1
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
      L - Local, P - Pruned, R - RP-bit set, F - Register flag,
      T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
      X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
      U - URD, I - Received Source Specific Host Report,
      Z - Multicast Tunnel, z - MDT-data group sender,
      Y - Joined MDT-data group, y - Sending to MDT-data group,
      G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
      N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
      Q - Received BGP S-A Route, q - Sent BGP S-A Route,
      V - RD & Vector, v - Vector, p - PIM Joins on route,
      x - VxLAN group

Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM Join
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(10.13.2.10, 232.1.1.1), 10:16:53/00:02:49, flags: sT
  Incoming interface: GigabitEthernet0/3, RPF nbr 0.0.0.0
  Outgoing interface list:
    GigabitEthernet0/2, Forward/Sparse, 00:17:52/00:02:49

```

**Example 17-38** displays the PIM topology tables on XR3 and XR4. The flags LI LH on XR4 indicate there is a receiver directly connected interested on group 239.1.1.1 (LI) and that XR4 is the LHR (LH). SPT SSM is indicating this is an SSM source tree (SPT)

### **Example 17-38** Mroute Table on XR3 and XR4

[Click here to view code image](#)

```

RP/0/0/CPU0:XR4#show pim topology 239.1.1.1
IP PIM Multicast Topology Table
Entry state: (*S,G) [RPT/SPT] Protocol Uptime Info
Entry flags: KAT - Keep Alive Timer, AA - Assume Alive, PA - Probe Alive
             RA - Really Alive, IA - Inherit Alive, LH - Last Hop
             DSS - Don't Signal Sources, RR - Register Received
             SR - Sending Registers, SNR - Sending Null Registers
             E - MSDP External, EX - Extranet
             MFA - Mofrr Active, MFP - Mofrr Primary, MFB - Mofrr Backup
             DCC - Don't Check Connected, ME - MDT Encap, MD - MDT Decap
             MT - Crossed Data MDT threshold, MA - Data MDT Assigned
             SAJ - BGP Source Active Joined, SAR - BGP Source Active Received,
             SAS - BGP Source Active Sent, IM - Inband mLDP
Interface state: Name, Uptime, Fwd, Info
Interface flags: LI - Local Interest, LD - Local Dissinterest,
                II - Internal Interest, ID - Internal Dissinterest,
                LH - Last Hop, AS - Assert, AB - Admin Boundary, EX - Extranet,
                BGP - BGP C-Multicast Join, BP - BGP Source Active Prune,
                MVS - MVPN Safi Learned, MV6S - MVPN IPv6 Safi Learned

(10.13.2.10,239.1.1.1)SPT SSM Up: 00:30:32
JP: Join(00:00:17) RPF: GigabitEthernet0/0/0/2,10.34.1.3 Flags:
    GigabitEthernet0/0/0/1    00:30:32 fwd LI LH
RP/0/0/CPU0:XR4#

```

```

RP/0/0/CPU0:XR3#show pim topology 239.1.1.1
! Output omitted for brevity

(10.13.2.10,239.1.1.1)SPT SSM Up: 00:00:32
JP: Join(00:00:17) RPF: GigabitEthernet0/0/0/2,10.13.2.10* Flags:
    GigabitEthernet0/0/0/1    00:00:32 fwd Join(00:02:57)
RP/0/0/CPU0:XR3#

```

## SSM Mapping

There are scenarios in which an organization might want to migrate to SSM but they have many legacy devices (that is, STBs) and applications that support only IGMPv2 and don't have the capability to support IGMPv3. For cases like this, SSM mapping could be deployed.

SSM mapping introduces a way for the LHR to discover sources. When SSM mapping is configured, if a router receives an IGMPv2 join for a group G, the router translates this report into one or more source IP addresses for group G as if it had received an IGMPv3 join. The router then sends out PIM joins toward (S1,G) to (Sn,G) and continues to be joined to these groups as long as it continues to receive IGMPv2 joins and as long as the SSM mapping for the group remains the same.

### Note

SSM mapping needs only to be configured on the last-hop router connected to receivers. It is not necessary on any other routers in the network.

SSM mapping enables LHRs to determine the source addresses either by a statically configured table on the router or by consulting a DNS server. When the statically configured table is

changed, or when the DNS mapping changes, the router leaves the current sources associated with the joined groups.

### DNS SSM Mapping

DNS SSM mapping enables the LHR to perform a forward DNS lookup to determine sources sending to groups. When DNS SSM mapping is configured and the router receives an IGMPv2 join for group G, the router creates a domain name that includes the group address G and performs a DNS forward lookup. The router uses the returned IP address(s) in the DNS response as the source address(s) associated with this group. The router then sends PIM (S,G) joins for all sources. SSM mapping supports up to 20 sources for each group.

To look up one or more source addresses for a group G that includes G1, G2, G3, and G4, the DNS resource records (RRs) shown in [Table 17-2](#) must be configured on the DNS server.

---

Name	Timeout	Class	RR Type	Source Address
<i>[multicast group][multicast-domain]</i>	<i>[timeout]</i>	IN	A	<i>[source-address]</i>

---

Table 17-2 DNS Resource Records Format

The *multicast-domain* argument is a configurable DNS prefix, the default is in-addr. arpa but can be changed to match your domain's prefix. The *timeout* argument indicates how long the router will keep the current mapping in its cache before querying the DNS server for this group again. The timeout can be configured for each group/source entry on the DNS server. You can configure this time for larger values if you want to minimize the number of DNS queries generated by the router. Configure this time for a low value if you want to be able to quickly update all routers with new source addresses.

In IOS, you enable DNS SSM mapping as follows:

#### Step 1. Enable SSM mapping (required).

SSM mapping is enabled with the global command **ip igmp ssm-map enable**.

#### Step 2. Enable DNS-based SSM mapping (optional).

SSM mapping is enabled with the global command **ip igmp ssm-map query dns**.

This command is enabled by default after enabling SSM mapping, as shown in Step 1. Use this command to reenable DNS-based SSM mapping if DNS-based SSM mapping is disabled.

#### Step 3. Change the default domain prefix (optional).

The domain prefix can be changed with the command **ip domain multicast domain-prefix** (for example, ssm-map.cisco.com). The default domain prefix is in-addr.arpa.

#### Step 4. Specify the address of one or more DNS servers (required).

DNS servers can be specified with the command **ip name-server server-address1 [server-address2...server-address6]**.

---

Note

IOS XR support only Static SSM mapping, as covered in the next section

Figure 17-15 illustrates Receiver 1 sending an IGMPv2 join to R2 (the LHR) for group 232.1.1.1; R2 then performs a forward DNS lookup on group 232.1.1.1.ssm-map.cisco.com. R2 receives a DNS response that includes the IP address 10.13.2.10, which is used as the source for 232.1.1.1. R2 sends a PIM join (10.13.2.10, 232.1.1.1) to its RPF neighbor to build an SPT.

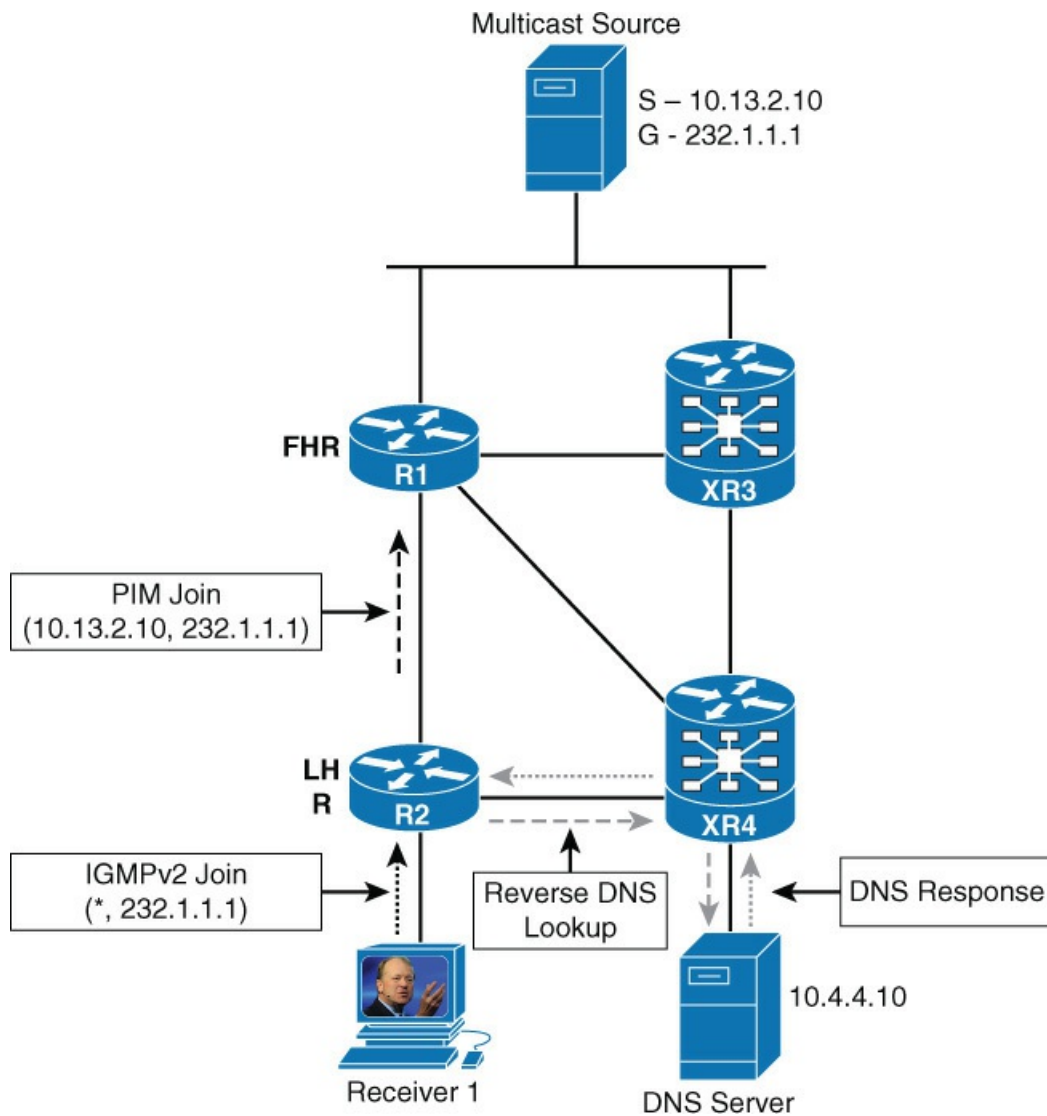


Figure 17-15 DNS SSM Topology

Example 17-39 shows the DNS-based configuration that would be required on R2.

### Example 17-39 DNS-Based SSM Mapping Configuration

[Click here to view code image](#)

```
R2
ip domain multicast ssm-map.cisco.com
ip name-server 10.4.4.10
ip igmp ssm-map enable
```

In IOS, you can use the **show ip igmp ssm-mapping** command to validate the configuration, as shown in [Example 17-40](#).

### Example 17-40 DNS SSM Mapping Verification

[Click here to view code image](#)

```
R2#show ip igmp ssm-mapping
SSM Mapping    : Enabled
DNS Lookup     : Enabled
Mcast domain   : ssm-map.cisco.com
Name servers   : 10.4.4.10
```

The forward lookup sent by the router to the DNS server would be `232.1.1.1.ssm-map.cisco.com`, which is a combination of the multicast group and the multicast domain. For this reason, when configuring a multicast domain on the router, it should also be configured on the DNS server.

For `232.1.1.1.ssm.map.cisco.com` to resolve to `10.13.2.10`, the DNS record format would look like this:

```
232.1.1.1.ssm-map.cisco.com 3600 IN A 10.13.2.10
```

The default timeout is 86400, and it was set to 3600 seconds.

If the same source were also transmitting other multicast groups, the DNS RR would look like this:

```
232.1.1.1.ssm-map.cisco.com 3600 IN A 10.13.2.10
```

```
232.1.1.2.ssm-map.cisco.com 3600 IN A 10.13.2.10
```

```
239.1.1.1.ssm-map.cisco.com 3600 IN A 10.13.2.10
```

#### Static SSM Mapping

Static SSM mapping can be deployed in smaller networks when DNS is not needed or to override DNS mappings that may be temporarily incorrect. Static SSM mappings take precedence over DNS mappings.

SSM static mapping enables the LHR to use a static map to determine the sources sending to groups. Static SSM mappings require an access control list (ACL) to define multicast group ranges. The groups permitted by the ACLs can then be mapped to sources using the IOS command **ip igmp ssm-map static *acl-name*** under global configuration mode and the IOS XR command **ssm map static source-address *acl-name*** under IGMP configuration mode.

#### Note

In IOS, if DNS mapping is not required, it should be disabled on the router with the command **no ip igmp ssm-map query dns**; otherwise, when the router receives a join for a group that is not statically configured, it will send a DNS forward lookup to determine the source for the group. The router will then ignore all other joins until the DNS times out.

[Example 17-41](#) demonstrates SSM static mapping configuration for IOS and IOS XR.

### Example 17-41 SSM Static Mapping Configuration

[Click here to view code image](#)

#### IOS

```
ip access-list standard Channel-1
  permit 232.1.1.1
ip access-list standard Channel-2
  permit 239.1.1.1
ip igmp ssm-map enable
ip igmp ssm-map static Channel-1 10.13.2.10
ip igmp ssm-map static Channel-2 10.13.2.11
```

#### IOS XR

```
ipv4 access-list Channel-1
  10 permit ipv4 host 232.1.1.1 any
!
ipv4 access-list Channel-2
  10 permit ipv4 host 239.1.1.1 any
!
router igmp
  ssm map static 10.13.2.10 Channel-1
  ssm map static 10.13.2.11 Channel-2
```

SSM static mapping configuration is verified with the IOS command **show ip igmp ssm-mapping group-address** and the IOS XR command **show igmp ssm-mapping group-address**, as shown in [Example 17-42](#).

### Example 17-42 SSM Mapping Verification

[Click here to view code image](#)

```
R2#show ip igmp ssm-mapping
SSM Mapping : Enabled
DNS Lookup  : Enabled
Mcast domain : ssm-map.cisco.com
Name servers : 10.4.4.10

R2#show ip igmp ssm-mapping 232.1.1.1
Group address: 232.1.1.1
Database      : Static
Source list   : 10.13.2.10

R2#show ip igmp ssm-mapping 239.1.1.1
Group address: 239.1.1.1
Database      : Static
Source list   : 10.13.2.11
R2#
```



#### IOS XR

```
RP/0/0/CPU0:XR4#show igmp ssm map
```

```
IGMP SSM Mapping Information
```

```
232.1.1.1 is static with 1 source
```

```
239.1.1.1 is static with 1 source
```

```
RP/0/0/CPU0:XR4#show igmp ssm map 232.1.1.1
```

```
IGMP SSM Mapping Information
```

```
232.1.1.1 is static with 1 source
```

```
RP/0/0/CPU0:XR4#show igmp ssm map 239.1.1.1
```

```
IGMP SSM Mapping Information
```

```
239.1.1.1 is static with 1 source
```

## MULTICAST SECURITY

This section covers different techniques you can apply in a multicast network to make it secure. It includes security measures to protect PIM domains against attacks. Before taking specific measures to secure a network against specific multicast attack forms, basic security must be deployed. This applies to routers, switches, and any other elements in the network.

### Auto-RP Scoping

A router controls the distance that the Auto-RP's Cisco-RP-announce and Cisco-RP-discovery messages are advertised by modifying a packet's Time-To-Live (TTL) value. Because the TTL value decrements between hops, this reduces the diameter of the Auto-RP messages as a method of identifying the PIM domain.

A TTL value can be configured on IOS and IOS XR by using the **scope** keyword, as shown in [Example 17-43](#). The TTL **scope** keyword has no predefined default value, and it is required to configure it.

### Example 17-43 Auto-RP TTL Scope Configuration Example

[Click here to view code image](#)

#### R2

```
ip pim send-rp-announce Loopback0 scope 2
ip pim send-rp-discovery Loopback0 scope 2
```

#### XR5

```
router pim
 address-family ipv4
  auto-rp mapping-agent Loopback0 scope 2 interval 60
  auto-rp candidate-rp Loopback0 scope 2 group-list 224-4 interval 60
```

The TTL scope selected for these messages should be set to a value that will ensure that they will reach all routers in the PIM domain; otherwise, serious problems could arise.

[Figure 17-16](#) demonstrates the how the scope can control the size of the multicast routing domain. In the topology, two C-RPs are configured with a scope of two hops. Because the MA is three hops away from C-RP 2, it will not receive its Cisco-RP-discovery messages, so it will have incomplete information. The MA will advertise only RP information for C-RP1 to the rest of the

domain.

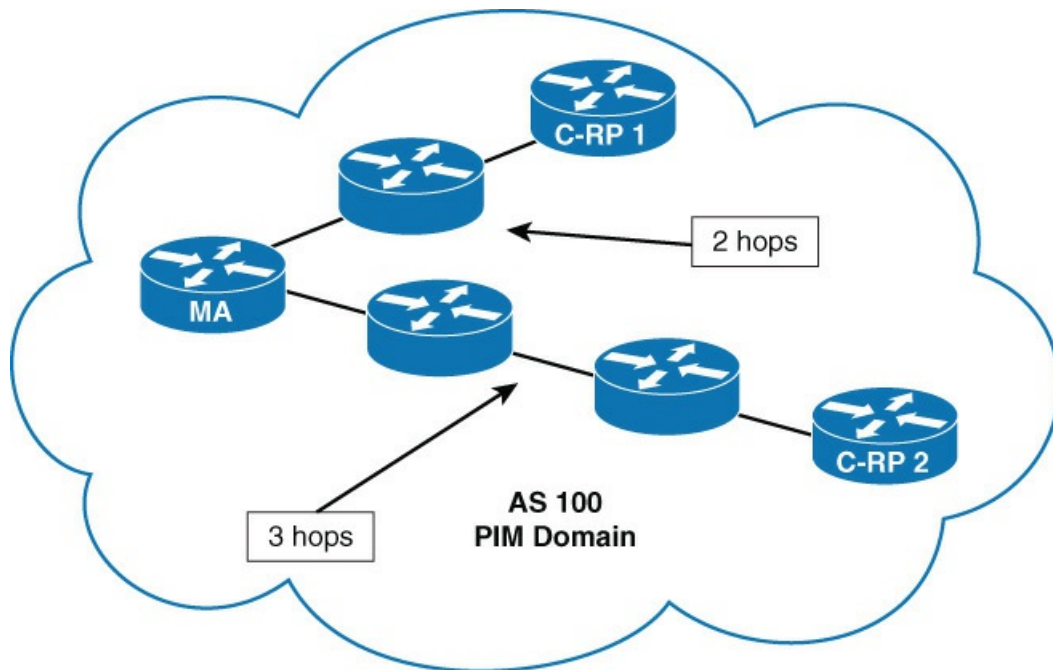


Figure 17-16 TTL Scope Topology

A recommendation to avoid this situation is to configure the TTL scope to the maximum number of hops in the network. This way, Cisco-RP-announce and Cisco-RP-discovery messages will reach all routers. The problem with this approach is that some of these messages could leak out of the PIM domain. This can be controlled by using multicast boundaries, as covered in the next section.

### Multicast Boundaries

Establishing multicast boundaries provides a better way to constrain multicast traffic that flows into or out of the PIM domain.

### Administratively Scoped Boundaries

Multicast control and data traffic can be constrained from leaking outside of the PIM domain or leaking into the PIM domain by deploying the multicast boundary feature. You can use the multicast boundary feature to filter data and control plane traffic, including IGMP, PIM join and prune, and Auto-RP messages. The PIM messages mentioned here are not filtered by IP multicast boundaries:

- PIM register messages (because they are sent using unicast)
- PIM BSR messages
- PIM hellos sent to 224.0.0.13

#### Note

The upcoming sections "PIM-SM Source Registration Filtering," "BSR Multicast Boundaries," and "PIM Neighbor Control," cover features that can be used to block the PIM messages in the preceding list.

The multicast boundary feature can be configured with the IOS interface parameter configuration command **ip multicast boundary access-list [filter-autorp | in | out]** and the IOS-XR

command **boundary** *acl-name* under multicast routing interface configuration mode.

In IOS, the *access-list* argument supports standard and extended ACLs:

- Standard ACLs can be used to define the group address range (\*,G) to be permitted or denied on an interface.
- Extended access lists can be used to
- Define (S,G) traffic to be permitted or denied on an interface by using the ACL arguments *<src-ip>* *<src-wildcard>* *<group-address>* *<group-mask>*.
- Define the group address range (\*,G) to be permitted or denied on an interface by specifying host 0.0.0.0 for the source address in the ACL entry.

In IOS, the **in**, **out**, and **filter-autorp** keywords are used as follows:

- The **in** keyword is used to filter multicast source traffic coming into the interface. This keyword is applicable to incoming interface (IIF) only.
- The **out** keyword is used to prevent multicast state from being created on an interface; that is, it will prevent IGMP reports and PIM joins from creating multicast state for groups and channels denied by the boundary ACL, and the interface will not be included in the outgoing interface list (OIL). This keyword is applicable to outgoing interfaces (OIFs) only.
- If a direction is not specified, the command filters multicast source traffic coming into the interface and prevents multicast state from being created on the interface.
- If the **filter-autorp** keyword is configured, Auto-RP Cisco-RP-announce and Cisco-RP-discovery messages are examined, and any Auto-RP group range announcements from the Auto-RP packets that are denied by the boundary ACL are removed. An Auto-RP group range announcement is permitted and passed by the **ip multicast boundary** command only if all addresses in the Auto-RP group range are permitted by the ACL. If any address is not permitted, the whole group range is filtered and removed from the Auto-RP message before the Auto-RP message is forwarded.

Note

Extended ACLs cannot be used with the **filter-autorp** keyword because Cisco-RP-announce messages do not contain source addresses.

### Auto-RP Multicast Boundaries

Preventing Auto-RP messages from leaking into or outside of a PIM domain helps prevent outages that could occur if multicast boundaries were not deployed.

For example, an attacker from outside of the PIM domain could advertise Cisco-RP-announce messages that would reach an internal MA. The MA could select the malicious C-RP messages and advertise the bogus information to all routers causing an outage. Now consider the attacker sending RP-discovery messages with bogus information into the network, these messages will

cause all the routers to switch back and forth between the valid MA group-to-RP information and the bogus MA group-to-RP information causing serious problems in the network.

**Example 17-44** demonstrates how the multicast boundary feature can be used to block Auto-RP. The ACL blocks 224.0.1.39 and 224.0.1.40, which are the multicast groups for Cisco-RP-announce and Cisco-RP-discovery messages, respectively. It also blocks administratively scoped multicast traffic in the 239.0.0.0/8 range and allows all other multicast traffic.

#### **Example 17-44** *Auto-RP Multicast Boundary Configuration*

[Click here to view code image](#)

```
IOS XR
ipv4 access-list 1
 10 deny ipv4 host 224.0.1.39 any
 20 deny ipv4 host 224.0.1.40 any
 30 deny ipv4 239.0.0.0 0.255.255.255 any
 40 permit ipv4 224.0.0.0 15.255.255.255 any
!
multicast-routing
 address-family ipv4
  interface GigabitEthernet0/0/0/2
   boundary 1
  !
 interface all enable
```

```
IOS
access-list 1 deny 224.0.1.39
access-list 1 deny 224.0.1.40
access-list 1 deny 239.0.0.0 0.255.255.255
access-list 1 permit 224.0.0.0 15.255.255.255

interface GigabitEthernet0/0
 ip multicast boundary 1
```

**Example 17-45** demonstrates the use of the **filter-autorp** keyword. Any Cisco-RP-announce or Cisco-RP-discovery messages that contain or overlap with the denied group range 239.0.0.0/8 will be filtered. For example, if the message contains the range 224.0.0.0/4, because it overlaps with the denied group range, it will be filtered. If the range advertised is 238.0.0.0/8, because it does not overlap with the denied group range, it will be allowed.

#### **Example 17-45** *Auto-RP Message Filtering Configuration*

[Click here to view code image](#)

```
IOS
access-list 1 deny 239.0.0.0 0.255.255.255
access-list 1 permit any

interface GigabitEthernet0/0
 ip multicast boundary 1 filter-autorp
```

## BSR Multicast Boundaries

Just as with Auto-RP, allowing BSR messages to leak into or out of a network can cause problems in the network. To block BSR messages from entering or exiting a given interface, the IOS interface parameter configuration command **ip pim bsr-border** or the IOS XR PIM interface configuration command **bsr-border**, as shown in [Example 17-46](#). BSR filtering is usually applied on all edge interfaces that are adjacent to other PIM domains.

### Example 17-46 PIM BSR Border Configuration

[Click here to view code image](#)

```
IOS
interface GigabitEthernet0/0
 ip pim bsr-border
```

```
IOS XR
multicast-routing
 address-family ipv4
   interface GigabitEthernet0/0/0/2
     bsr-border
   !
 interface all enable
```

#### Note

The IOS and IOS XR **bsr-border** commands block only BSR messages. They do not block any multicast traffic, including PIM joins/prunes and so on. To block multicast traffic and BSR messages, use the multicast boundary feature in combination with the **bsr-border** command.

## Auto-RP Cisco-RP-Announce Message Filtering

In Auto-RP, MAs can be configured to only accept C-RP announcements from well-known routers in the network. This will prevent C-RP announcements from bogus routers or attackers from being accepted and potentially being selected as the RP.

To configure Cisco-RP-announce filtering, use the IOS command **ip pim rp-announce-filter {group-list acl-number | rp-list acl-number}**.

#### Note

Cisco-RP-announce message filtering is not supported on IOS XR.

The **rp-list** keyword specifies the IP addresses of the C-RPs from which to permit or deny RP announcements messages. The **group-list** keyword specifies the multicast groups included in RP announcement messages to be permitted or denied. RP announcement messages received that match the access list specified for **rp-list** keyword and the access list matched by the **group-list** keyword are filtered by the RP mapping agent. If the **rp-list** keyword is not configured, the command will permit all C-RPs. If the **group-list** keyword is not configured, the command will deny all groups.

[Example 17-47](#) demonstrates how to configure the MA to accept RP announcements from the C-RPs defined in access list 1 for the group range defined in access list 2. If the MA receives an RP-announcement that is not from one of the RPs specified, the announcement will be denied. If the

MA receives an RP-announcement from one of the specified RPs containing a multicast group that is not part of the 239.0.0.0/8 range, the RP-announcement will be denied.

### Example 17-47 Auto-RP Cisco-RP-Announce Message Filtering Configuration

[Click here to view code image](#)

```
ip pim rp-announce-filter rp-list 1 group-list 2
access-list 1 permit 192.168.2.2
access-list 1 permit 192.168.5.5
access-list 2 permit 239.0.0.0 0.255.255.255
```

#### Note

The **ip pim rp-announce-filter** should be configured only on MAs. The same filters must be configured on all MAs in the PIM domain to avoid inconsistencies in Auto-RP operations.

### PIM-SM Source Registration Filtering

As a security measure, source registration can be restricted only to authorized sources. This will, in most cases, prevent traffic from unauthorized sources from traversing the FHR and reaching other hosts in the network.

Source registration filtering is enabled using the IOS command **ip pim accept-register {list access-list-name/access-list-number | route-map map-name}** under global configuration mode and with the IOS XR command **accept-register access-list-name** under PIM configuration mode.

[Example 17-48](#) shows how to configure source registration filtering. Only sources in the 10.14.2.0/24 subnets are allowed to send register messages to the RP and only for multicast groups in the 239.0.0.0/8 range.

### Example 17-48 PIM-SM Source Registration Filtering

[Click here to view code image](#)

```
IOS
ip access-list extended AUTHORIZED_SOURCES
  permit ip 10.14.2.0 0.0.0.255 239.0.0.0 0.255.255.255
!
ip pim accept-register list AUTHORIZED_SOURCES
```

```
IOS XR
ipv4 access-list AUTHORIZED_SOURCES
  10 permit ipv4 10.14.2.0 0.0.0.255 239.0.0.0 0.255.255.255
!
router pim
  address-family ipv4
    rp-address 192.168.25.25
    accept-register AUTHORIZED_SOURCES
```

#### Note

Source registration filtering should only be configured on RPs. If the RP is also the FHRDR for directly connected sources, PIM register packets will not be filtered using the IOS and IOS XR **accept-register** command. For this case, use the multicast boundary feature to filter the directly connected source traffic.

### PIM-SM Accept RP

This is a security feature used to prevent unauthorized RPs or groups from becoming active in the PIM-SM domain. For this feature to be effective, it must be configured on every router on the path from the LHRs to the RP. An alternative is to configure it only on the RP, but this would allow unauthorized PIM joins to reach the RP where they would be dropped.

This feature is only supported in IOS and is configured with the command **ip pim accept-rp** {*rp-address* | **auto-rp**} [*access-list-name/access-list-number*].

Routers configured with **pim accept-rp** will only accept (\*,G) PIM join messages toward the RP address specified in the command. If the *access-list* argument is specified, the router will only accept PIM join messages for groups matching the access list.

If this command is configured on the RP and the group address in the PIM join or register message does not match the groups specified in the access list, the RP will not accept the messages and will respond immediately to register messages with register-stop messages.

**Example 17-49** demonstrates a configuration that will only allow multicast PIM join or register messages for group 239.0.0.0 destined to RPs 192.168.2.2 and 192.168.5.5.

### Example 17-49 PIM-SM Accept RP Configuration

[Click here to view code image](#)

```
access-list 1 permit 239.0.0.0
!
ip pim accept-rp 192.168.5.5 1
ip pim accept-rp 192.168.2.2 1
```

#### Note

This command does not affect (S,G) joins that take place during an SPT switchover

### PIM Neighbor Control

Unwanted PIM neighbor relationships can be prevented with the IOS interface parameter command **ip pim neighbor-filter** *access-list* or with the IOS XR command **neighbor-filter** *access-list* under PIM configuration mode or PIM interface configuration mode.

#### Note

PIM neighbor control does not protect against an attacker that sends spoofed PIM packets to pretend to be a valid PIM neighbor. In these cases, Layer 2 security mechanisms such as IP Source Guard may be used to prevent source address spoofing.

**Example 17-50** shows a sample configuration of how to configure PIM neighbor control in IOS and IOS XR.

## Example 17-50 PIM Neighbor Control Configuration

[Click here to view code image](#)

```
IOS
ip access-list standard PIM_Filter
  deny 10.13.2.3
  permit any
!
interface GigabitEthernet0/3
  ip address 10.13.2.1 255.255.255.0
  ip pim neighbor-filter PIM_Filter
```

```
IOS XR
ipv4 access-list PIM-Filter
  10 deny ipv4 host 10.13.2.1 any
  20 permit ipv4 any any
!
router pim
  address-family ipv4
    interface GigabitEthernet0/0/0/2
      neighbor-filter PIM-Filter
  !
!
!
```

### PIM Register Rate Limit

In IOS, the FHR DR can be configured to rate limit register messages to protect the RP against a potential flood of PIM-SM register messages. This can be done based on either packets per second or bits per second depending on the release. The command used to achieve this in IOS is **ip pim register-rate-limit *bits-per-second* | *packets-per-second***.

#### Note

Cisco IOS releases earlier than 12.2(33)SRE, 15.0(1)M and IOS XE Release 2.1 only support the *packets-per-second* argument. Newer releases only support the *bits-per-second* argument.

Setting the rate-limit value using the *packets-per-second* argument sets rate limiting on all PIM-SM registers, and setting the rate-limit value using the *bits-per-second* argument sets rate limiting on PIM-SM registers on a per-RP basis.

**Example 17-51** demonstrates how to configure the register message rate limiting on a DR with a maximum rate of 4 register packets per second or 8000 bits per second.

### Example 17-51 PIM Register Rate-Limit Configuration

[Click here to view code image](#)



#### IOS

```
! Packets per second  
ip pim register-rate-limit 2
```

```
! Bits per second  
ip pim register-rate-limit 8000
```

## MULTICAST TRAFFIC ENGINEERING

Unicast and multicast traffic may follow separate paths in many different cases. When this occurs, the unicast and multicast topologies are said to be incongruent. Some of these cases are as follows:

- When multicast is not enabled on all interfaces of a router
- When generic routing encapsulation (GRE) tunnels bypass sections of a network designated only for unicast traffic
- When a policy dictates that multicast traffic should follow different paths
- When multicast traffic is required to flow over low-utilization or backup links for better load balancing
- When the unicast and multicast networks are incongruent, certain limitations come into play, such as:
  - When RPF calculation cannot use the unicast routing table because that would mean the unicast and multicast networks are congruent
  - When some other source of RPF information other than the unicast routing table must be used

### RPF Rules

PIM-SM uses the RPF lookup function primarily for the following:

- **Control plane:** To determine the RPF interface and the RPF neighbor, it needs to send PIM joins to.
- **Data plane:** To prevent multicast traffic loops by performing RPF checks on multicast packets to make sure that they come from an RPF neighbor.

The RPF information is determined by checking the routes found in the unicast routing table, which is populated by static routing or IGP, but there can be other RPF sources, as shown in the following list, that can populate the multicast RIB that can be used to determine RPF information. When there are multiple RPF sources for the same prefix, the longest match rule (prefix length) is used to pick the best route. If all prefix lengths are the same, the lowest administrative distance (AD) is preferred. If AD values are equal, the RPF sources are preferred in the following order:

1. Static multicast routes (mroutes)
2. Multiprotocol BGP (MBGP)

### 3. Unicast routes

#### Note

Some IOS devices might need to be configured with the hidden command **ip multicast longest-match** for the longest match to be preferred over the AD. This command should be used only after thorough testing or from the advice of Cisco technical support.

#### Static Mroutes

Static mroutes affect only the path multicast traffic takes; they do not affect the unicast data path. Static mroutes are configured in IOS with the command **ip mroute** *source-address mask [protocol] {rpf-address | interface-type interface-number} [distance]*.

In IOS XR, static mroutes are configured as follows:

#### Step 1. Initialize the static routing process.

The static routing process is initialized with the command **router static**.

#### Step 2. Identify the address family.

Initialize the appropriate address family with the command **address-family {ipv4 | ipv6} multicast address-family**. IPv4 static mroutes will use the **ipv4** multicast option.

#### Step 3. Configure the static route.

Directly attached static routes use the syntax **network {subnet-mask | /prefix-length} interface-type interface-number rpf-address**.

#### Note

Static mroutes are locally significant and cannot be redistributed.

Figure 17-17 demonstrates the usage of static mroutes in a topology running OSPF and SSM:

- All links are the same cost, so the (S,G) PIM join from R2 should by default go directly to R1 because it is the shortest path to the source, and the SPT would be formed between R1 and R2.
- Static mroutes are used for traffic to change the RPF path so that traffic coming from source 10.13.2.10 takes the R1 --> R2 data path and traffic from source 10.13.2.20 to takes the data path XR3 --> XR4 --> R2.

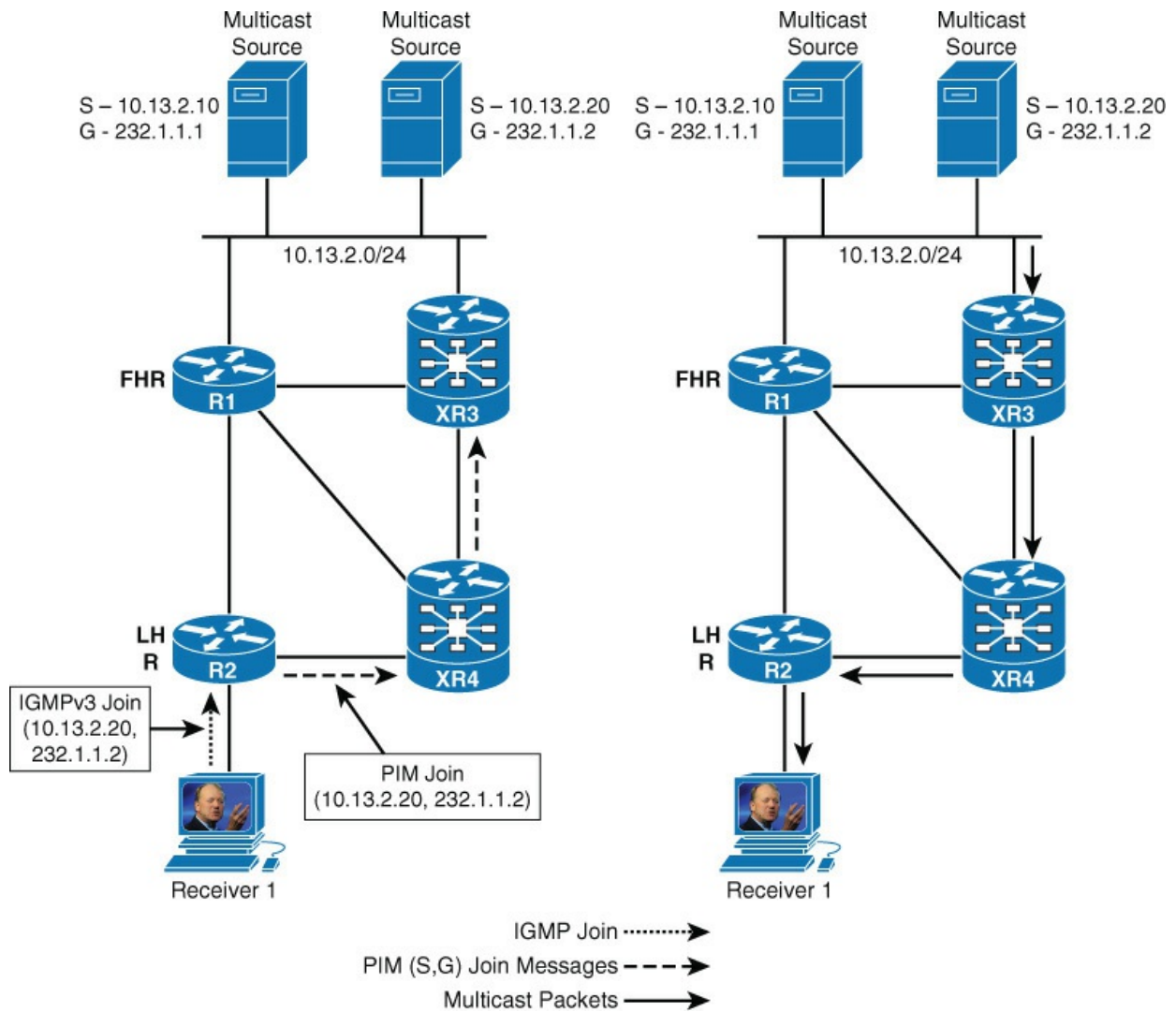


Figure 17-17 Multicast Traffic Engineering Topology

With static mroutes, the RPF interfaces/neighbors can be manipulated, and the PIM joins will follow the path dictated by them. PIM joins are used to build the multicast tree and are used to determine how the multicast data path is constructed.

Example 17-52 demonstrates the static mroute configurations required for the data path from source 10.13.2.20 to take a different path without manipulating the unicast routing table. The static mroute on R2 specifies the RPF for the source 10.13.2.20/32 should be XR4 10.24.1.4. The static mroute on R2 specifies the RPF for the source 10.13.2.20/32 should be XR3 10.34.1.3, and the RPF interface should be Go/o/o/2. There is no need to configure anything on XR3 because the source is directly connected.

### Example 17-52 Static Mroute Configuration

[Click here to view code image](#)

```
R2
ip mroute 10.13.2.20 255.255.255.255 10.24.1.4
```

**XR4**

```
router static
address-family ipv4 multicast
10.13.2.20/32 GigabitEthernet0/0/0/2 10.34.1.3
```

**Example 17-53** shows how the RPF neighbor for 10.13.2.10 is R1 (10.12.1.1) and the RPF neighbor for 10.13.2.20 is XR4 (10.24.1.4) and that it is statically configured with a static mroute.

**Example 17-53 R2 Mroute Table**

[Click here to view code image](#)

```
R2#show ip mroute 232.1.1.1
! Output omitted for brevity

(10.13.2.10, 232.1.1.1), 00:00:16/00:02:55, flags: sTI
Incoming interface: GigabitEthernet0/1, RPF nbr 10.12.1.1
Outgoing interface list:
GigabitEthernet0/4, Forward/Sparse, 00:00:16/00:02:55
```

```
R2#show ip mroute 232.1.1.2
! Output omitted for brevity

(10.13.2.20, 232.1.1.2), 00:00:18/00:02:48, flags: sTI
Incoming interface: GigabitEthernet0/2, RPF nbr 10.24.1.4, Mroute
Outgoing interface list:
GigabitEthernet0/4, Forward/Sparse, 00:00:18/00:02:48
```

**Example 17-54** shows XR4 has an entry for 232.1.1.2 and the RPF neighbor is XR3 (10.34.1.3). This is the direction in which the PIM joins will be sent for this group and where the MDT will be built.

**Example 17-54 XR4 Mroute Table**

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show mrib route 232.1.1.2
! Output omitted for brevity

(10.13.2.20,232.1.1.2) RPF nbr: 10.34.1.3 Flags:
Up: 00:03:04
Incoming Interface List
GigabitEthernet0/0/0/2 Flags: A, Up: 00:03:04
Outgoing Interface List
GigabitEthernet0/0/0/0 Flags: F NS, Up: 00:03:04
```

The source is directly connected to XR3 so the RPF neighbor for 10.13.2.20 is the source 10.13.2.20, as shown in [Example 17-55](#).

**Example 17-55 XR3 Mroute Table**

[Click here to view code image](#)

```
RP/0/0/CPU0:XR3#show mrib route 232.1.1.2
! Output omitted for brevity

(10.13.2.20,232.1.1.2) RPF nbr: 10.13.2.20 Flags:
Up: 00:03:33
Incoming Interface List
  GigabitEthernet0/0/0/2 Flags: A, Up: 00:03:33
Outgoing Interface List
  GigabitEthernet0/0/0/1 Flags: F NS, Up: 00:03:33
```

Example 17-56 shows the data path for 10.13.2.10 is taking the R1 --> R2 data path.

### Example 17-56 R1 Mroute Table

[Click here to view code image](#)

```
R1#show ip mroute 232.1.1.1
! Output omitted for brevity

(10.13.2.10, 232.1.1.1), 00:03:38/00:02:51, flags: sT
Incoming interface: GigabitEthernet0/3, RPF nbr 0.0.0.0
Outgoing interface list:
  GigabitEthernet0/2, Forward/Sparse, 00:03:38/00:02:51
```

```
R1#show ip mroute 232.1.1.2
Group 232.1.1.2 not found
```

If there were a requirement for all multicast traffic to follow the XR3 --> XR4 --> R2 path without manipulation of the unicast routing table, a static default mroute could be used as shown in Example 17-57. On R2, the RPF neighbor for all sources will be XR4 (10.24.1.4), so all PIM joins will be directed to it; and on XR4, the RPF neighbor would be XR3 (10.34.1.3), so all PIM joins would be directed to it.

### Example 17-57 Default Mroute Configuration

[Click here to view code image](#)

```
R2
ip mroute 0.0.0.0 0.0.0.0 10.24.1.4
```

```
XR4
router static
address-family ipv4 multicast
  0.0.0.0/0 GigabitEthernet0/0/0/2 10.34.1.3
```

## MBGP

As explained earlier in this chapter, MBGP is used for RPF calculations and it can be used for intradomain deployments as well. Using [Figure 17-17](#) as a reference topology, if there were a requirement for traffic coming from source 10.13.2.10 to take the R1 --> R2 data path and source 10.13.2.20 to take the data path XR3 --> XR4 --> R2, this could be achieved with MBGP, as demonstrated in [Example 17-58](#).

### Example 17-58 MBGP for Configuration

[Click here to view code image](#)

#### R2

```
router bgp 100
  no bgp default ipv4-unicast
  neighbor 192.168.4.4 remote-as 100
  neighbor 192.168.4.4 update-source Loopback0
  !
address-family ipv4 multicast
  neighbor 192.168.4.4 activate
exit-address-family
```

#### XR3

```
router bgp 100
  address-family ipv4 multicast
  network 10.13.2.20/32
  !
  neighbor 192.168.4.4
  remote-as 100
  update-source Loopback0
  address-family ipv4 multicast
```

#### XR4

```
router static
  address-family ipv4 unicast
  10.13.2.20/32 Null0
  !
router bgp 100
  address-family ipv4 multicast
  !
  neighbor 192.168.2.2
  remote-as 100
  update-source Loopback0
  address-family ipv4 multicast
  route-reflector-client
  !
  !
  neighbor 192.168.3.3
  remote-as 100
  update-source Loopback0
  address-family ipv4 multicast
  route-reflector-client
```

[Example 17-59](#) shows the RPF information for 10.13.2.10 and 10.13.2.20. R2 is using OSPF for the RPF calculations for source 10.13.2.10 and has chosen RPF neighbor 10.12.1.1. It is using

MBGP for the RPF calculations for source 10.13.2.20 and has chosen RPF neighbor 10.12.1.1

### Example 17-59 RPF Information for 10.13.2.10 and 10.13.2.20

[Click here to view code image](#)

```
R2#show ip rpf 10.13.2.10
RPF information for ? (10.13.2.10)
RPF interface: GigabitEthernet0/1
RPF neighbor: ? (10.12.1.1)
RPF route/mask: 10.13.2.0/24
RPF type: unicast (ospf 1)
Doing distance-preferred lookups across tables
RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

```
R2#show ip rpf 10.13.2.20
RPF information for ? (10.13.2.20)
RPF interface: GigabitEthernet0/2
RPF neighbor: ? (10.24.1.4)
RPF route/mask: 10.13.2.20/32
RPF type: multicast (bgp 100)
Doing distance-preferred lookups across tables
RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

Example 17-60 shows the next hop for prefix 10.13.2.20/32 is XR3 (192.168.3.3).

### Example 17-60 R2's MBGP Table

[Click here to view code image](#)

```
R2#show ip bgp ipv4 multicast
! Output omitted for brevity
BGP table version is 7, local router ID is 192.168.2.2

   Network          Next Hop          Metric LocPrf Weight Path
*>i 10.13.2.20/32    192.168.3.3          0   100     0 I
```

If the prefix 10.13.2.20/24 were advertised on XR4, instead of the static route 10.13.2.20/32, OSPF and BGP would both have the same prefix with the same mask in their tables. The AD would be the tiebreaker to decide which RPF source is preferred, and OSPF would win because it has an AD of 110 versus iBGP, which has an AD of 200 for iBGP. Example 17-61 shows how OSPF was preferred over BGP for the RPF calculations.

### Example 17-61 OSPF Preferred over BGP for RPF Calculations

[Click here to view code image](#)

```

R2#show ip rpf 10.13.2.10
RPF information for ? (10.13.2.10)
  RPF interface: GigabitEthernet0/1
  RPF neighbor: ? (10.12.1.1)
  RPF route/mask: 10.13.2.0/24
  RPF type: unicast (ospf 1)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base

```

```

R2#show ip rpf 10.13.2.20
RPF information for ? (10.13.2.20)
  RPF interface: GigabitEthernet0/1
  RPF neighbor: ? (10.12.1.1)
  RPF route/mask: 10.13.2.0/24
  RPF type: unicast (ospf 1)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base, originated from ipv4 unicast base

```

```

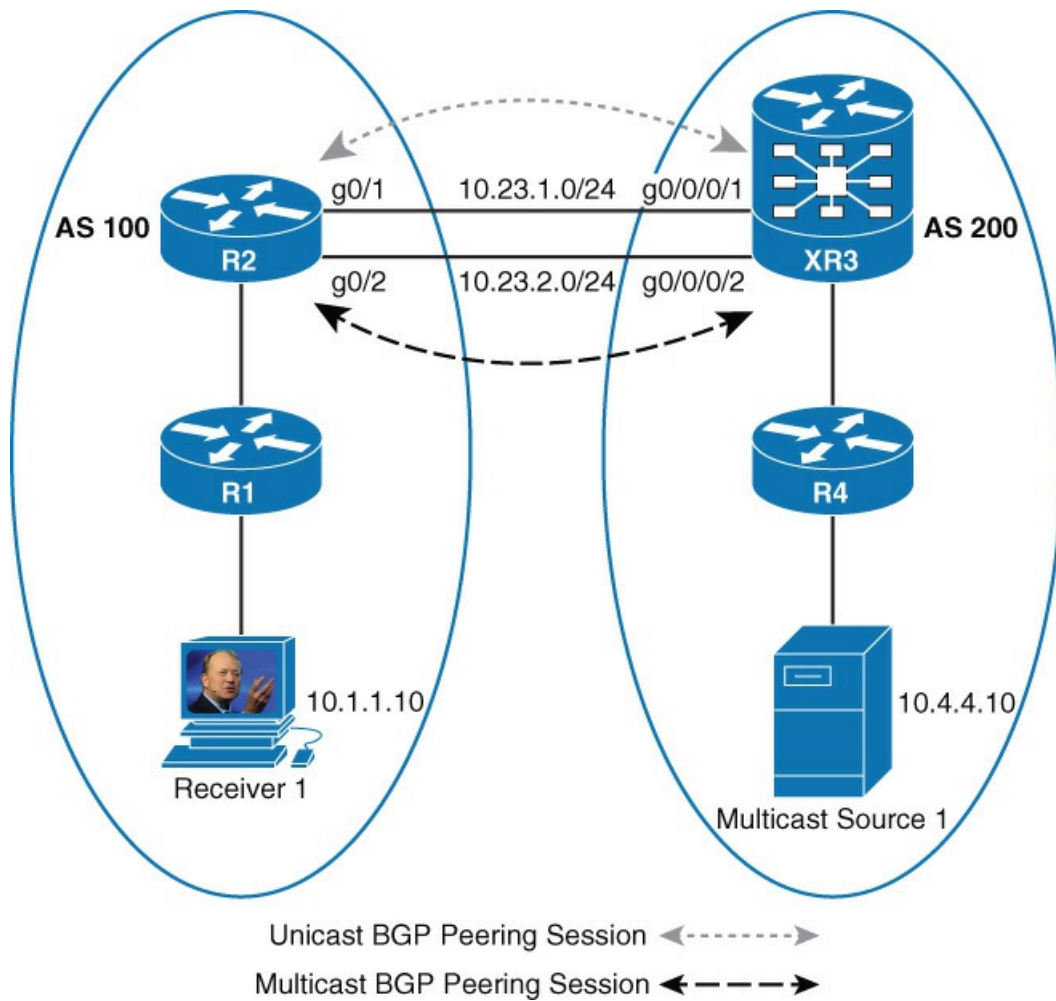
R2#show ip bgp ipv4 multicast
! Output omitted for brevity
BGP table version is 9, local router ID is 192.168.2.2

      Network          Next Hop          Metric LocPrf Weight Path
-----
r>i 10.13.2.0/24      192.168.3.3          0    100    0  i

```

**Figure 17-18** illustrates a topology used to demonstrate how MBGP can be used to create dissimilar unicast and multicast topologies. SSM is enabled for the default SSM range (232.0.0.0/8) in AS100 and AS200, and PIM-SM is configured on all interfaces including the interdomain links. There is a unicast BGP session between R2 and XR3 over the 10.23.1.0/24 link, from now on referred to as the *unicast link*, and a multicast BGP session over the 10.23.2.0/24 link, from now on referred to as the *multicast link*. On XR3, the source subnet 10.4.4.0/24 is being advertised into unicast BGP and into multicast BGP simultaneously. Unicast traffic should use the unicast link between R2 and XR3, and multicast traffic should use the multicast link between R2 and XR3. In case the multicast link fails, the unicast link should carry both unicast and multicast traffic.





**Figure 17-18** *Topology with Incongruent Unicast and Multicast Data Paths*

Example 17-62 shows the relevant configuration for the topology in Figure 17-18.

**Example 17-62** *MBGP Configuration for Incongruent Unicast and Multicast Data Paths*

[Click here to view code image](#)

**R2**

```
router bgp 100
  bgp log-neighbor-changes
  no bgp default ipv4-unicast
  neighbor 10.23.1.3 remote-as 200
  neighbor 10.23.2.3 remote-as 200
  neighbor 192.168.1.1 remote-as 100
  neighbor 192.168.1.1 update-source Loopback0
  !
  address-family ipv4
    network 10.1.1.0 mask 255.255.255.0
    neighbor 10.23.1.3 activate
    neighbor 192.168.1.1 activate
  exit-address-family
  !
  address-family ipv4 multicast
    neighbor 10.23.2.3 activate
    neighbor 192.168.1.1 activate
  exit-address-family
```

**XR3**

```
router bgp 200
  address-family ipv4 unicast
    network 10.4.4.0/24
  !
  address-family ipv4 multicast
    network 10.4.4.0/24
  !
  neighbor 10.23.1.2
    remote-as 100
    address-family ipv4 unicast
      route-policy PASSALL in
      route-policy PASSALL out
  !
  !
  neighbor 10.23.2.2
    remote-as 100
    address-family ipv4 multicast
      route-policy PASSALL in
      route-policy PASSALL out
  !
  !
  neighbor 192.168.4.4
    remote-as 200
    update-source Loopback0
    address-family ipv4 unicast
  !
  address-family ipv4 multicast
```

**Example 17-63** demonstrates the RPF for source 10.4.4.10 is via the multicast link and that this information was learned via multicast BGP. The example also shows the source subnet 10.4.4.0/24 is known via unicast and multicast BGP, but multicast BGP is preferred (as dictated by the RPF rules).

## Example 17-63 Multicast BGP Is the Chosen RPF Source of Information

[Click here to view code image](#)

```
R2#show ip rpf 10.4.4.10
RPF information for ? (10.4.4.10)
  RPF interface: GigabitEthernet0/2
  RPF neighbor: ? (10.23.2.3)
  RPF route/mask: 10.4.4.0/24
  RPF type: multicast (bgp 100)
  Doing distance-preferred lookups across tables
  RPF topology: ipv4 multicast base
```

```
R2#show ip bgp ipv4 multicast
! Output omitted for brevity
```

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.4.4.0/24	10.23.2.3	2		0	200 i

```
R2#show ip bgp ipv4 unicast
! Output omitted for brevity
```

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	10.1.1.0/24	10.12.1.1	2		32768	i
*>	10.4.4.0/24	10.23.1.3	2		0	200 i

Example 17-64 demonstrates that once Receiver 1 joins multicast group 232.1.1.1 and the source starts transmitting for this group, the multicast traffic is expected to come in through incoming interface Gi0/2, which is on the multicast link. It also shows the RPF information was learned via multicast BGP and that the RPF neighbor is 10.23.2.3.

## Example 17-64 R2's Mroute Table

[Click here to view code image](#)

```
R2#show ip mroute 232.1.1.1
! Output omitted for brevity

(10.4.4.10, 232.1.1.1), 03:14:16/00:03:18, flags: sT
  Incoming interface: GigabitEthernet0/2, RPF nbr 10.23.2.3, Mbgp
  Outgoing interface list:
    GigabitEthernet0/3, Forward/Sparse, 03:14:16/00:03:18
```

Example 17-65 demonstrates that unicast traffic is using the unicast link.

## Example 17-65 Traceroute to Demonstrate Unicast Traffic Is Using 10.23.1.0/24 Link

[Click here to view code image](#)

```
R1#traceroute 10.4.4.10 source 10.1.1.1
Type escape sequence to abort.
Tracing the route to 10.4.4.10
VRF info: (vrf in name/id, vrf out name/id)
 1 10.12.1.2 2 msec 8 msec 5 msec
 2 10.23.1.3 6 msec 14 msec 10 msec
 3 10.34.1.4 18 msec 30 msec 25 msec
 4 10.4.4.10 [AS 200] 15 msec * 23 msec
```

```
R4#traceroute 10.1.1.10 source 10.4.4.4
Type escape sequence to abort.
Tracing the route to 10.1.1.10
VRF info: (vrf in name/id, vrf out name/id)
 1 10.34.1.3 2 msec 2 msec 2 msec
 2 10.23.1.2 7 msec 7 msec 12 msec
 3 10.12.1.1 15 msec 23 msec 26 msec
 4 10.1.1.10 [AS 100] 25 msec * 28 msec
```

Example 17-66 demonstrates that when the multicast link fails, the only source of RPF information is unicast BGP, so the unicast link will be used for multicast and unicast traffic.

### **Example 17-66** *Unicast BGP Is Chosen as the RPF Source of Information After Multicast Link Failure*

[Click here to view code image](#)

```
R2#show ip rpf 10.4.4.10
RPF information for ? (10.4.4.10)
RPF interface: GigabitEthernet0/1
RPF neighbor: ? (10.23.1.3)
RPF route/mask: 10.4.4.0/24
RPF type: unicast (bgp 100)
Doing distance-preferred lookups across tables
RPF topology: ipv4 multicast base, originated from ipv4 unicast base
```

Example 17-67 demonstrates that the multicast traffic is expected to come in through incoming interface Gi0/1, which is part of the 10.23.1.0/24 link, and that the RPF neighbor is 10.23.1.3.

### **Example 17-67** *R2's mroute Table After Multicast Link Failure*

[Click here to view code image](#)

```
R2#show ip mroute 232.1.1.1
! Output omitted for brevity

(10.4.4.10, 232.1.1.1), 03:36:13/00:02:36, flags: sT
Incoming interface: GigabitEthernet0/1, RPF nbr 10.23.1.3
Outgoing interface list:
GigabitEthernet0/3, Forward/Sparse, 00:03:17/00:02:36
```

## Static IGMP Joins

Sometimes it might be desirable to statically configure an interface to join a group as if there were a directly connected receiver on the interface or configure a router to emulate a multicast receiver. For example, static groups can be used in scenarios where multicast traffic needs to be forwarded out a port for processing by a video-quality monitoring system.

Static IGMP joins can be accomplished with two different commands. In IOS, this can be accomplished with the interface parameter command `ip igmp static-group group-address [source {source-address | ssm-map}]` and the interface parameter command `ip igmp join-group group-address [source source-address]`. In IOS XR, this can be accomplished with the command `static-group group-address [inc-mask mask count cnt] [source-address [inc-mask mask count cnt]]` under IGMP configuration mode and the command `join-group group-address [source-address]` under IGMP configuration mode.

### Note

Static groups configured with the `ip igmp static-group` command are CEF switched and do not have any performance impact on the router.

Figure 17-19 illustrates a topology that is a reference for the next series of examples. R3 and XR5 are test routers emulating multicast receivers.

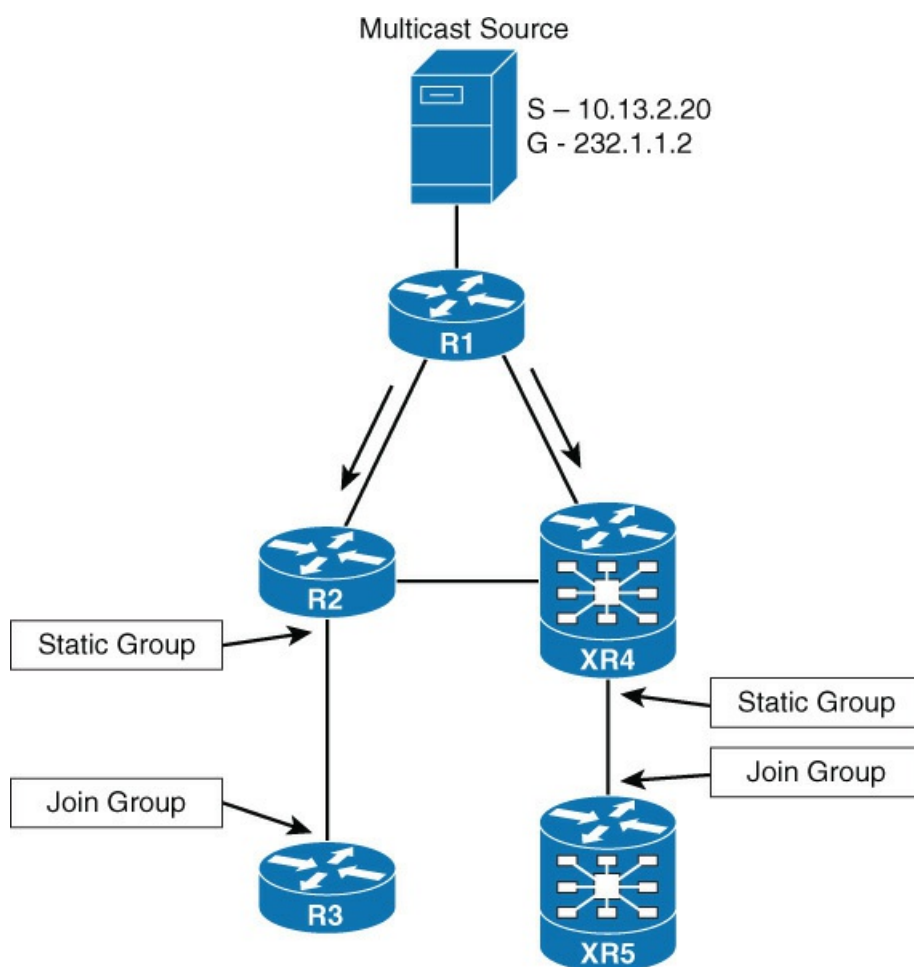


Figure 17-19 Reference Topology for IGMP Static IGMP Join Examples

Example 17-68 shows the configuration necessary on R3 and XR5 to send IGMPv2 joins for 232.1.1.2.

## Example 17-68 IGMPv2 Join Group Configuration

[Click here to view code image](#)

```
R3
interface GigabitEthernet0/1
 ip address 10.2.2.10 255.255.255.0
 ip igmp join-group 239.1.1.2
```

```
XR5
interface GigabitEthernet0/0/0/0
 ipv4 address 10.4.4.10 255.255.255.0
!
router igmp
 interface GigabitEthernet0/0/0/0
  join-group 239.1.1.2
```

[Example 17-69](#) shows the configuration necessary on R3 and XR5 to send IGMPv3 joins for 232.1.1.2.

## Example 17-69 IGMPv3 Join Group Configuration

[Click here to view code image](#)

```
R3
interface GigabitEthernet0/1
 ip igmp join-group 239.1.1.2 source 10.13.2.20
```

```
XR5
router igmp
 interface GigabitEthernet0/0/0/0
  join-group 239.1.1.2 10.13.2.20
!
```

### Note

Using the **join-group** commands in IOS and IOS XR can result in significant performance impact because all subscribed multicast packets are punted to the route processor. For this reason, it is strongly recommended to just use these commands in lab environments for testing purposes or temporarily in a production network for troubleshooting purposes.

[Example 17-70](#) shows how to configure static groups on R2 and XR4 on the interfaces facing R3 and XR5 respectively as if they had received IGMPv2 joins.

## Example 17-70 IGMPv2 Static Group Configuration

[Click here to view code image](#)

**R2**

```
interface GigabitEthernet0/1
 ip igmp static-group 239.1.1.2
```

**XR4**

```
router igmp
 interface GigabitEthernet0/0/0/0
 static-group 239.1.1.2
```

**Example 17-71** shows what the mroute table looks like on R3 and XR5 after configuring the static groups. The C flag indicates there are directly connected receivers for group 239.1.1.2.

**Example 17-71 mroute Table on R3 and XR5**

[Click here to view code image](#)

```
R3#show ip mroute 239.1.1.2
! Output omitted for brevity
```

```
(* , 239.1.1.2), 00:03:55/stopped, RP 0.0.0.0, flags: SJC
 Incoming interface: Null, RPF nbr 0.0.0.0
 Outgoing interface list:
   GigabitEthernet0/1, Forward/Sparse, 00:03:55/00:02:04
```

```
RP/0/0/CPU0:XR5#show mrib route 239.1.1.2
! Output omitted for brevity
```

```
(* , 239.1.1.2) RPF nbr: 0.0.0.0 Flags: C
 Up: 00:06:55
 Outgoing Interface List
   GigabitEthernet0/0/0/0 Flags: F NS LI, Up: 00:06:55
```

**Example 17-72** shows how to configure static groups on R2 and XR4 on the interfaces facing R3 and XR5, respectively, as if they had received IGMPv3 joins.

**Example 17-72 IGMPv3 Static Group Configuration**

[Click here to view code image](#)

**R2**

```
interface GigabitEthernet0/1
 ip igmp static-group 239.1.1.2 source 10.13.2.20
```

**XR4**

```
router igmp
 interface GigabitEthernet0/0/0/0
 static-group 239.1.1.2 10.13.2.20
```

Example 17-73 shows what the mroute table looks like on R3 and XR5 after configuring the static groups. Because the source and the group were specified, (S,G) state was created.

### Example 17-73 mroute Table on R3 and XR5

[Click here to view code image](#)

```
R3#show ip mroute 239.1.1.2
! Output omitted for brevity
(10.13.2.20, 239.1.1.2), 00:00:06/00:02:53, flags: sTI
  Incoming interface: GigabitEthernet0/2, RPF nbr 10.24.1.4, Mbgp
  Outgoing interface list:
    GigabitEthernet0/1, Forward/Sparse, 00:00:06/00:02:53

RP/0/0/CPU0:XR5#show mrrib route 239.1.1.2
! Output omitted for brevity
(10.13.2.20,239.1.1.2) RPF nbr: 10.34.1.3 Flags:
  Up: 00:01:55
  Incoming Interface List
    GigabitEthernet0/0/0/2 Flags: A, Up: 00:01:55
  Outgoing Interface List
    GigabitEthernet0/0/0/0 Flags: F NS LI, Up: 00:01:55
```

#### Note

For SSM static groups outside of the 232.0.0.0/8 range to work, SSM should be enabled using access lists and the **range** keyword, as described earlier in this chapter.

## MULTICAST TROUBLESHOOTING

Multiple **show** commands have been presented to describe the behavior of IP multicast routing throughout this chapter and Chapter 16. Table 17-3 lists the most important IOS and IOS XR **show** commands available that can be useful for baselining a multicast network or for troubleshooting purposes. In most troubleshooting scenarios, the commands for viewing the PIM topology table, **mrrib** and **mfib** are the most crucial.

OS	Command	Description
IOS	<code>show ip mroute <i>src-ip-address/group-address</i></code>	These commands can be used to display the PIM topology table, MRIB, and MFIB information. These are extremely useful commands for reviewing the multicast distribution tree state and troubleshooting purposes.
	<code>show ip mrrib route <i>src-ip-address/group-address</i></code>	
	<code>show ip mfib route <i>src-ip-address/group-address</i></code>	
XR	<code>show pim topology <i>src-ip-address/group-address</i></code>	These are extremely useful commands for reviewing the multicast distribution tree state and troubleshooting purposes.
	<code>show mrrib route <i>src-ip-address/group-address</i></code>	
	<code>show mfib route <i>src-ip-address/group-address</i></code>	

Table 17-3 **show** Commands Used in This Chapter

### Mtrace

Mtrace is a UNIX utility supported in IOS and IOS XR that traces the path a packet would take from a source to a receiver. Mtrace does not have to be initiated from a router on the shared or SPT, but the router has to be enabled for multicast routing. If mtrace is executed from multiple routers in the domain, the same mtrace results should be seen. The results provided are not meant to show exactly what the problem is, but it is useful for isolating multicast routing failures or identifying suboptimal paths.



The **mtrace** request generated by the **mtrace** command is multicast to the multicast group specified in the command to find the LHR to the specified receiver. The trace then follows the multicast path from destination (LHR) to the source (FHR) by passing the mtrace request packet via unicast hop by hop. Responses are unicast to the initiator of the mtrace request by the FHR.

Mtrace is executed from EXEC mode in IOS and IOS XR with the command **mtrace source-address destination-address**.

Figure 17-20 illustrates a PIM-SM topology that will be used in the coming mtrace utility examples.

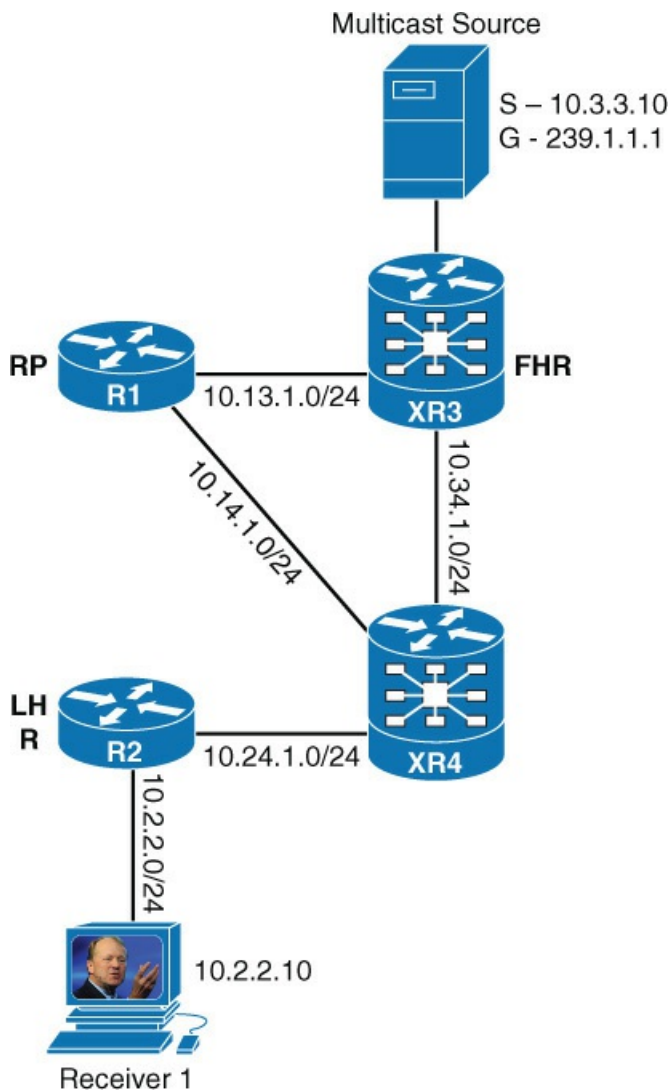


Figure 17-20 Reference Topology for Mtrace Examples

Receiver 1 has sent an IGMP join to R2 for multicast group 239.1.1.1. The source is not yet active, so there is only a shared tree formed between the R1 and R2. Example 17-74 shows what the mtrace output looks like when executed from R2 and XR4. The trace is executed upstream in reverse direction from the receiver to the RP, and it stops there because there is no active source.

#### Example 17-74 mtrace with No Active Source

[Click here to view code image](#)

```
R2#mtrace 10.3.3.10 10.2.2.10 239.1.1.2
Type escape sequence to abort.
Mtrace from 10.3.3.10 to 10.2.2.10 via group 239.1.1.2
From source (?) to destination (?)
Querying full reverse path...
 0 10.2.2.10
-1 10.2.2.2 ==> 10.24.1.2 PIM [using shared tree]
-2 10.24.1.4 ==> 10.14.1.4 PIM [10.3.3.0/24]
-3 10.14.1.1 ==> 0.0.0.0 PIM_MT Reached RP/Core [using shared tree]
```

```
RP/0/0/CPU0:XR4#mtrace 10.3.3.10 10.2.2.10 239.1.1.2
Type escape sequence to abort.
Querying full reverse path...

 0 10.2.2.10
-1 10.24.1.2 PIM [10.3.3.10/63]
-2 10.14.1.4 PIM [10.3.3.0/24]
-3 0.0.0.0 CBT/Special Reached RP/Core [10.3.3.10/63]
```

**Example 17-75** shows the output of the **mtrace** command when the source starts transmitting multicast traffic to group 239.1.1.1. The SPT switchover takes place immediately, and the RP is bypassed, as shown in the output.

### Example 17-75 mtrace with Active Source

[Click here to view code image](#)

```
R2#mtrace 10.3.3.10 10.2.2.10 239.1.1.1
Type escape sequence to abort.
Mtrace from 10.3.3.10 to 10.2.2.10 via group 239.1.1.1
From source (?) to destination (?)
Querying full reverse path...
 0 10.2.2.10
-1 10.2.2.2 ==> 10.24.1.2 PIM [10.3.3.0/24]
-2 10.24.1.4 ==> 10.14.1.4 PIM [10.3.3.0/24]
-3 10.34.1.3 ==> 10.3.3.3 PIM [10.3.3.0/24]
-4 10.3.3.10
```

```
RP/0/0/CPU0:XR4#mtrace 10.3.3.10 10.2.2.10 239.1.1.1
Type escape sequence to abort.
Querying full reverse path...

 0 10.2.2.10
-1 10.24.1.2 PIM [10.3.3.0/24]
-2 10.14.1.4 PIM [10.3.3.0/24]
-3 10.3.3.3 PIM [10.3.3.0/24]
```

**Example 17-76** shows the results of the **mtrace** command after causing a failure by disabling PIM on the link between R1 and XR3. The trace made it all the way to the RP and this indicates the RP should be checked for issues.

## Example 17-76 mtrace with PIM Disabled Between R1 and XR3

[Click here to view code image](#)

```
R2#mtrace 10.3.3.10 10.2.2.10 239.1.1.1
Type escape sequence to abort.
Mtrace from 10.3.3.10 to 10.2.2.10 via group 239.1.1.1
From source (?) to destination (?)
Querying full reverse path...
 0 10.2.2.10
-1 10.2.2.2 ==> 10.24.1.2 PIM [using shared tree]
-2 10.24.1.4 ==> 10.14.1.4 PIM [10.3.3.0/24]
-3 10.14.1.1 ==> 0.0.0.0 None No route
```

```
RP/0/0/CPU0:XR4#mtrace 10.3.3.10 10.2.2.2 239.1.1.1
Type escape sequence to abort.
Querying full reverse path...

 0 10.2.2.2
-1 10.24.1.2 PIM [10.3.3.10/63]
-2 10.14.1.4 PIM [10.3.3.0/24]
-3 0.0.0.0 None No route
RP/0/0/CPU0:XR4#
```

## SUMMARY

This chapter covered advanced multicast topics that are typically used in large-scale multicast deployments. It included an overview of interdomain routing, SSM, RP redundancy, SSM, multicast security concepts, traffic engineering, and troubleshooting.

MSDP allows RPs in the same or different PIM-SM domain share information about active sources. MBGP carries unicast prefixes to perform multicast RPF check calculations.

Rendezvous point redundancy key concepts covered in this chapter include the following:

- Multiple RPs can be deployed within a PIM domain for redundancy and load balancing.
- For Auto-RP, multiple MAs and C-RPs can be deployed. All MAs deployed are active at the same time and advertise the same information. The C-RP with the highest IP address is elected as the RP for an overlapping group range.
- For BSR, multiple C-BSRs can be deployed. The C-BSR with the highest priority is elected as the BSR; if priorities are equal, the C-BSR with the highest IP address is elected as the BSR.
- For BSR, multiple C-RPs can be deployed. The C-RP with the lowest priority is elected as the RP; if priorities are equal, the C-RP with the highest IP address is elected as the RP.
- Anycast RP with MSDP mesh groups can be deployed within a PIM domain for load balancing and redundancy. Static-RP, Auto-RP, and BSR can be used, with Static-RP being the most commonly used.

SSM is one of the most simple PIM protocol variations to deploy, maintain, and troubleshoot. SSM can be used for intradomain and interdomain deployments. For interdomain deployments, it does not require the use of MSDP. It uses only source trees, so there is no need to deploy RPs. SSM operates with IGMPv3 and requires IGMPv3 support on the multicast routers, the receiver where the application is running, and the application itself.

Multicast security should be deployed in every multicast-enabled network to secure it against network attacks. For any PIM domain running Auto-RP or BSR, at a minimum, multicast boundaries should be deployed to prevent protocol messages from leaking outside of the domain and to prevent unwanted PIM protocol traffic from coming into the domain.

Multicast traffic engineering key concepts covered in this chapter include the following:

- Static mroutes or MBGP can be used for multicast data path manipulation without affecting unicast routing.
- When there are multiple RPF sources for the same prefix, the longest match rule (prefix length) is used to pick the best route. If all prefix lengths are the same, then the lowest AD is preferred. If AD values are equal, the RPF sources are preferred in this order: static mroutes, MBGP, and finally unicast routes.
- Static IGMP joins can be used to pin traffic to a specific interface. This is useful for some applications that require always on traffic or lab testing purposes.

For multicast troubleshooting purposes, analyzing the PIM topology table, the MRIB and the MFIB, can help identify most multicast failures. The mtrace utility can be used for isolating multicast routing failures or identifying suboptimal paths.

## REFERENCES IN THIS CHAPTER

Williamson, Beau. *Developing IP Multicast Networks, Volume I*. Cisco Press, 1999.

Cisco. Cisco IOS Software Configuration Guides, <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides, <http://www.cisco.com>

RFC 3446, *Anycast Rendezvous Point (RP) Mechanism Using Protocol Independent Multicast (PIM) and Multicast Source Discovery Protocol (MSDP)*, D. Kim, D. Meyer, H. Kilmer, D. Farinacci, <http://www.ietf.org/rfc/rfc3446.txt>, January 2003

RFC 4607, *Source-Specific Multicast for IP*, H. Holbrook, B. Cain, <http://www.ietf.org/rfc/rfc4607.txt> August 2006

RFC 3618, *Multicast Source Discovery Protocol (MSDP)*, B. Fenner, D. Meyer, <http://www.ietf.org/rfc/rfc3618.txt> October 2003

RFC 4611, *Multicast Source Discovery Protocol (MSDP) Deployment Scenarios*, M. McBride, J. Meylor, D. Meyer, <http://www.ietf.org/rfc/rfc4611.txt> August 2006

RFC 1112, *Host Extensions for IP Multicasting*, S. Deering, IETF,  
<http://www.ietf.org/rfc/rfc1112.txt>, August 1989.

RFC 4601, *Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)*, B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, IETF,  
<http://www.ietf.org/rfc/rfc4601.txt>, August 2006.

RFC 3973, *Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)*, A. Adams, J. Nicholas, W. Siadak, IETF, <http://www.ietf.org/rfc/rfc3973.txt>, January 2005.

RFC 5015, *Bidirectional Protocol Independent Multicast (BIDIR-PIM)*, M. Handley, I. Kouvelas, T. Speakman, L. Vicisano, IETF, <http://www.ietf.org/rfc/rfc5015.txt>, October 2007.

RFC 5059, *Bootstrap Router (BSR) Mechanism for Protocol Independent Multicast (PIM)*, N. Bhaskar, A. Gall, J. Lingard, S. Venaas, IETF, <http://www.ietf.org/rfc/rfc5059.txt>, January 2008.

RFC 5771, *IANA Guidelines for IPv4 Multicast Address Assignments*, M. Cotton, L. Vegoda, D. Meyer, IETF, <http://www.ietf.org/rfc/rfc5771.txt>, March 2010.

RFC 2236, *Internet Group Management Protocol, Version 2*, W. Fenner, IETF,  
<http://www.ietf.org/rfc/rfc2236.txt>, November 1997

RFC 3376, *Internet Group Management Protocol, Version 3*, B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan, <http://www.ietf.org/rfc/rfc3376.txt>, October 2002.

IPv4 Multicast Address Space Registry, Stig Venaas,  
<http://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml>, May 2014

## Part VI: IPv6

## Chapter 18. IPv6 Addressing

This chapter covers the following topics:

- [IPv6 address structure](#)
- [IPv6 address types](#)
- [Neighbor Discovery Protocol](#)

IPv6 is the latest version of the Internet Protocol (IP). The development of the protocol began in the early 1990s when it became clear that IPv4 address exhaustion was a real possibility. IPv6 is designed to have an enormous address space, which is its main advantage over IPv4.

Unfortunately, the IPv6 protocol stack is not backward compatible with IPv4, so a host that has only the IPv4 stack enabled cannot directly communicate with another host that is using only IPv6. To do so, the devices either need to run both protocols, also known as *dual stack*, or leverage a network transition technology such as tunneling or protocol translation.

Many components of the IPv6 protocol are vastly different from IPv4. Complete coverage of the IPv6 protocol, its benefits, and deployment strategies is beyond the scope of this book. The focus of this chapter is on the IPv6 addressing structure, address types, and the Neighbor Discovery Protocol that is used for basic link-layer connectivity.

### IPV6 ADDRESS STRUCTURE

IPv6 uses a 128-bit long address, which allows for the creation of  $2^{128}$  or 340,282,366,920,938,463,374,607,431,770,000,000 unique IP addresses! IPv6's long address length makes it unwieldy to represent in dotted-decimal notation, so instead it is represented in colon-hexadecimal text notation. Each hexadecimal character represents 4 binary bits that are called a *nibble*. Four nibbles represent a *hexet* or 16 binary bits. A total of 8 hexets are used to represent the entire 128 bit IPv6 address. In addition, a colon (:) separates every hexet of the address to make the full address easier to read.

Example IPv6 address: fe80:0000:0000:0000:c800:00ef:fe74:0c00

#### Note

RFC 5952 recommends writing an IPv6 address in lowercase characters to ensure compatibility with case-sensitive applications. IOS and IOS XR are not case-sensitive and will accept either a lowercase or an uppercase text representation of an IPv6 address.

**Figure 18-1** demonstrates the bit length values for an IPv6 address. A group of 4 hexadecimal characters is equivalent to 16 binary bits for a total of 8 hexets.

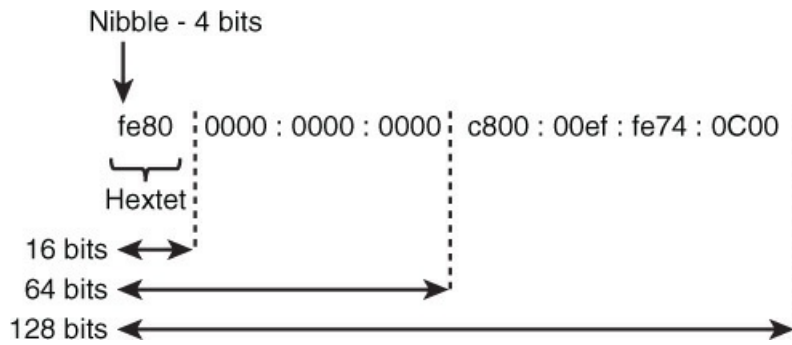


Figure 18-1 IPv6 Bit Length

An IPv6 unicast address consists of two components: the *subnet prefix* and *interface identifier*. The subnet prefix is used to identify the network, and the interface ID is used to identify a unique interface within the network.

Note

IPv6 addressing allows multiple IP addresses to be configured on an interface. The IPv4 concept of a primary and secondary interface address does not exist.

The IPv6 subnet prefix provides the same network boundary function as IPv4's network ID. A host may send a packet directly to another host if its subnet prefix is the same, similarly to how an IPv4 host may send a packet directly to another host with the same network ID. If the subnet prefix is different, a router is needed to provide the communication between networks.

IPv4 uses a dotted-decimal subnet mask to define the network boundary and calculate the network ID. In comparison, IPv6 uses the simpler prefix length notation to identify the subnet prefix network boundary. The IPv6 prefix works in exactly the same way as IPv4 CIDR notation. The number following the (/) in the prefix notation indicates the number of binary bits that comprise the subnet prefix length.

Figure 18-2 identifies the components of an IPv6 address. The address `2001:0db8:0000:0000:c800:00ef:fe74:0c00/64` is composed of a 64-bit long subnet prefix of `2001:0db8:0000:0000` and an interface ID of `c800:00ef:fe74:0c00`.

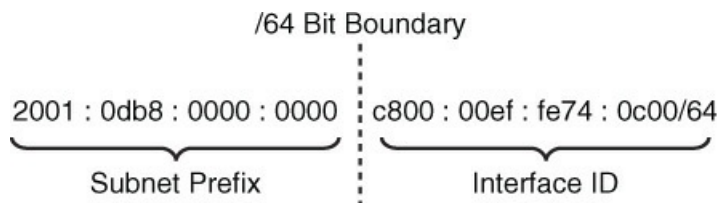


Figure 18-2 IPv6 Address Components

**Text Representation Address Abbreviation**

It is common for an address to contain long strings of 0 bits. RFC 4291 outlines special rules that allow for the abbreviation of the address to make it easier to work with. The address can be compressed using the two abbreviation rules outlined in Table 18-1.



Rule 1	Leading 0s	The leading 0s in a 16-bit field may be removed as long as there is at least 1 character in each field (except for the case described in rule 2).
Rule 2	Groups of 0s	Double colons (::) may be used to indicate multiple groups of 16-bits of 0s. The :: can appear only once in an address.

Table 18-1 RFC 4291 Address Abbreviation Rules

To implement rule 1, examine the address for groups that start with a leading 0. Figure 18-3 demonstrates the leading 0s that may be removed to compress the address.

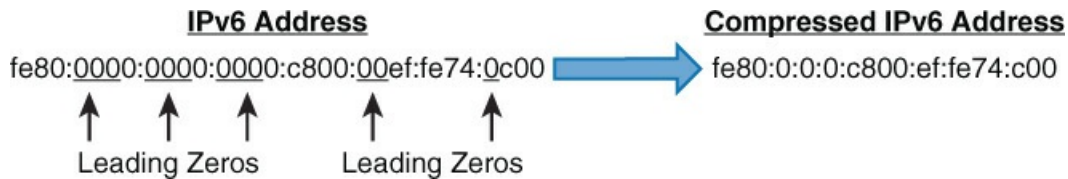


Figure 18-3 Rule 1: Leading 0s

To implement rule number 2, the :: abbreviation syntax can be used to compress a single-0 16-bit group or a series of contiguous 0-bit groups within the address. Figure 18-4 demonstrates the consecutive 0s that may be removed to compress the address.

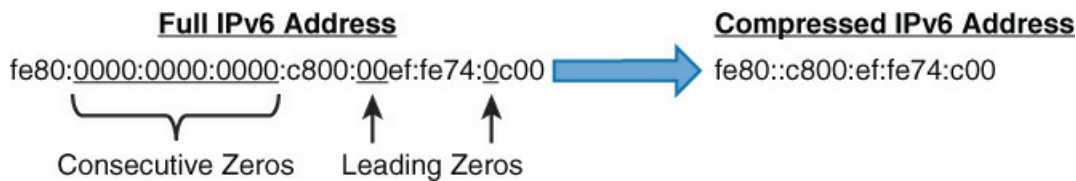


Figure 18-4 Rule 2: Groups of 0s

Table 18-2 provides additional examples of compressing an address using the leading 0s and groups of 0s text representation rules. The 0s that are shaded in the Full Address column are removed to form the compressed address. Notice that in the last example either compressed address format is acceptable as long as the rules for compressed 0s are followed properly.

Full IPv6 Address	Compressed IPv6 Address
fe80:0000:0000:0000:1300:a000:0000:0001	fe80::1300:a000:0:1
fe80:000a:000b:000c:0000:0000:0abc:1001	fe80:a:b:c::abc:1001
fe80:0000:0000:0000:0000:0000:0000:0001	fe80::1
fe80:0000:0000:000a:000b:0000:0000:0001	fe80::a:b:0:0:1
or	or
fe80:0000:0000:000a:000b:0000:0000:0001	fe80:0:0:a:b::1

Table 18-2 IPv6 Address Abbreviation Examples

The address abbreviation rules may also be used to compress the text representation of an IPv6 prefix. Table 18-3 displays the prefix 2001:0db8:0000:0000:0000:0000:0000:0000/32 in compressed format. The first row demonstrates compressing the leading 0s, and the second row

demonstrates compressing the leading os and groups of os. Typically, an IPv6 address or prefix is notated using both rules.

Full IPv6 Prefix	Compressed IPv6 Prefix
2001:0db8:0000:0000:0000:0000:0000/32	2001:db8:0:0:0:0:0/32
2001:0db8:0000:0000:0000:0000:0000/32	2001:db8::/32

Table 18-3 IPv6 Prefix Abbreviation

### IPv6 Hexadecimal to Binary Conversion

A hexadecimal address can represent 16 unique values (base 16). The 16 values are 0–9 and A–F. Table 18-4 provides a reference for the hex to decimal to binary values.

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Table 18-4 Hex to Decimal to Binary Values

Four binary bits (1 nibble) are necessary for creating the 16 possible hex address combinations. Binary numbers can have only two values, represented as either a 1 or a 0. If the binary bit value is 1, it is considered “on” and counts toward the computed decimal value of the IPv6 hex address. If the bit is 0, it is considered “off” and does not count toward the final decimal value tally. Table 18-5 displays the four possible binary bit values for a hex address.

Bit	1	2	3	4
Value	8	4	2	1

Table 18-5 Binary Bit Value

Figure 18-5 demonstrates converting the first 4 hex character (16 bits) of the IPv6 address fe80::c800:00ef:fe74:0c00 to binary. To simplify the conversion, the 4 hex characters fe80 are individually examined and converted to decimal. The binary value for each 4-bit nibble is calculated from each decimal value. Refer to Chapter 2, “IPv4 Addressing,” for a refresher on decimal to binary calculations.

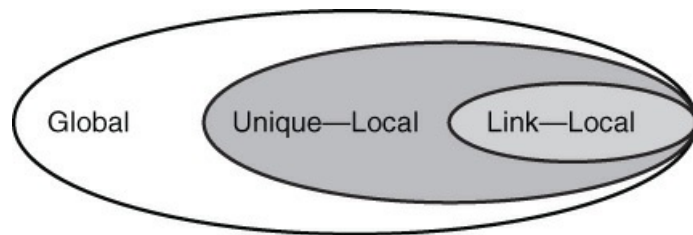


■ **Multicast:** A multicast address is used to stream packets to one or multiple destinations. A host joins a multicast group to indicate that it wants to receive the packet stream.

IPv6 does not include a broadcast address type. Instead, multicast provides the mechanism for allowing communication with multiple hosts at the same time.

### Unicast

IPv6 unicast addresses work similarly to how they do in IPv4. A unicast address provides connectivity for one interface or host. [Figure 18-6](#) illustrates the three IPv6 unicast reachability scopes, explained in the list that follows.



**Figure 18-6** IPv6 Unicast Address Scope

■ **Global:** A global address is similar to an IPv4 public IP address. It is globally routable and may be used to communicate with the Internet.

■ **Unique local unicast:** A unique local address is similar to an IPv4 private IP address. It is routable, but is intended only to be used for internal private communication within an organization.

■ **Link-local:** A link-local address is nonroutable and is used only for communicating on a directly connected link

#### Note

RFC 3513 outlines an additional site-local IPv6 unicast address that may be used within a site and is not globally routable. This address type is deprecated by RFC 3879 and is no longer supported.

[Example 18-1](#) demonstrates how to apply an IPv6 address to an interface. The interface command **ipv6 address *ipv6-address/prefix-length*** applies the address to the interface. IOS routers do not perform routing for the IPv6 address family by default. The global command **ipv6 unicast-routing** enables IPv6 routing.

### Example 18-1 IPv6 Address Configuration

[Click here to view code image](#)

#### IOS

```
ipv6 unicast-routing
interface GigabitEthernet2/0
ipv6 address 2001:db8::1/64
```

#### IOS XR

```
interface GigabitEthernet0/0/0/1
ipv6 address 2001:db8::1/64
```

#### Note

IPv6 Cisco Express Forwarding (CEF) is not enabled by default on IOS routers. The global configuration command **ipv6 cef** enables IPv6 CEF.

The command **show ipv6 interface** or **show ipv6 interface brief** may be used in both IOS and IOS XR to verify the interface address assignments. [Example 18-2](#) demonstrates the output for the **show ipv6 interface brief** command.

### Example 18-2 IPv6 Interface Status

[Click here to view code image](#)

```
R1#show ipv6 interface brief
! Output omitted for brevity
GigabitEthernet0/0      [up/up]
FE80::8A43:E1FF:FE80:FB48
2001:db8::1
```

```
RP/0/0/CPU0:XR1#show ipv6 interface brief
! Output omitted for brevity
GigabitEthernet0/0/0/0 [Up/Up]
fe80::7:c0ff:fe7f:f959
2001:db8::1
```

The commands **show ipv6 traffic** and **show interface interface-type interface-number accounting** may be used in IOS and IOS XR to view packet statistics. [Example 18-3](#) displays the IPv6 packet statistics for an IOS and IOS XR interface.

### Example 18-3 IPv6 Interface Status

[Click here to view code image](#)

```
R1#show interfaces GigabitEthernet 0/0 accounting
GigabitEthernet0/0
Protocol  Pkts In  Chars In  Pkts Out  Chars Out
! Output omitted for brevity
IPv6     583     679322    555     587986
RP/0/RSP0/CPU0:XR1#show interfaces GigabitEthernet 0/0/0/0 accounting
GigabitEthernet0/0/0/0
Protocol          Pkts In      Chars In      Pkts Out      Chars Out
IPV6_UNICAST      37           2544           0              0
IPV6_MULTICAST   103          8982           0              0
IPV6_ND           122         11924          90            7440
```

## Global Unicast

A global unicast address (GUA) is simply an address that is routable on the Internet. Currently, all global unicast addressing is allocated from the 2000::

### Note

A full list of the globally allocated unicast prefixes is available at <http://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xhtml>.

IPv6 has adopted a hierarchical approach to address allocations to simplify summarization and improve route table efficiency in comparison to the older IPv4 model. At the highest level, IANA allocates address pools as large as /12 to regional Internet registries (RIRs) from the massive 2000::

In the Asia Pacific region, there may be an additional layer to the IP address allocation hierarchy, depending on which county the address request originates. A National Internet Registry requests direct allocation from the RIR (APNIC).

### Note

An organization may request a direct allocation from the RIR. Direct allocations are called *provider-independent (PI)* address space and can be used with any ISP to allow connectivity to the Internet. The size of the prefix allocation is usually /48 but can vary in size depending on the organization's needs. In 2011, RFC 6177 recommended the Internet community adopt the more conservative /56 prefix allocation for general use to avoid wasting unused address space.

Figure 18-7 demonstrates how IPv6 prefix blocks are hierarchically allocated for the Internet community.

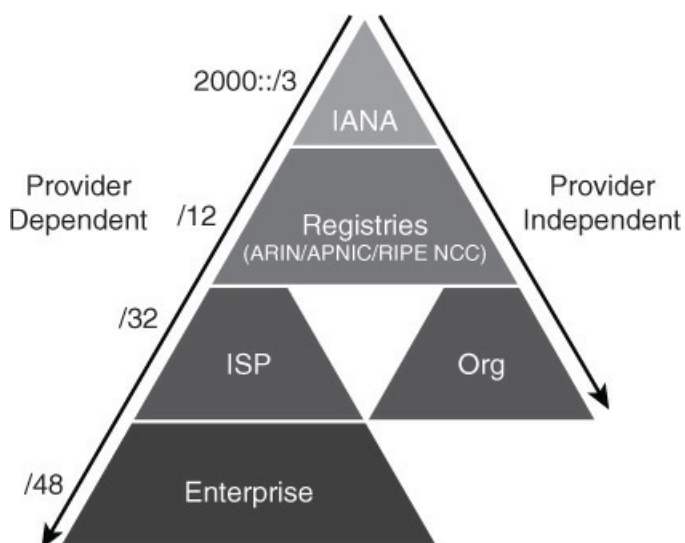


Figure 18-7 IPv6 Address Allocation Hierarchy

The global unicast address format makes it possible to easily summarize prefixes on the Internet.

The global routing prefix is hierarchically structured using the following format:

- Global routing prefix
- Subnet ID
- Interface ID

Figure 18-8 displays the IPv6 hierarchically structured address format.

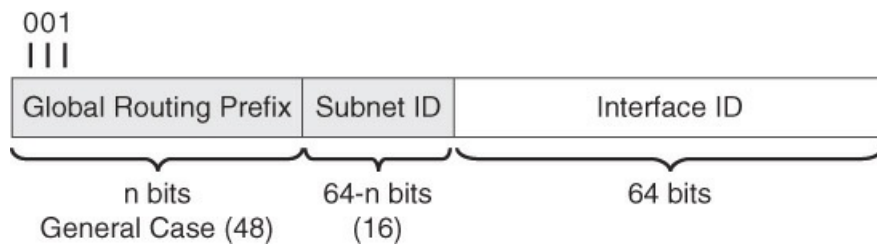


Figure 18-8 IPv6 Aggregatable Global Address Format

The global routing prefix field represents the leading bits from the address allocation. The remaining bits are available for the organization to assign as addresses. A general use /48 prefix allocation provides an organization with 80 bits of usable IP address space. It is highly recommended that this address block is subnetted into /64 subnet prefixes. Implementing /64 subnets provides an organization with 65,536 ( $2^{16}$ ) unique networks. The remaining 64 bits in the address are used to create the interface ID for identifying an interface on the network. Incredibly, a single /64 subnet contains more addresses than the entire IPv4 address space:

**IPv6 /64 subnet ( $2^{64}$ ):** 18,446,744,073,709,551,616

**IPv4 entire address space ( $2^{32}$ ):** 4,294,967,296

Figure 18-9 demonstrates how a network administrator might introduce route aggregation boundaries throughout the organization for a /48 prefix allocation. In the following example, three layers of summarization are introduced at the local, regional, and organization levels.

**/56:** Local offices

**/52:** Regional

**/48:** Organization

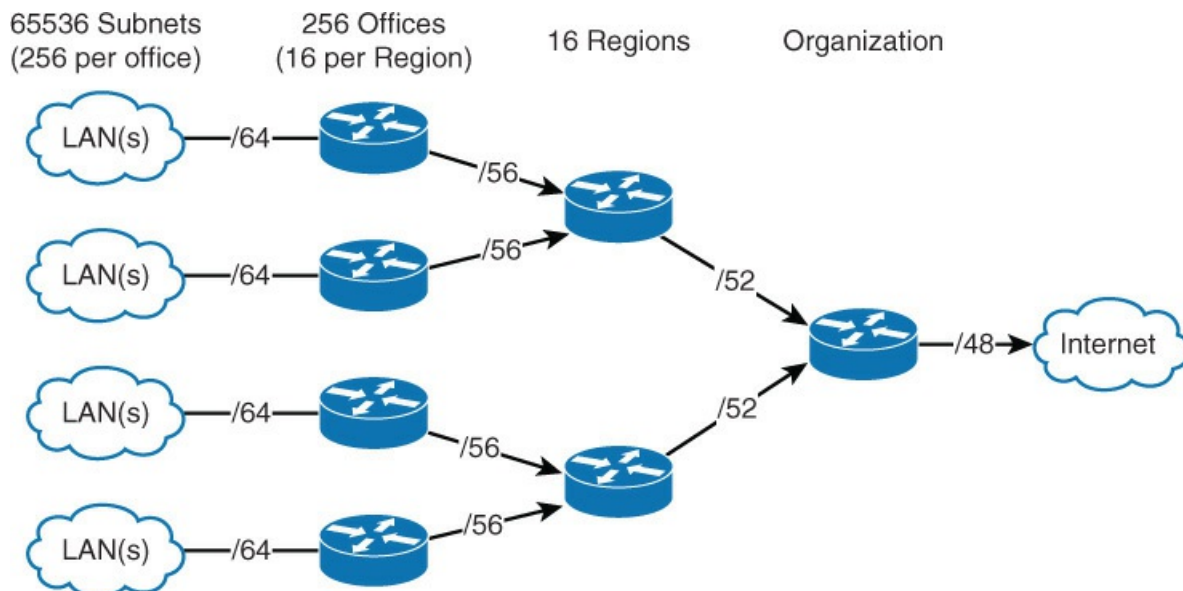


Figure 18-9 IPv6 Route Aggregation Plan

The network `2001:db8:1000::/48` is used to describe the subnet boundaries in Figure 18-9. In the illustration, the route aggregation is based on the 4-bit nibble boundary or hex character of the subnet ID, which begins at the leading binary bit 49. Allocating a 4-bit nibble provides 16 regions ( $2^4$ ) with a prefix aggregation length of /52. Prefix numbering begins at 0 rather than 1; therefore, the local offices and regional names will do so as well.

The region 0 is assigned `2001:db8:1000:0::/52`, region 1 is assigned `2001:db8:1000:1::/52` until finally region 15 is assigned `2001:db8:1000:F::/52`.

Each region is further subdivided into 16 regional offices ( $2^4$ ) using another 4-bit nibble with an aggregation prefix length of /56. For example, if region 0 is `2001:db8:1000:0::/52` and region 1 is `2001:db8:1000:1::/52`, each office in region 0 would be assigned a prefix in the range of `2001:db8:1000:00::/56` to `2001:db8:1000:0F::/56` compared to region 7, which would have an assignment in the range of `2001:db8:1000:71::/56` to `2001:db8:1000:7F::/56` for each office.

Finally, the remaining 8 bits of the subnet ID provide for 256 /64 prefix blocks ( $2^8$ ) per office. Figure 18-10 outlines how the bit boundaries are identified in the prefix address.

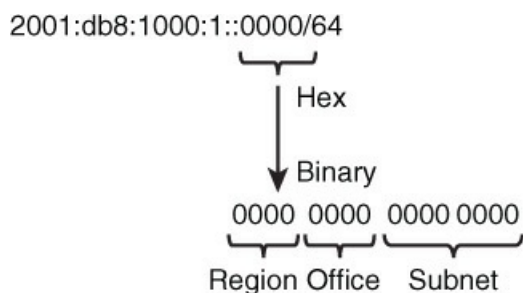


Figure 18-10 Prefix Bit Boundaries

Table 18-7 provides a helpful subnetting chart for calculating the number of available /64 subnet blocks based on the length of the subnet ID value.



Prefix Length	Subnet ID Length	/64 Subnets per Block
/48	1	65,536
/49	2	32,768
/50	4	16,384
/51	8	8192
/52	16	4096
/53	32	2048
/54	64	1024
/55	128	512
/56	256	256
/57	512	128
/58	1024	64
/59	2048	32
/60	4096	16
/61	8192	8
/62	16,384	4
/63	32,768	2
/64	65,536	1

**Table 18-7** IPv6 Subnetting Chart

### Unique Local Unicast

RFC 4193 defines the `fc00::/7` prefix block for unique local unicast (ULA) addressing. ULA addresses may be used to provide internal communication within an organization. In many respects, ULA addresses are similar to the IPv4 private addresses in the range: `10.0.0.0/8`, `172.16.0.0/12` and `192.168.0.0/16`. The ULA addresses have a limited scope and are not intended to be globally routable, and therefore they should be filtered at the Internet boundaries between ISPs and enterprises.

If Internet connectivity is required, a host will need to configure an additional globally routable address on its interface. [Figure 18-11](#) demonstrates how ULA and global unicast addresses may be used together to provide internal and external connectivity. In the illustration, the network with ULA addressing provides local connectivity between a corporate user PC (`fd00::a/64`) and a printer (`fd00::b/64`), while the global unicast address `2001:db8:1::a/64` provides the corporate user's PC connectivity to the Internet.

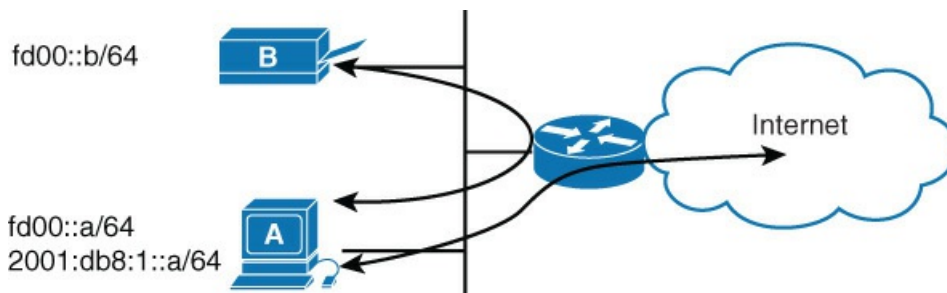


Figure 18-11 ULA and Global Address Communication

Note

RFC 6724 states that hosts should prefer to communicate using global unicast addresses over ULA addresses because ULA addresses might not be reachable end to end.

Figure 18-12 displays the fields of a ULA address. ULA addresses contain five fields: prefix, Local bit, global ID, subnet ID, and Interface ID.

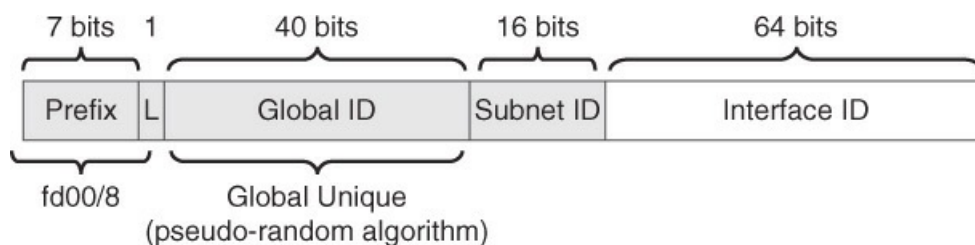


Figure 18-12 Unique Local Address Fields

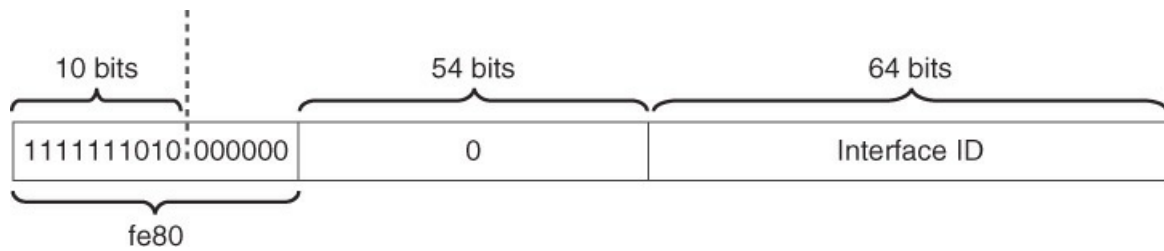
The address always starts with the binary value 1111 110, which maps to the prefix fcoo::/7. The L bit is currently set to 1 to indicate that the address is locally assigned. Setting the L bit to 0 has not yet been defined and is reserved for future use. Combining the prefix and the L bit create two potential address blocks:

- **fcoo::/8**: Reserved for future use
- **fdoo::/8**: Locally assigned

It is highly recommended that every organization generate a unique global ID. The goal of doing so is to allow sites or enterprises to easily merge, thus avoiding the hassle of IP renumbering that is present when there are overlapping address ranges. The full algorithm instructions for generating the pseudo-random global ID are outlined in section 3.2.2 of RFC 4193. Alternatively, several Internet sites can be used to generate unique local addresses. For example, the popular site SixXS provides a free ULA generator tool: <https://www.sixxs.net/tools/grh/ula/>.

### Link-Local Unicast

A link-local address (LLA) is a nonroutable address that is strictly used for local communication on a directly connected link. It is primarily used for neighbor discovery, duplicate address detection, and for interior gateway protocol (IGP) communication. The entire address block fe80::/10 has been reserved for link-local addressing. The first 10 bits start with 1111 1110 10 (fe8), the next 54 bits are all 0s, and the final 64 bits are allocated for the interface ID. Figure 18-13 displays the LLA format.



**Figure 18-13** Link-Local Address Format

An LLA address may be assigned manually or dynamically using IPv6’s default built-in stateless address autoconfiguration (SLAAC) mechanism. When using the dynamic method, Cisco routers always assigns an address to the interface with the network prefix `fe80::/64`. The Interface ID of the address is derived from the interface MAC address using the modified EUI-64 format. Full coverage of the EUI-64 interface identifier is covered later in this chapter in the section, “[IPv6 Stateless Address Autoconfiguration](#).”

The LLA is automatically assigned when IPv6 is enabled on an interface or when an address is first assigned to the interface. [Example 18-4](#) demonstrates how to enable IPv6 on an interface using the command **`ipv6 enable`**.

#### **Example 18-4** Enabling IPv6 on an Interface

[Click here to view code image](#)

```
IOS
ipv6 unicast-routing
interface GigabitEthernet1/0
ipv6 enable
```

```
IOS XR
interface GigabitEthernet0/0/0/0
ipv6 enable
```

Notice in [Example 18-5](#) that an LLA EUI-64 formatted address is automatically assigned to the XR router interface Gigabit Ethernet 0/0/0/0 once the protocol is enabled.

#### **Example 18-5** LLA Assignment

[Click here to view code image](#)

```
RP/0/RSP0/CPU0:XR1#show ipv6 interface GigabitEthernet 0/0/0/0 brief
GigabitEthernet0/0/0/0 [Up/Up]
fe80::6e9c:edff:fe7f:fb32
```

The link-local IP address can be manually assigned to an interface to make it easier to read and work with. For example, manually assigning a LLA to an interface avoids a potential address change if the associated line card is replaced in the future. [Example 18-6](#) demonstrates how to manually assign an LLA address using the command **`ipv6 address ipv6-address link-local`**.

## Example 18-6 Manually Assigning an IPv6 LLA

[Click here to view code image](#)

### IOS

```
ipv6 unicast-routing
interface GigabitEthernet2/0
ipv6 address FE80::1 link-local
```

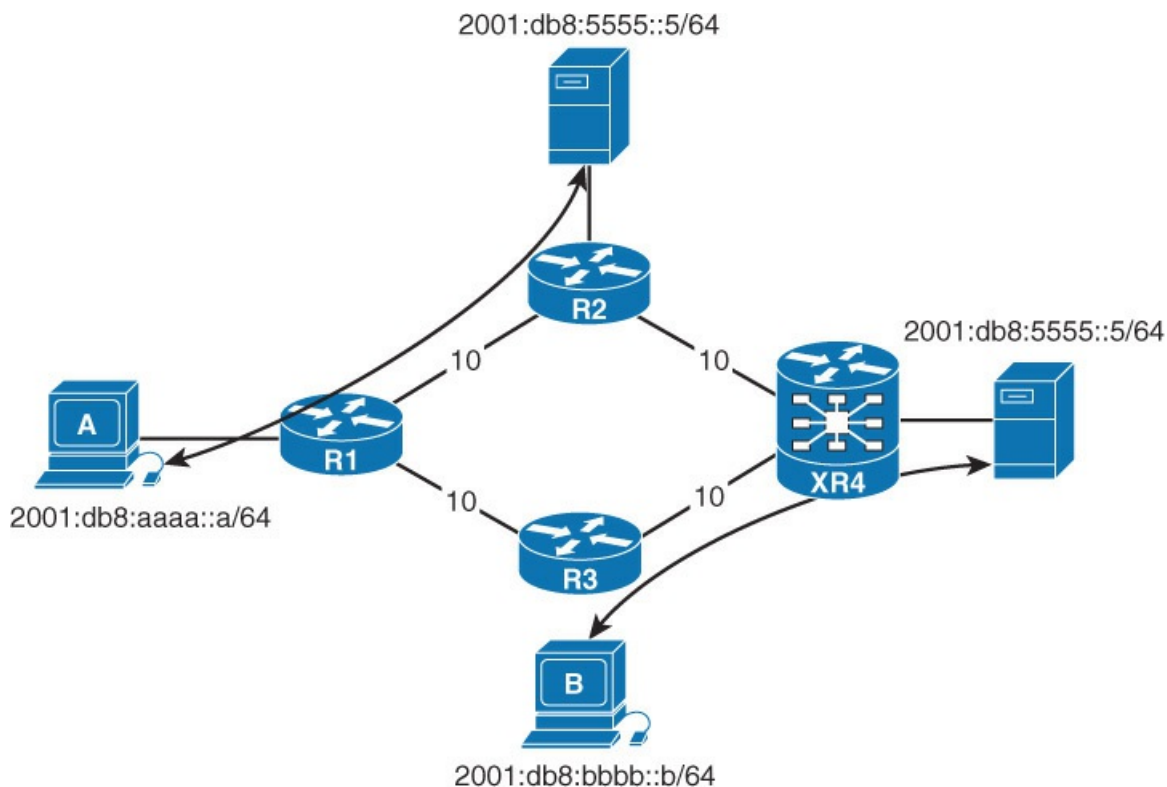
### IOS XR

```
interface GigabitEthernet0/0/0/1
ipv6 address fe80::2 link-local
```

## Anycast

Anycast addresses are similar to unicast addresses because they share the same address space and format. The difference is that a unicast address is assigned to one interface, whereas an anycast address may be assigned to multiple nodes or interfaces in the network.

Anycast addresses are typically used for fault tolerance. Two or more networks may be set up with identical addressing. **Figure 18-14** illustrates a network topology where two servers share the same address but are physically located in different parts of the network. Traffic destined to the anycast address `2001:db8:5555::5` will be forwarded to the closest server as identified by the routing table. Router R1 will forward the anycast traffic from client A to router R2 because it is the shortest path to network `2001:db8:5555::/64`, while router R3 will forward traffic from client B to router XR4's attached network.



**Figure 18-14** Anycast Network Topology

A network outage on R2 will trigger a network topology update. R1 recalculates the network path

for 2001:db8:5555::/64 and determines that traffic destined to the anycast server address 2001:db8:5555::5 should take the R1, R3, XR4 network path. Figure 18-15 shows the updated network path for the anycast topology.

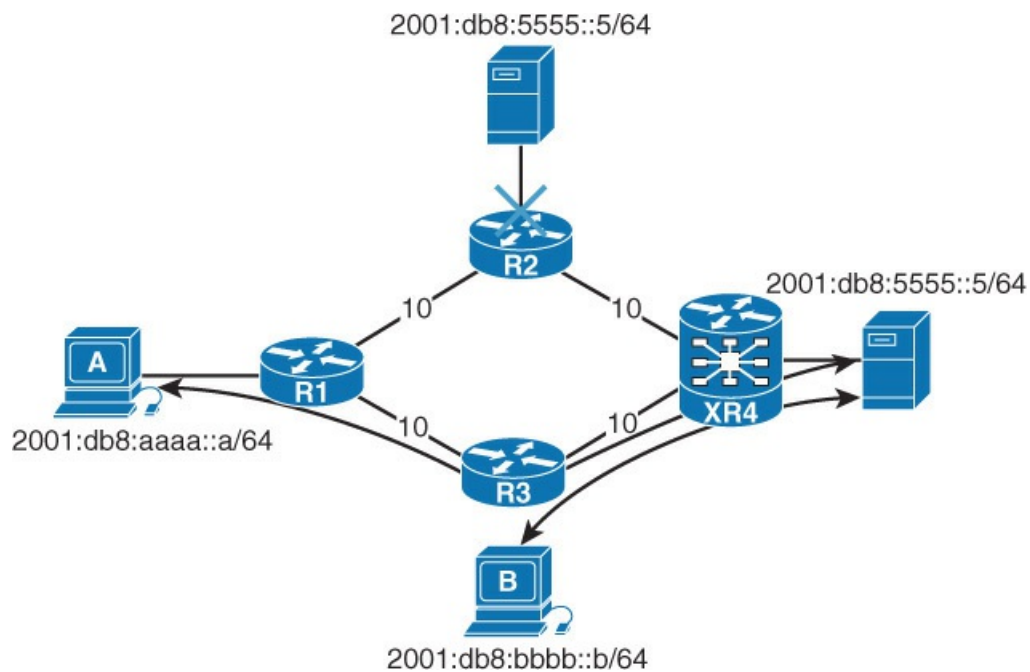


Figure 18-15 Updated Anycast Network Topology

Anycast addresses are configured in the same manner as unicast addresses. In some instances it may be desirable to disable duplicate address detection (DAD) for the network. To disable DAD the interface command in IOS is **ipv6 address ipv6-address/prefix-length anycast** and the command in IOS XR is **ipv6 nd dad attempts 0**. The IOS command disables DAD for the prefix. The IOS XR command disables DAD for the entire interface, including the link-local prefix. Use caution when disabling the DAD protection in IOS XR because all prefixes will be impacted.

Example 18-7 demonstrates disabling duplicate address detection for an anycast network.

### Example 18-7 Disabling Duplicate Address Detection

[Click here to view code image](#)

#### IOS

```
interface GigabitEthernet0/0
ipv6 address 2001:DB8:15:1111::1/64 anycast
```

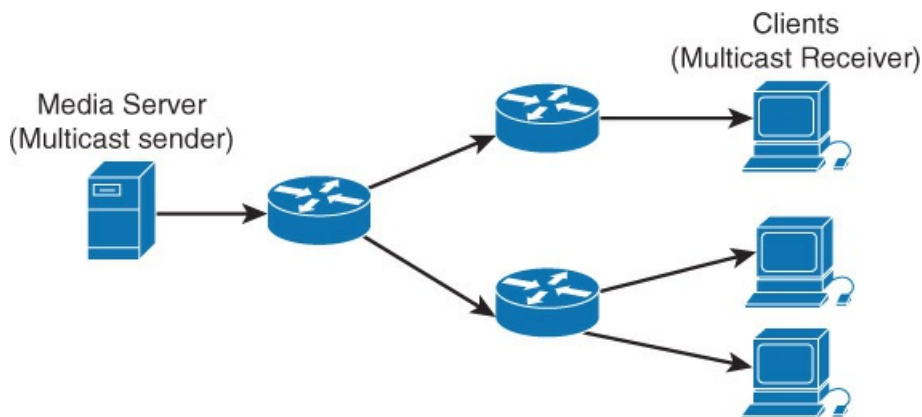
#### IOS XR

```
interface GigabitEthernet0/0/0/0
ipv6 nd dad attempts 0
```

## Multicast

IP multicast has the capability to stream packets to one or many destinations. The address is not actually configured on an interface; instead, the end device signals that it would like to join a multicast group and receive the stream for the associated address. IPv6 routers use multicast for a variety of network connectivity purposes, such as communicating routing protocol information, neighbor discovery, and IPv6 address resolution. A full coverage of the neighbor discovery protocol is covered later in the section “[Neighbor Discovery Protocol](#).”

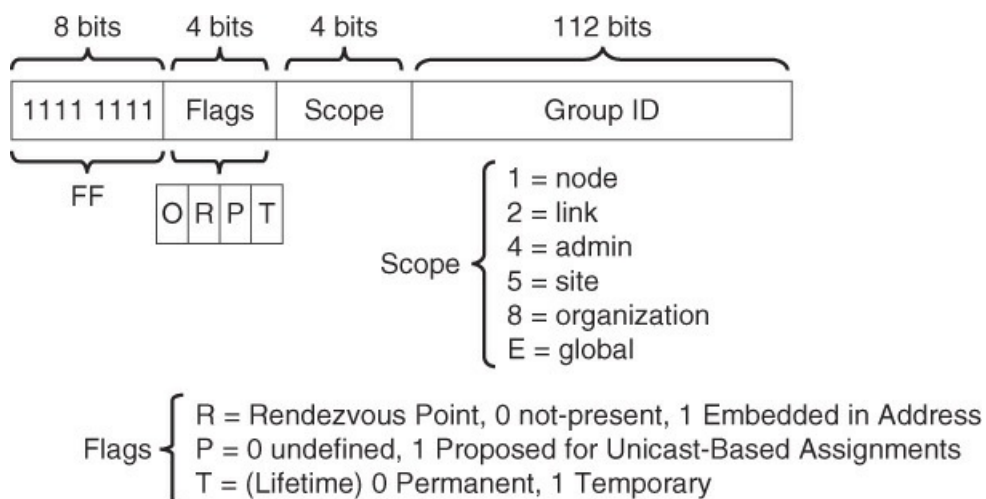
[Figure 18-16](#) illustrates a network topology where clients are receiving a multicast stream from a media server. Notice that the media server is sending one packet stream and that the routers are replicating the stream for each network segment that has clients.



**Figure 18-16** Multicast Network Topology

IPv6 multicasts prefix always starts with the leading binary value 1111 1111. This means that all multicast addresses will be allocated from the `ff::/8` address block. The next 8 bits of the address determine the flags and scope of the address. The last 112 bits of the address is the Group ID, which is used to uniquely identify the multicast service.

[Figure 18-17](#) displays the fields of an IPv6 multicast address.



**Figure 18-17** IPv6 Multicast Address Format

The transient flag within the flags field has two possible bit values:

- 0 for permanent
- 1 for temporary

A permanent address is a well-known address that is formally assigned by IANA. A temporary address is transient and may be used for testing. The Scope field is used by the router to determine whether the multicast traffic may be forwarded. For example, the Enhanced Interior Gateway Routing Protocol (EIGRP) multicast address ff02::a has a link-local scope and cannot be forwarded by the router beyond the local link. [Table 18-8](#) describes the RFC 4291 multicast scope types.

Scope Type	IPv6 Prefix	Description
Interface-Local	ff01	The interface-local scope does not allow multicast packets to traverse an interface. The scope provides a similar function to a unicast loopback address because the multicast packets are contained within the node.
Link-Local	ff02	The link-local scope contains multicast traffic to the local interface link.
Admin-Local	ff04	The admin-local scope is routable and is considered the smallest administrative scope. The scope boundary has no defined restriction but typically traffic is limited to a single department or floor in a building.
Site-Local	ff05	The site-local scope is routable and is considered larger than an admin-local scope. The boundary is usually limited to a building or region.
Organization-Local	ff08	Organization-local scope is routable and should be contained to a single organization.
Global	ff0e	The global scope does not have any reachability restrictions.

**Table 18-8** Multicast Scope Types

[Table 18-9](#) displays a list of the most common multicast addresses. A more comprehensive list of assignments is available on the IANA website: <http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml>.

Address	Description
ff02::1	All nodes address
ff02::2	All-routers address
ff02::5	OSPFv3 all routers
ff02::6	OSPFv3 all designated router (DR) address
ff02::8	IS-IS
ff02::9	RIPng
ff02::a	EIGRP
ff02::d	PIM
ff02::16	MLDv2 reports
ff02::1:2	Link-scoped all DHCPv6 relay agents and servers
ff05::1:3	Site-scoped all DHCPv6 servers
ff02::1:ffxx:xxxx	The solicited-node address is used by neighbor discovery for address resolution (The last 24 bits [xx:xxxx] of the address map to the last 24 bits of the IPv6 unicast address.)

**Table 18-9** Common IPv6 Multicast Addresses

Every device automatically joins the all nodes (ff02::1) and solicited-node (ff02::1:ffxx:xxxx) multicast groups. The all-node group is used to communicate with all interfaces on the local link, and the solicited-nodes multicast group is required for link-layer address resolution. Routers also join a third multicast group, the all-routers group (ff02::2).

Example 18-8 displays the output of the **show ipv6 interface** *interface-type interface-number* command. Notice that the default behavior of the routers is to join the all nodes, solicited node, and all router multicast groups.

### **Example 18-8** Active IPv6 Multicast Groups

[Click here to view code image](#)



```

R1#show ipv6 interface GigabitEthernet 0/0
GigabitEthernet0/0 is up, line protocol is up
IPv6 is enabled, link-local address is FE80::8A43:E1FF:FE80:FB48
No Virtual link-local address(es):
No global unicast address is configured
Joined group address(es):
FF02::1
FF02::2
FF02::1:FF80:FB48
MTU is 1500 bytes
! Output omitted for brevity
RP/0/RSP0/CPU0:XR1#show ipv6 interface GigabitEthernet 0/0/0/0
GigabitEthernet0/0/0/0 is Up, ipv6 protocol is Up, Vrfid is default (0x60000000)
IPv6 is enabled, link-local address is fe80::6e9c:edff:fe7f:fb32
No global unicast address is configured
Joined group address(es): ff02::1:ff7f:fb32 ff02::2 ff02::1
MTU is 1514 (1500 is available to IPv6)
! Output omitted for brevity

```

## Special IPv6 Addresses

In addition to the common unicast address type, there are two special purpose addresses:

- **Unspecified address:** The unspecified address 0:0:0:0:0:0:0:0 (::) indicates that an IP address has not yet been configured on an interface. The host uses this address when it is first initializing and has not yet learned its own address.
- **Loopback address:** The loopback address is 0:0:0:0:0:0:0:1 (::1). This special address is also called the *local host address* and is used by the device to send a packet to itself.

The IETF and IANA have reserved various address blocks for special purposes and features. [Table 18-10](#) includes the special-purpose IP address registries outlined in RFC 6890, along with the RFC that documents the purpose of the reservation.

Address Block	Name	Routed on IPv6 Internet	RFC
::/128	Unspecified	No	RFC 4291
::1/128	Loopback	No	RFC 4291
::ffff:0:0:/96	IPv4-Mapped Address	No	RFC 4291
100::64	Discard-Only Address Block	No	RFC 6666
2000::/3	Current Global Unicast Delegation	Yes	RFC 4291
2001::/23	IETF Protocol Assignments	No	RFC 2928
2001::/32	Teredo Tunneling	Yes	RFC 4380
2001:10::/28	Orchid	No	RFC 4843
2001:2::/48	Benchmarking	No	RFC 5180
2001:db8::/32	Documentation	No	RFC 3849
2002::/16	6to4	Yes	RFC 3056
64:ff9b::/96	IPv4-IPv6 Translate	Yes	RFC 6042
fc00::/7	Unique Local Unicast	No	RFC 4193
fe80::/10	Link-Local Unicast	No	RFC 4291
ff00::/8	Multicast	Yes	RFC 4291

**Table 18-10** IPv6 Address Blocks

## NEIGHBOR DISCOVERY PROTOCOL

The Neighbor Discovery Protocol (NDP) is a mechanism for nodes to learn about each other. The protocol is specifically responsible for discovering the prefix and router information for the connected network, determining the link-layer address of a neighbor, and for tracking the reachability state of the neighboring device.

Neighbor discovery is responsible for solving all of the following problems:

- Router discovery
- Prefix discovery
- Parameter discovery
- Address autoconfiguration
- Address resolution
- Next-hop determination
- Neighbor unreachability detection
- Duplicate address detection
- Redirect

Neighbor discovery uses ICMPv6 and solicited-node multicast addresses for basic communication. [Table 18-11](#) describes the five different types of ICMPv6 packets that are used by the NDP.

Message Type	Source Address	Destination Address
Router solicitation (ICMP type 133)	Local interface address or The unspecified address (::) if an address has not yet been assigned	All-routers multicast address (ff02::2)
Router advertisement (ICMP type 134)	Link-local address	All-nodes multicast address (ff02::1)
Neighbor solicitation (ICMP type 135)	Local interface IP address or The unspecified address (::) if an address has not yet been assigned	Solicited-node multicast address ff02::1:ffxx:xxxx, where x is the last 24 bits of the target address
Neighbor advertise- ment (ICMP type 136)	Local interface address	All-nodes multicast address (ff02::1) or In response to a neighbor solicita- tion, the unicast source address of the requesting host
Redirect (ICMP type 137)	Link-local address	Unicast source address of the requesting host

**Table 18-11** Neighbor Discovery ICMPv6 Message Types

#### Router, Prefix, and Parameter Discovery.

A router periodically sends out *router advertisement* (RA) messages to the all nodes multicast address ff02::1 on every IPv6-enabled interface. The RA message informs the hosts on the attached network segment that a router is present.

Hosts depend on the RA messages to learn the network parameters, such as what local prefixes are to be used on the network, how long to maintain the prefix per the advertised lifetime, and which router to use as the primary network gateway. In addition, link parameters such as maximum transmission unit (MTU) and packet hop limit count are also derived from the RA messages.

[Figure 18-18](#) displays two routers, R1 and XR1, advertising RA messages to the local attached network 2001:db8::/64.

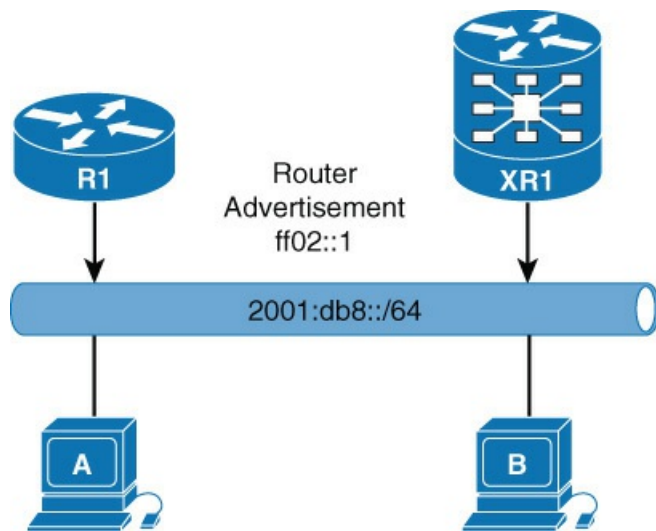


Figure 18-18 Neighbor Discovery Router Advertisement Messages

A new host that does not yet know the network parameters for its local interface does not need to wait for the router to send out the periodic RA message. The host may speed up the process by sending a router solicitation (RS) message to the network requesting the router information. The RS message is sent to the all-routers multicast address `ff02::2` and is sourced from an already configured interface address or the unspecified address (`::`).

Figure 18-19 demonstrates a PC (host C) sending an RS message to the attached network segment `2001:db8::/64`. The two routers on the segment receive the RS message and immediately respond with an RA message.

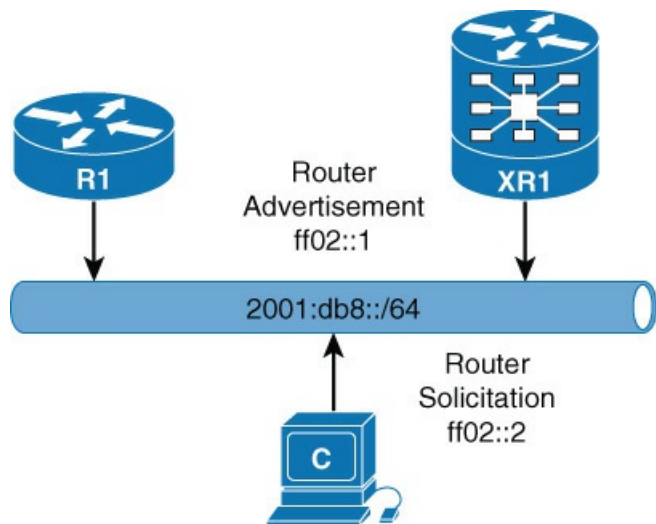


Figure 18-19 Neighbor Solicitation Message and Response

Figure 18-20 displays a packet capture of the RA message from router R1. The RA message is an ICMP type 134 packet that provides information for the router, link parameter, and prefix information for the local link. The host decodes the RA message and determines that `2001:db8::/64` is the local unicast prefix on the directly attached interface. The host then automatically assigns an IP address to its attached interface from the learned prefix using a mechanism called *stateless address autoconfiguration*.

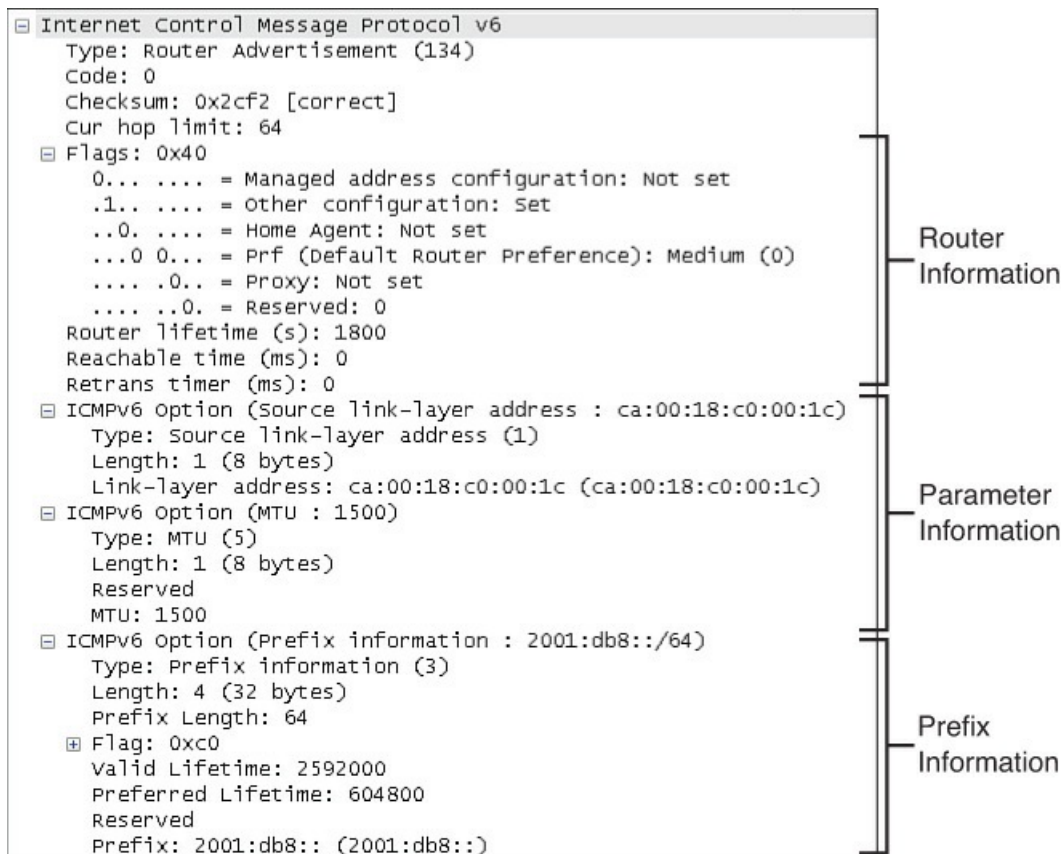


Figure 18-20 Router Advertisement Message Packet Capture

The RA message includes a valid and preferred lifetime value. The preferred lifetime value indicates how long the host should prefer using the address as the primary method for communication. Once the preferred lifetime expires, the host may continue using the address until the valid lifetime expires. The goal of the timers is to allow for easy subnet renumbering.

The two categories of valid and preferred lifetime allow the host enough time to gracefully transition to a new addressing scheme without tearing down active TCP/IP sockets. The default value for IOS and IOS XR nodes are seven days for the preferred lifetime and 30 days for the valid lifetime.

Note

A prefix that has an infinite preferred and valid lifetime will not time out.

The syntax for changing the valid lifetime and preferred lifetime values is **ipv6 nd prefix ipv6-prefix [valid-lifetime-in-seconds] [preferred-lifetime-in-seconds]**. Example 18-9 includes the configuration for setting the prefix valid lifetime to one day and the preferred lifetime to 1 hour.

**Example 18-9** Modifying the Prefix Valid and Preferred Lifetime

[Click here to view code image](#)

#### IOS

```
interface GigabitEthernet2/0
ipv6 nd prefix 2001:DB8::/64 86400 1440
```

#### IOS XR

```
interface GigabitEthernet0/0/0/1
ipv6 nd prefix 2001:db8::/64 86400 1440
```

When multiple routers are present on a subnet, a client PC should attempt to load share between each router per RFC 4311. This is not always true for many client IPv6 OS implementations. In many cases, the first router to advertise on the segment will be chosen as the default gateway for the network. To influence the decision process for the default gateway, the RA packet's router gateway preference flag may be set to high, medium, or low. The IOS command to change the gateway preference is **ipv6 nd router-preference {high | medium | low}**.

[Example 18-10](#) demonstrates changing the default router preference from medium to high.

### Example 18-10 Modifying Default Router Preference to High

[Click here to view code image](#)

#### IOS

```
interface GigabitEthernet2/0
ipv6 nd router-preference high
```

[Example 18-11](#) demonstrates viewing the neighbor discovery parameters on an interface with the command **show ipv6 interface interface-type interface-number**. The output validates that the IOS router R1's default router preference has been configured for high. Notice that the XR router XR1 does not have a Router Preference field.

### Example 18-11 IPv6 Interface Status

[Click here to view code image](#)

```
R1#show ipv6 interface GigabitEthernet 2/0
! Output omitted for brevity
ND DAD is enabled, number of DAD attempts: 1
ND reachable time is 30000 milliseconds (using 30000)
ND advertised reachable time is 0 (unspecified)
ND advertised retransmit interval is 2000 milliseconds
ND router advertisements are sent every 200 seconds
ND router advertisements live for 1800 seconds
ND advertised default router preference is High
Hosts use stateless autoconfig for addresses.
```

```
RP/0/0/CPU0:XR1#show ipv6 interface GigabitEthernet 0/0/0/2
! Output omitted for brevity
ND DAD is enabled, number of DAD attempts 1
ND reachable time is 0 milliseconds
ND cache entry limit is 1000000000
ND advertised retransmit interval is 0 milliseconds
ND router advertisements are sent every 160 to 240 seconds
ND router advertisements live for 1800 seconds
! IOS XR does not have router preference configuration option
Hosts use stateless autoconfig for addresses.
```

Modifying the default router lifetime value in the RA message is an alternative way to influence which router is chosen as the default gateway. The default RA lifetime value is 30 minutes (1800 seconds). A router with a lifetime value of 0 is not eligible for gateway selection.

**Example 18-12** demonstrates manually setting the default RA lifetime to 0 minutes on R1 and 1 hour on XR1. Using these settings XR1 will be elected the local gateway for the hosts. The command **ipv6 nd ra lifetime seconds** modifies the lifetime in IOS and **ipv6 nd ra-lifetime seconds** modifies the lifetime in IOS XR.

### **Example 18-12** *Modifying the Router Lifetime to Influence Gateway Election*

[Click here to view code image](#)

```
IOS
interface GigabitEthernet2/0
ipv6 nd ra lifetime 0
```

```
IOS XR
interface GigabitEthernet0/0/0/1
ipv6 nd ra-lifetime 3600
```

**Example 18-13** demonstrates how to use the command **show ipv6 interface interface-type interface-number** to validate the advertised router lifetime values. Notice that R1 is no longer advertising itself as a default router candidate.

### **Example 18-13** *IPv6 RA Lifetime*

[Click here to view code image](#)

```

R1#show ipv6 interface GigabitEthernet 2/0
! Output omitted for brevity
ND DAD is enabled, number of DAD attempts: 1
ND reachable time is 30000 milliseconds (using 30000)
ND advertised reachable time is 0 (unspecified)
ND advertised retransmit interval is 0 (unspecified)
ND router advertisements are sent every 200 seconds
ND is not advertising as a default router
Hosts use stateless autoconfig for addresses.

RP/0/0/CPU0:XR1#show ipv6 interface GigabitEthernet 0/0/0/2
! Output omitted for brevity
ND DAD is enabled, number of DAD attempts 1
ND reachable time is 0 milliseconds
ND cache entry limit is 1000000000
ND advertised retransmit interval is 0 milliseconds
ND router advertisements are sent every 160 to 240 seconds
ND router advertisements live for 3600 seconds
Hosts use stateless autoconfig for addresses.

```

A network administrator may also completely disable RA messages on a per-prefix or per-interface basis. The IOS command to suppress advertisement on a prefix is **ipv6 nd prefix ipv6-prefix/prefix-length [no-advertise]**, and the XR command is **ipv6 nd prefix ipv6-prefix/prefix-length [no-adv]**. The IOS command **ipv6 nd ra suppress** and the IOS XR command **ipv6 nd suppress-ra** may be used to suppress RA messages for the entire interface.

Example 18-14 demonstrates disabling RA advertisement messages on an interface.

### Example 18-14 Disabling RA Messages

[Click here to view code image](#)

```

IOS
interface GigabitEthernet2/0
ipv6 nd ra suppress

```

```

IOS XR
interface GigabitEthernet0/0/0/2
ipv6 nd suppress-ra

```

Example 18-15 validates that the interfaces are not advertising RA messages.

### Example 18-15 Disabling RA Messages

[Click here to view code image](#)



```
R1#show ipv6 interface GigabitEthernet 2/0
! Output omitted for brevity
ND DAD is enabled, number of DAD attempts: 1
ND reachable time is 30000 milliseconds (using 30000)
ND RAs are suppressed (periodic)
Hosts use stateless autoconfig for addresses.
! ND RA messages are suppressed so there are no RA data in output
```

```
RP/0/0/CPU0:XR1#show ipv6 interface GigabitEthernet 0/0/0/2
! Output omitted for brevity
ND DAD is enabled, number of DAD attempts 1
ND reachable time is 0 milliseconds
ND cache entry limit is 1000000000
ND advertised retransmit interval is 0 milliseconds
Hosts use stateless autoconfig for addresses.
```

#### Note

It is strongly recommend that a first-hop redundancy protocol (FHRP) be used when multiple gateway routers are present, instead of relying on client PCs performing the gateway selection process. FHRPs such as Hot Standby Router Protocol Version 6 (HSRPV6) or Gateway Load Balancing Protocol Version 6 (GLBPV6) ensure that the correct router is chosen as the primary gateway and provide a faster failover and recovery time for hosts, as compared with the native neighbor discovery protocol mechanisms.

## Redirect

A router may send a neighbor discovery ICMPv6 (137) redirect message to a host to notify it that a more preferred gateway router exists for the local network. Figure 18-21 illustrates a router notifying a host via a redirect message that a more preferred gateway exists on the local network.

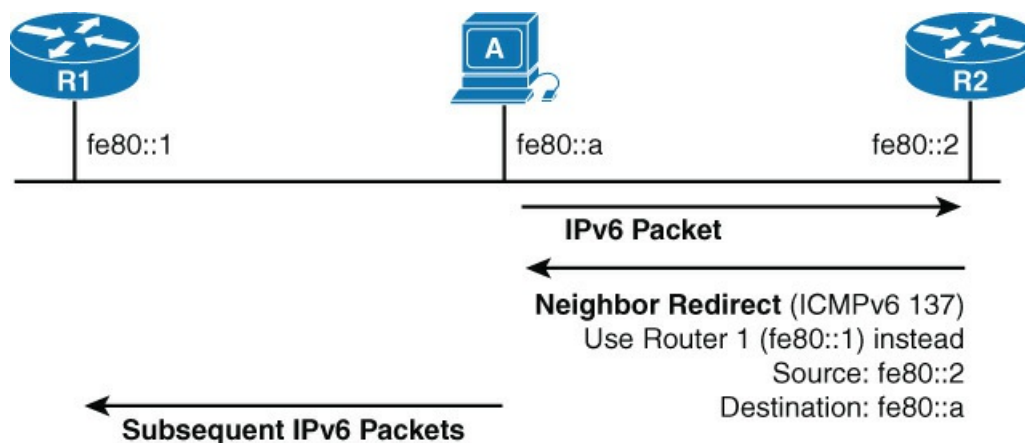


Figure 18-21 Router Redirect

Redirects may also be used to inform a host that the packet's destination address resides on the local link.

The command **show ipv6 interface** may be used to verify whether ICMPv6 redirect is enabled. Example 18-16 demonstrates viewing the ICMPv6 redirect status for an interface. Notice that the redirection is disabled by default for IOS XR.

### Example 18-16 IPv6 Interface Redirect Status

[Click here to view code image](#)

```
R2#show ipv6 interface GigabitEthernet 0/0
! Output omitted for brevity
ICMP error messages limited to one every 100 milliseconds
ICMP redirects are enabled
ICMP unreachable are sent
```

```
RP/0/RSP0/CPU0:XR1#show ipv6 int GigabitEthernet 0/0/0/0
! Output omitted for brevity
MTU is 1514 (1500 is available to IPv6)
ICMP redirects are disabled
ICMP unreachable are enabled
```

The IOS XR command to enable redirects is **ipv6 nd redirects**. IOS has redirects enabled by default. The redirection may be disabled with the interface command **no ipv6 redirects**.

[Example 18-17](#) demonstrates enabling IPv6 redirects on an IOS XR router.

### Example 18-17 Enabling Interface IPv6 Redirects on IOS XR

[Click here to view code image](#)

```
IOS XR
interface GigabitEthernet0/0/0/0
ipv6 nd redirects
```

### IPv6 Stateless Address Autoconfiguration

The default behavior for IPv6 is to automatically assign an IP address to an interface when the protocol stack is first enabled. This built-in mechanism for automatically assigning an IP address to an interface is called *stateless address autoconfiguration* (SLAAC) and is described in detail in RFC 4862.

The other two possible methods for assigning an IP address are to use a Dynamic Host Configuration Protocol Version 6 (DHCPv6) server or manually assigning the address. The key benefit SLAAC provides over the alternative methods is that it does not require any administrative overhead. This is accomplished by assigning an IEEE's 64-bit extended unique identifier (EUI-64) for the interface ID.

### Extended Unique Identifier

The EUI-64 ID is derived from the interface's Ethernet MAC address. EUI-64 IDs are 64 bits long, which is 16 bits longer than a 48-bit MAC address. The process for creating a 64 bit EUI-ID using a reference MAC address of 6c:9c:ed:7f:fb:32 is as follows:

**Step 1.** Split the MAC address into two 24 bit pairs. The first pair indicates the manufacturer's organizational unique identifier (OUI) and the second pair is specific to the network interface card (NIC).

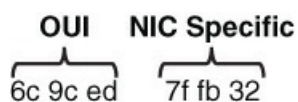


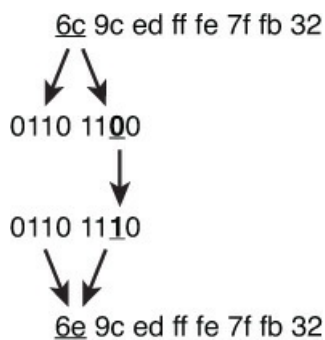
Figure 18-22 MAC Address Format

**Step 2.** Next, a hexadecimal value of fffe is inserted into the middle of the two pairs. Fffe is a special IEEE reserved value that indicates that the EUI-ID was generated from a MAC address.



**Figure 18-23** Split MAC Address and Insert fffe in the Middle

**Step 3.** The seventh bit of the MAC address, also known as the Universal/Local bit (U/L), is flipped. This bit indicates whether the interface is universally or locally administered. A value of 1 indicates locally administered and a value of 0 indicates globally unique (OUI enforced). IEEE-assigned MAC addresses always have this bit set to 0. Setting the bit to 1 indicates that the manufacturer’s address has been modified.



**Figure 18-24** Step 3: Invert the U/L Bit (Seventh Bit)

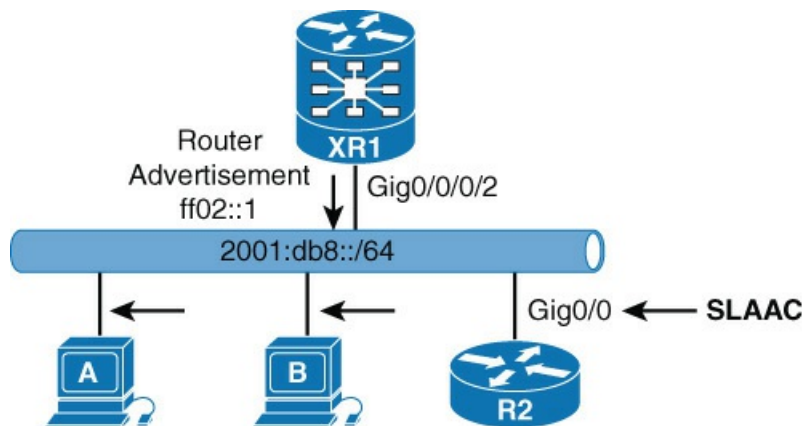
The MAC address of 6c:9c:ed:7f:fb:32 produces the 64 bit EUI-ID 6e9c:edff:fe7f:fb32.

Note

EUI-64 identifiers are a fixed 64 bits in length and may be used only with a /64 subnet. This rule is true regardless of the link type, including point-to-point links, where only two addresses are used.

**SLAAC Router Configuration**

In **Figure 18-25**, the router XR1’s interface GigabitEthernet0/0/0/2 is configured to use an EUI-64 IPv6 address with a prefix of 2001:db8::/64. Router R2’s interface Gig0/0 is configured for SLAAC and will assign an EUI-64 IPv6 address to the interface based on the prefix information in the received RA messages from XR1.



**Figure 18-25** SLAAC Network

**Example 18-18** displays the interface MAC address for XR1’s interface GigabitEthernet 0/0/0/2.

## Example 18-18 Ethernet Interface MAC Address

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show interfaces GigabitEthernet 0/0/0/2
! Output omitted for brevity
GigabitEthernet0/0/0/2 is up, line protocol is up
Hardware is GigabitEthernet, address is 6c9c.ed7f.fb32 (bia 6c9c.ed7f.fb32)
```

Example 18-19 demonstrates how to generate a EUI-64 IPv6 address on an interface with the command **ipv6 address ipv6-prefix/prefix-length [eui-64]**.

## Example 18-19 Assigning an EUI-64 IPv6 Address

[Click here to view code image](#)

```
IOS
interface GigabitEthernet2/0
ipv6 address 2001:DB8::/64 eui-64
```

```
IOS XR
interface GigabitEthernet0/0/0/2
ipv6 address 2001:db8::/64 eui-64
```

Example 18-20 displays the addressing on interface GigabitEthernet0/0/0/2 for router XR1. Notice that both the link-local and globally routable addresses are using EUI-64 format. The two addresses are on different subnets so there is not a conflict due to having the same interface ID 6e9c:edff:fe7f:fb32.

## Example 18-20 Viewing the EUI-64 IPv6 Address

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show ipv6 interface brief
! Output omitted for brevity
GigabitEthernet0/0/0/2 [Up/Up]
fe80::6e9c:edff:fe7f:fb32
2001:db8::6e9c:edff:fe7f:fb32
```

Example 18-21 displays router R2's interface configuration. The command **ipv6 address autoconfig [default]** enables automatic address configuration on the interface. Including the optional **default** keyword will result in a default route being added to the route table.

## Example 18-21 IOS SLAAC Configuration

[Click here to view code image](#)

## IOS

```
interface GigabitEthernet0/0
ipv6 address autoconfig default
```

**Example 18-22** displays the route table for router R2. Notice the default (::/0) route pointing to XR1's link-local address fe80::6e9c:edff:fe7f:fb32 has been added to the Routing Information Base (RIB) by the Neighbor Discovery Protocol (ND in this output).

### **Example 18-22** Neighbor Discovery Default Route

[Click here to view code image](#)

```
R2#show ipv6 route
IPv6 Routing Table - default - 6 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
B - BGP, R - RIP, H - NHRP, I1 - ISIS L1
I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary, D - EIGRP
EX - EIGRP external, ND - ND Default, NDp - ND Prefix, DCE - Destination
NDR - Redirect, O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1
OE2 - OSPF ext 2, ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
a - Application
ND ::/0 [2/0]
via FE80::6E9C:EDFF:FE7F:FB32, GigabitEthernet0/0
NDp 2001:DB8::/64 [2/0]
via GigabitEthernet0/0, directly connected
L 2001:DB8::8A43:E1FF:FE80:FB48/128 [0/0]
via GigabitEthernet0/0, receive
C 2001:DB8:23::/64 [0/0]
via GigabitEthernet0/1, directly connected
L 2001:DB8:23::2/128 [0/0]
via GigabitEthernet0/1, receive
L FF00::/8 [0/0]
via Null0, receive
```

#### Note

As of the writing of this book, stateless autoconfiguration client is supported only on IOS interfaces.

## RA Options for DNS

Initial IPv6 addressing techniques using the default SLAAC mechanisms worked well, except it did not provide essential DNS information to the hosts during the address autoconfiguration process. RFC 6106 addresses these limitations by introducing Recursive DNS Server (RDNSS) and DNS Search List (DNSSL) options for IPv6 hosts using standard SLAAC mechanisms. Using router advertisement neighbor discovery messages the DNS server and search list information are advertised to hosts.

The IOS command to advertise DNS server information in the RA message is **ipv6 nd ra dns server ipv6-address**. [Example 18-21](#) demonstrates how to configure the router to advertise the DNS servers 2001:db8:abcd::1 and 2001:db8:abcd::2 to the IPv6 clients.

### **Example 18-23** IOS SLAAC Configuration

[Click here to view code image](#)

#### IOS

```
interface GigabitEthernet0/0
ipv6 nd ra dns server 2001:db8:abcd::1
ipv6 nd ra dns server 2001:db8:abcd::2
```

#### Note

Support for RDNS was introduced in Cisco IOS XE Release 3.9S and Cisco IOS Release 15.3(2)S and 15.4(1)T. As of the writing of this book, IOS does not support DNSSEC, and IOS XR does not currently include support for RDNS and DNSSEC.

## Stateless DHCPv6

To provide DNS information a router may be configured with stateless DHCPv6 extensions that include the additional DNS and domain information. The router notifies the client to request the additional DHCPv6 extensions by setting the other configuration flag to *on* (1) in the RA messages. The client observes the RA other configuration flag and invokes a DHCPv6 request. If the router is configured as a stateless DHCPv6 server, it will respond with the additional DNS server and domain name information. The router does not maintain DHCPv6 binding states for hosts when configured as a stateless DHCPv6 server.

The client's interface address is assigned using the standard SLAAC modified EUI-64 mechanism.

### IOS Stateless DHCPv6 Configuration

The steps for configuring stateless DHCPv6 on an IOS router are as follows:

#### Step 1. Create a DHCPv6 pool.

The command **ipv6 dhcp pool** *pool-name* is issued to create a DHCPv6 pool and enter the pool configuration mode.

#### Step 2. Define domain parameters.

Parameters such as **dns-server** *ipv6-address* and **domain-name** *domain-name* may be assigned within the IPv6 pool configuration mode.

#### Step 3. Configure the interface.

The DHCPv6 pool is assigned to an interface with the command **ipv6 dhcp server** *pool-name*. The command **ipv6 nd other-config-flag** is used to inform the client machines via neighbor discovery messages that they should perform a DHCPv6 request for the DNS extensions.

Example 18-24 displays a stateless DHCPv6 configuration example.

### Example 18-24 IOS Stateless DHCPv6 Configuration

[Click here to view code image](#)

## IOS

```
ipv6 dhcp pool STATELESS-DHCP
dns-server 2001:db8:abcd::1
dns-server 2001:db8:abcd::2
domain-name cisco.com
!
interface GigabitEthernet0/0
ipv6 enable
ipv6 address 2001:db8::1/64
ipv6 nd other-config-flag
ipv6 dhcp server STATELESS-DHCP
```

### IOS XR Stateless DHCPv6 Configuration

The steps for configuring stateless DHCPv6 on an IOS XR router are as follows:

#### Step 1. Enable the DHCPv6 service.

The DHCPv6 service is enabled globally with the command **dhcp ipv6**.

#### Step 2. Create a server profile.

Once you are in the DHCP IPv6 configuration mode, a profile needs to be created using the command **profile *profile-name* server**.

#### Step 3. Define domain parameters.

Next, the DNS server and domain name list may be configured using the commands **dns-server *ipv6-address*** and **domain-name *domain-name***.

#### Step 4. Associate the interface with the profile.

Finally, the DHCPv6 service is enabled on an interface with the command **interface *interface-type interface-number* server profile *profile-name*** in the DHCPv6 server profile configuration mode.

Example 18-25 displays an IOS XR stateless DHCPv6 configuration. Notice that multiple DNS servers are defined in the same **dns-server** configuration line for server redundancy.

#### Example 18-25 IOS XR Stateless DHCPv6 Configuration

[Click here to view code image](#)

#### **IOS XR**

```
dhcp ipv6
profile STATELESS-DHCP server
dns-server 2001:db8:abcd::1 2001:db8:abcd::2
domain-name cisco.com
!
interface GigabitEthernet0/0/0/0 server profile STATELESS-DHCP
!
interface GigabitEthernet0/0/0/0
ipv6 nd other-config-flag
ipv6 address 2001:db8::1/64
ipv6 enable
```

### **Stateless DHCPv6 Verification**

**Example 18-26** demonstrates viewing the DHCPv6 pool parameter information with the IOS command **show ipv6 dhcp pool pool-name** and the IOS XR command **show dhcp ipv6 server profile name profile-name**. Notice that there are no active clients reported. This will always be the case because stateless DHCPv6 does not track the state of the clients. IPv6 address allocation is provided via the standard SLAAC mechanisms.

### **Example 18-26 Viewing Stateless DHCPv6 Pool Information**

[Click here to view code image](#)

```
R1#show ipv6 dhcp pool STATELESS-DHCP
DHCPv6 pool: STATELESS-DHCP
DNS server: 2001:DB8:ABCD::1
DNS server: 2001:DB8:ABCD::2
Domain name: cisco.com
Active clients: 0
```

```
RP/0/RSP0/CPU0:XR1#show dhcp ipv6 server profile name STATELESS-DHCP
Profile: DHCPv6
DNS Addresses:
2001:db8:abcd::1
2001:db8:abcd::2
Domain Name: cisco.com
Client Lease Time: 0 secs (00:00:00)
Interface References:
GigabitEthernet0/0/0/0
```

### **Stateful DHCPv6, Relay Agent, and Relay Proxy**

Central administration of addresses is possible using stateful DHCPv6. In this model, a server or router running DHCPv6 services is responsible for allocating the addresses and maintaining binding information for the clients.

**Figure 18-26** demonstrates a network where the IOS router R3 is acting as a centralized DHCPv6 server for clients attached to the network segments 2001:db8:100::/64 and 2001:db8:200::/64.



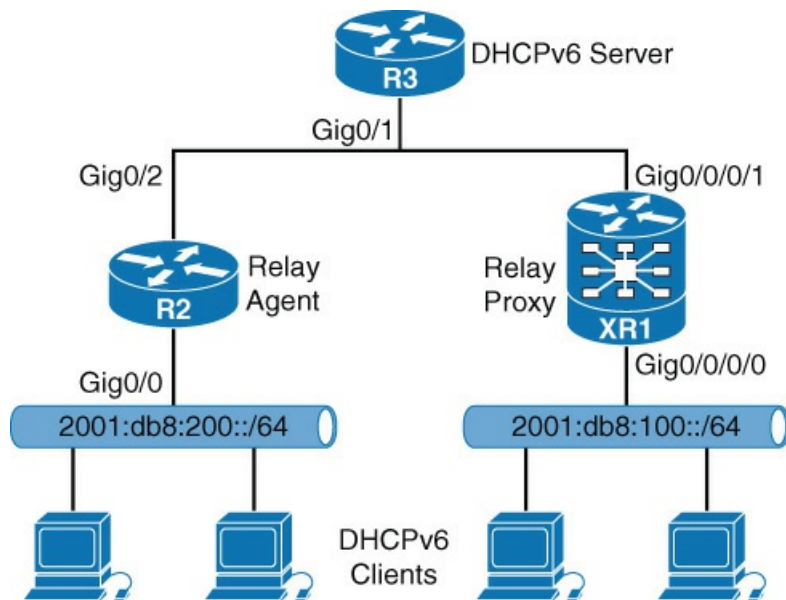


Figure 18-26 DHCPv6 Stateful Topology

The IOS router R2 is the local gateway router for network 2001:db8:200::/64. R2 is responsible for notifying the clients via RA messages that they should use stateful auto configuration (DHCPv6) instead of SLAAC.

An IOS XR router may be configured as either a relay agent or relay proxy. In the example, XR1 is configured as a DHCPv6 relay proxy. A DHCPv6 relay proxy router is more sophisticated than a relay agent. As a proxy router, XR1 responds to the client DHCPv6 requests directly instead of transparently forwarding the packets to the DHCPv6 server. The benefit of this approach is that the server details are concealed from the clients, which isolates and helps protect the servers from potential denial-of-service (DoS) attacks.

Note

DHCPv6 does not provide the default gateway information for the clients. Instead, the clients discover the default gateway information from the neighbor discovery router advertisement messages.

### IOS Relay Agent Configuration

The steps for configuring a relay agent on an IOS router are as follows:

#### Step 1. Set IPv6 the managed address configuration flag.

The interface parameter command `ipv6 nd managed-config-flag` sets the managed address configuration flag in the RA ICMP messages. The flag option notifies the clients to request an address using DHCPv6. Upon reception of the RA message, the client will submit a DHCPv6 information request using the well-known DHCP servers and relay agents multicast group address `ff02::1:2`.

#### Step 2. Assign DHCPv6 server forwarding addresses.

The interface parameter command `ipv6 dhcp relay destination ipv6-address` notifies R2 to join the `ff02::1:2` multicast group and listen for the client DHCPv6 information request messages.

Any DHCPv6 request messages received on the configured interface are relayed to the DHCPv6 server R3.

[Example 18-27](#) provides the relevant IPv6 relay agent configuration for router R2's GigabitEthernet0/0 interface.

### **Example 18-27** Router R2: IPv6 Relay Agent Configuration

[Click here to view code image](#)

```
IOS
interface GigabitEthernet0/0
ipv6 address 2001:DB8:200::1/64
ipv6 enable
ipv6 nd managed-config-flag
ipv6 dhcp relay destination 2001:DB8:23::3
```

### **IOS Relay Agent Verification**

[Example 18-28](#) displays the output for the command **show ipv6 dhcp interface**. The output verifies the configured relay interfaces and destination DHCPv6 servers.

### **Example 18-28** Verifying Relay Agent Interfaces

[Click here to view code image](#)

```
R2#show ipv6 dhcp interface
GigabitEthernet0/0 is in relay mode
Relay destinations:
2001:DB8:23::3
```

[Example 18-29](#) displays the IPv6 status for interface GigabitEthernet0/0. Notice that the interface is actively listening for messages on the DHCP multicast group ff02::1:2.

### **Example 18-29** Verifying Relay Agent Interfaces

[Click here to view code image](#)

```
R2#show ipv6 interface GigabitEthernet 0/0
GigabitEthernet0/0 is up, line protocol is up
IPv6 is enabled, link-local address is FE80::8A43:E1FF:FE80:FB48
No Virtual link-local address(es):
Global unicast address(es):
2001:DB8:100::1, subnet is 2001:DB8:100::/64
Joined group address(es):
FF02::1
FF02::2
FF02::1:2
FF02::1:FF00:1
FF02::1:FF80:FB48
! Output omitted for brevity
```

## IOS XR Proxy Agent Configuration

The steps for configuring stateless DHCPv6 on an IOS XR router are as follows:

### Step 1. Enable the DHCPv6 service.

The DHCPv6 service is enabled globally with the command **dhcp ipv6**.

### Step 2. Create a proxy profile.

Once you are in the DHCP IPv6 configuration mode, a proxy profile needs to be created using the command **profile profile-name proxy**.

### Step 3. Assign DHCPv6 server forwarding addresses.

The command **helper-address ipv6-address** is used to specify the DHCPv6 server. Multiple entries may be entered if there are redundant servers.

### Step 4. Associate the interface with the profile.

Enable the DHCPv6 proxy service on the interfaces that will receive the DHCPv6 client requests with the command **interface interface-type interface-number proxy profile-name**.

[Example 18-30](#) provides the relevant IPv6 relay proxy configuration for router XR1.

### Example 18-30 IPv6 Relay Proxy Configuration

[Click here to view code image](#)

```
IOS XR
dhcp ipv6
profile DHCPv6 proxy
helper-address 2001:db8:13::3
!
interface GigabitEthernet0/0/0/0 proxy profile DHCPv6
!
interface GigabitEthernet0/0/0/0
ipv6 nd managed-config-flag
ipv6 address 2001:db8:100::100/64
ipv6 enable
```

## IOS XR Proxy Agent Verification

The commands **show dhcp ipv6 proxy interface**, **show dhcp ipv6 proxy statistics**, and **show dhcp ipv6 proxy binding** may be used to verify the operation of the relay proxy.

[Example 18-31](#) displays the active proxy interfaces for router XR1.

### Example 18-31 IPv6 Relay Proxy Interface Verification

[Click here to view code image](#)

```
RP/1/RSP0/CPU0:XR1#show dhcp ipv6 proxy interface
```

```
Codes: Amb - Ambiguous VLAN, B - Base, R - Relay, P - Proxy,  
SR - Server, S - Snoop, C - Client, INV - Invalid  
CID - Circuit Id, RID - Remote Id, INTF - Interface
```

Interface	Mode	Profile Name	Amb	Lease	Limit
Gi0/0/0/0	P	DHCPv6	No	None	

## IOS Stateful DHCPv6 Server Configuration

In [Figure 18-26](#), IOS router R3 is configured as a simple DHCPv6 server for the two networks, 2001:db8:100::/64 and 2001:db8:200::/64. The steps for configuring stateful DHCPv6 on an IOS router are as follows:

### Step 1. Create a DHCPv6 pool.

The command **ipv6 dhcp pool** *pool-name* is issued to create a DHCPv6 pool and enter the pool configuration mode.

### Step 2. Assign an address prefix.

The address range for each pool is specified with the command **address prefix** *ipv6-prefix*. The optional **link-address** *ipv6-prefix* command allows the DHCPv6 server to correctly identify which LAN the client request is originating from. The downstream relay or proxy agent inserts the locally attached interface IP address as a *link-address* parameter in the DHCP offer. The server evaluates the *link-address* value in the request message to determine which address pool to allocate an address from.

### Step 3. Define pool parameters.

Parameters such as **dns-server** *ipv6-address* and **domain-name** *domain-name* may be assigned within the IPv6 pool configuration mode.

### Step 4. Configure the interface.

The last configuration task is to enable the DHCP server on the receiving interface with the command **ipv6 dhcp server**.

[Example 18-32](#) provides the full DHCPv6 server configuration for R3.

### Example 18-32 DHCPv6 Server Configuration

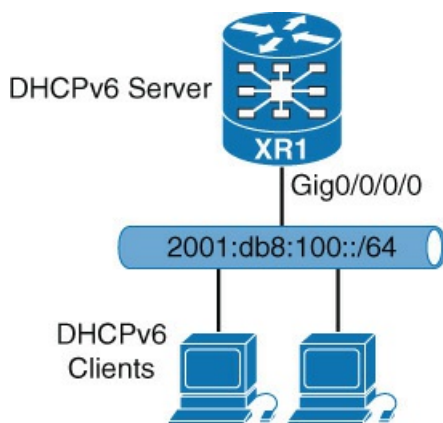
[Click here to view code image](#)

## IOS

```
ipv6 dhcp pool NET100
address prefix 2001:DB8:100::/64
link-address 2001:DB8:100::/64
dns-server 2001:DB8:ABCD::1
dns-server 2001:DB8:ABCD::2
domain-name cisco.com
!
ipv6 dhcp pool NET200
address prefix 2001:DB8:200::/64
link-address 2001:DB8:200::/64
dns-server 2001:DB8:ABCD::1
dns-server 2001:DB8:ABCD::2
domain-name cisco.com
!
interface GigabitEthernet0/1
ipv6 address 2001:DB8:23::3/64
ipv6 enable
ipv6 dhcp server
```

## IOS XR Stateful DHCPv6 Server Configuration

IOS XR routers currently support stateful DHCPv6 only for directly connected networks. [Figure 18-27](#) illustrates an example IOS XR DHCPv6 topology.



**Figure 18-27** IOS XR Stateful DHCPv6 Server

The steps for configuring stateful DHCPv6 on an IOS XR router are as follows:

### Step 1. Create an address pool.

The first configuration task is to define a pool with the command **pool ipv6 pool-name**. Within the pool, the address range is specified with the command **address-range begin-ipv6-address end-ipv6-address**. Currently, the maximum supported pool size is 65,535 addresses.

### Step 2. Enable the DHCPv6 service.

The DHCPv6 service is enabled globally with the command **dhcp ipv6**.

### Step 3. Create a server profile.

Once you are in the DHCP IPv6 configuration mode, a profile needs to be created using the

command **profile** *profile-name* **server**.

#### Step 4. Define domain parameters.

Next, the DNS server and domain name list may be configured using the commands **dns-server** *ipv6-address* and **domain-name** *domain-name*.

#### Step 5. Associated the pool with the server profile.

The pool that was created in step one is attached to the new server profile with the command **address-pool** *profile-name*.

#### Step 6. Associate the interface with the profile.

Finally, the DHCPv6 service is enabled on an interface with the command **interface** *interface-type interface-number* **server profile** *profile-name* in the DHCPv6 server profile configuration mode.

[Example 18-33](#) displays the full DHCPv6 configuration for XR1.

#### Example 18-33 IOS XR DHCPv6 Server Configuration

[Click here to view code image](#)

```
IOS XR
pool vrf default ipv6 NET100
address-range 2001:db8:100::1 2001:db8:100::ffff
!
dhcp ipv6
profile NET100 server
dns-server 2001:db8:abcd::2
domain-name cisco.com
address-pool NET100
!
interface GigabitEthernet0/0/0/0 server profile NET100
```

#### Stateful DHCPv6 Server Verification

The IOS command **show ipv6 dhcp interface** and the IOS XR command **show dhcp ipv6 server interface** may be used to verify that the router is enabled as a DHCPv6 server and is ready to receive client DHCPv6 requests.

[Example 18-34](#) displays the output for each command. Notice that the IOS router R3 may be configured to receive requests for any IPv6 pool on interface GigabitEthernet0/1, while the XR1 router interface GigabitEthernet0/0/0/0 is expecting requests to be directed at the specific pool NET100.

#### Example 18-34 DHCPv6 Server Pool

[Click here to view code image](#)

```
R3#show ipv6 dhcp interface
GigabitEthernet0/1 is in server mode
Using pool:
Preference value: 0
Hint from client: ignored
Rapid-Commit: disabled
```

```
RP/0/RSP0/CPU0:XR1#show dhcp ipv6 server interface
Codes: Amb - Ambiguous VLAN, B - Base, R - Relay, P - Proxy,
SR - Server, S - Snoop, C - Client, INV - Invalid
CID - Circuit Id, RID - Remote Id, INTF - Interface
```

Interface	Mode	Profile Name	Amb Lease Limit
-----	-----	-----	-----
Gi0/0/0/0	SR	NET100	No None

The command IOS **show ipv6 dhcp pool** and the IOS XR command **show pool ipv6** may be used to review the DHCPv6 address pool total utilization at a high level. [Example 18-35](#) displays the output of the two commands. The number of active clients for each pool is highlighted.

### Example 18-35 DHCPv6 Server Pool

[Click here to view code image](#)

```
R3#show ipv6 dhcp pool
DHCPv6 pool: NET100
Address allocation prefix: 2001:DB8:100::/64 valid 172800 preferred 86400 (30 in
use, 0 conflicts)
Link-address prefix: 2001:DB8:100::/64
DNS server: 2001:DB8:ABCD::1
DNS server: 2001:DB8:ABCD::2
Domain name: cisco.com
Active clients: 30
DHCPv6 pool: NET200
Address allocation prefix: 2001:DB8:200::/64 valid 172800 preferred 86400 (17 in
use, 0 conflicts)
Link-address prefix: 2001:DB8:200::/64
DNS server: 2001:DB8:ABCD::1
DNS server: 2001:DB8:ABCD::2
Domain name: cisco.com
Active clients: 17
```

```
RP/0/RSP0/CPU0:XR1#show pool ipv6
```

```
Allocation Summary
```

```
-----  
Used: 5  
Excl: 0  
Free: 131065  
Total: 131070  
Utilization: 0%
```

Pool Name	Pool ID	VRF	Used	Excl	Free	Total
NET100	1	default	5	0	65530	65535

The IOS command **show ipv6 dhcp bindings** and the IOS XR command **show dhcp ipv6 server bindings** may be used to examine the active client bindings. [Example 18-36](#) displays the client information, such as the IPv6 address, lifetime, DHCP unique identifier (DUID), and interface association identifier (IAID).

### Example 18-36 DHCPv6 Server Bindings

[Click here to view code image](#)

```
R3#show ipv6 dhcp binding
```

```
Client: FE80::4433:CCD2:F224:DB31
```

```
DUID: 000100011A56AD75005056A13C8A
```

```
Username : unassigned
```

```
VRF : default
```

```
IA NA: IA ID 0x18005056, T1 43200, T2 69120
```

```
Address: 2001:DB8:100:0:D918:4900:DBB6:F057
```

```
preferred lifetime 86400, valid lifetime 172800
```

```
expires at Jan 05 2014 02:10 AM (172695 seconds)
```

```
! Output omitted for brevity
```

```
RP/1/RSP0/CPU0:XR1#show dhcp ipv6 server binding
```

```
Summary:
```

```
Total number of clients: 5
```

```
DUID : 000300018843e180fb48
```

```
MAC Address: 8843.e180.fb48
```

```
Client Link Local: fe80::8a43:e1ff:fe80:fb48
```

```
Sublabel: 0x0
```

```
IA ID: 0x30001
```

```
STATE: BOUND
```

```
IPv6 Address: 2001:db8:100::3 (GigabitEthernet0/0/0/0)
```

```
lifetime : 86400 secs (1d00h)
```

```
expiration: 86262 secs (23:57:42)
```

```
! Output omitted for brevity
```



## IPv6 Address Resolution and Neighbor Unreachability Detection

The NDP provides the mechanism for determining the link-layer hardware address for a host. Solicited-node multicast ICMPv6 type 135 neighbor solicitation (NS) messages are used to request the link-layer MAC address of another node on the same local network. The targeted device responds back with a unicast ICMPv6 type 136 neighbor advertisement (NA) message that contains the link-layer address information.

The Address Resolution Protocol (ARP) that is used by IPv4 to resolve link-layer addressing is replaced by the NDP in IPv6. [Table 18-12](#) compares the link-layer resolution method used in IPv4 to IPv6.

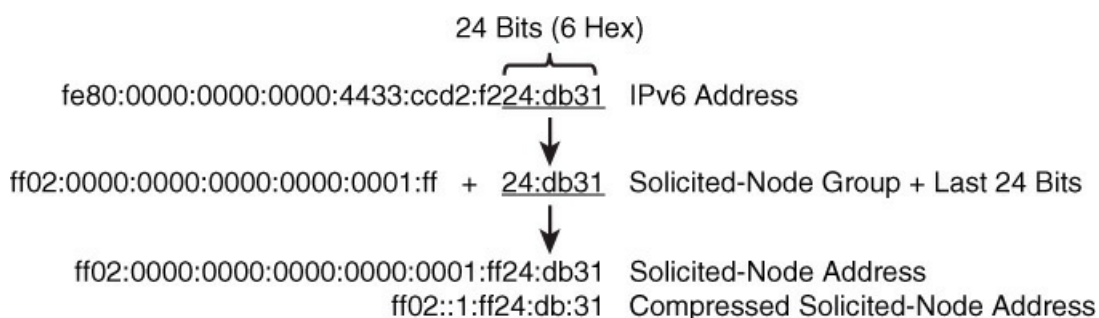
IPv4	IPv6
ARP request	Neighbor solicitation
Broadcast	Solicited node multicast
ARP reply	Neighbor advertisement
Unicast	Unicast

**Table 18-12** IPv4 Versus IPv6 Address Resolution

An IPv6 device determines how to create the solicited-node multicast address by taking the last 24 bits of the target IPv6 unicast address and appending it to the multicast link-local prefix ff02::1:ff00/104. For example, the derived solicited-node multicast address for IPv6 address fe80::4433:ccd2:f224:db31 is ff02::1:ff24:db31.

- **IPv6 Address:** fe80::4433:ccd2:f224:db31
- **Last 24 bits:** 24:db31
- **Solicited-Node group:** ff02::1:ffxx:xxxx
- **Solicited-Node address:** ff02::1:ff24:db31

[Figure 18-28](#) illustrates the process for deriving the solicited-node multicast address.



**Figure 18-28** Solicited-Node Multicast Address

In [Figure 18-29](#), router XR1 wants to communicate with host A (fe80::4433:ccd2:f224:db31). An IPv6 NS message is sent to the host using the solicited-node multicast group address that maps to the destination IP address of host A (ff02::1:ff24:db31). In response to the NS message, host A responds back with an NA message that includes the link-layer address resolution information in the data portion of the packet.

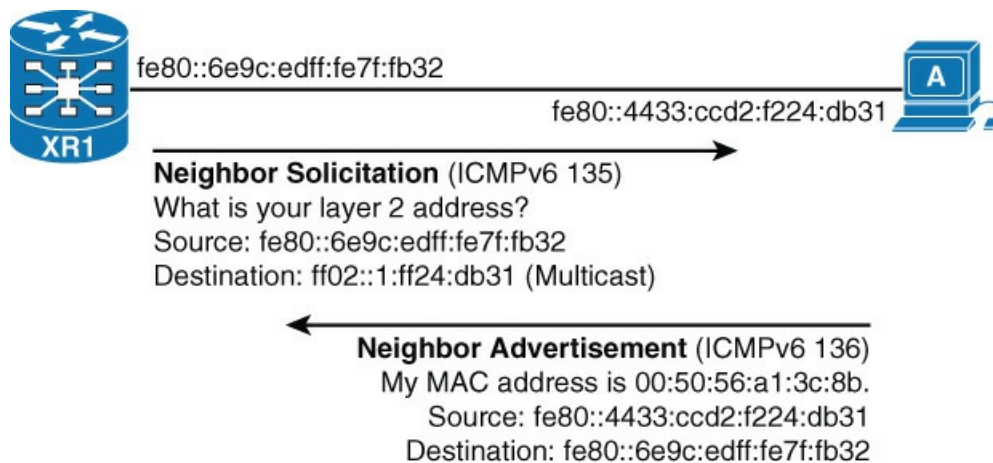


Figure 18-29 IPv6 Address Resolution

Example 18-37 demonstrates how to view the active IPv6 address resolution cache information with the command **show ipv6 neighbors**. One cosmetic difference between the outputs is that the IOS router displays the address age in minutes, whereas the IOS XR router displays the age in seconds. In addition, the IOS XR system includes the multicast adjacency resolution in the output, whereas the IOS system only includes unicast addresses.

### Example 18-37 IPv6 Address Resolution Cache

[Click here to view code image](#)

```
R2#show ipv6 neighbors
```

IPv6 Address	Age	Link-layer Addr	State	Interface
FE80::4433:CCD2:F224:DB31	0	0050.56a1.3c8b	STALE	Gi0/0
FE80::6E9C:EDFF:FE7F:FB32	0	6c9c.ed7f.fb32	REACH	Gi0/0

```
RP/0/0/CPU0:XR1#show ipv6 neighbors
```

IPv6 Address	Age	Link-layer Addr	State	Interface
fe80::4433:ccd2:f224:db31	146	0050.56a1.3c8b	REACH	Gi1/0/0/0
[Mcast adjacency]	-	0000.0000.0000	REACH	Gi1/0/0/0
2001:db8:13::3	3	588d.095e.1552	REACH	Gi1/0/0/1
fe80::5a8d:9ff:fe5e:1552	3	588d.095e.1552	REACH	Gi1/0/0/1
[Mcast adjacency]	-	0000.0000.0000	REACH	Gi1/0/0/1

There are five possible neighbor discovery reachability states:

- 1. Incomplete:** In the reachability state *incomplete*, address resolution is taking place. The router sends an NS message to the destination address and waits for a response. If the router receives a response, the neighbor state changes to reachable.
- 2. Reachable:** As long as there is bidirectional traffic, the neighbor state will always report reachable. When the neighbor stops transmitting packets, a 30-second reachability countdown timer begins. Once the reachability timer expires, the address will be moved to the stale state.
- 3. Stale:** The stale state signifies that address resolution is needed. Stale entries are kept in the ND cache for 4 hours before being removed. Packets destined for a neighbor in the stale state will

still be forwarded by the router. Once forwarded the neighbor unreachability detection (NUD) mechanism begins, and the neighbor reachability state transitions to the delay state.

**4. Delay:** The delay state indicates that the router is waiting for a packet response.

**5. Probe:** If there is no response after 5 seconds, the neighbor reachability state changes to probe and three NS messages are sent for address resolution. If a response is received from the neighbor the state changes to reachable again. If there is no response to the three NS requests, the neighbor entry is deleted.

The command **ipv6 nd reachable-time** *milliseconds* may be used to modify the reachability value. The command **ipv6 nd ns-interval** *milliseconds* modifies the NS timeout value.

Example 18-38 demonstrates how to modify the neighbor reachability time to 15 seconds and the probe timeout to 6 seconds. Three NS messages may be sent to verify neighbor reachability during the probe state. If there is no response, the entry will time out after 6 seconds.

### **Example 18-38** *Modifying Neighbor Solicitation Interval and Neighbor Reachable Time*

[Click here to view code image](#)

#### **IOS**

```
interface GigabitEthernet1/0
ipv6 nd ns-interval 2000
ipv6 nd reachable-time 15000
```

#### **IOS XR**

```
interface GigabitEthernet0/0/0/2
ipv6 nd ns-interval 2000
ipv6 nd reachable-time 15000
```

It may be necessary to clear neighbor address resolutions for troubleshooting purposes. The command **clear ipv6 neighbors** *interface-type interface-number* can be used to manually clear an IPv6 address resolution on both IOS and IOS XR operating systems.

### **Duplicate Address Detection**

To avoid having two identical IP addresses on the same network, an IPv6 device will perform duplicate address detection (DAD) before activating a new address on an interface. To perform DAD, the router sends an NS message to the solicited-node multicast address group corresponding to the new address. The packet is sourced using the unspecified address (::). If an NA message is received back, the system knows that the address is not unique.

#### Note

DAD is performed on both link-local and global unicast autoconfigured addresses. An exception may be made for anycast addressing, which by its nature is supposed to be shared on the network.

Cisco routers follow RFC 4862 guidelines and will log an error message and disable interfaces that have detected duplicate link-local IP addresses. Example 18-39 demonstrates the result of a duplicate IPv6 address on the network.

## Example 18-39 IPv6 Duplicate IP Address Detection Expected Output

[Click here to view code image](#)

```
R2#show log
%IPV6_ND-4-DUPLICATE: Duplicate address FE80::6E9C:EDFF:FE7F:FB32 on
GigabitEthernet0/0
!
R2#show ipv6 interface GigabitEthernet 0/0
GigabitEthernet0/0 is up, line protocol is up
IPv6 is stalled, link-local address is FE80::6E9C:EDFF:FE7F:FB32 [DUP]
No Virtual link-local address(es):
Stateless address autoconfig enabled
No global unicast address is configured
Joined group address(es):
FF02::1
FF02::2
MTU is 1500 bytes
ICMP error messages limited to one every 100 milliseconds
ICMP redirects are enabled
ICMP unreachable are sent
ND DAD is enabled, number of DAD attempts: 1
! Output omitted for brevity
```

```
RP/0/0/CPU0:XR1#show log
ipv6 nd[244]: %IP-IPV6_ND-3-ADDRESS_DUPLICATE : Duplicate address
fe80::6e9c:edff:fe7f:fb32 has been detected

RP/0/0/CPU0:XR1#show ipv6 interface GigabitEthernet 0/0/0/0
GigabitEthernet0/0/0/0 is Up, ipv6 protocol is Down, Vrfid is default (0x60000000)
IPv6 is down (link local duplicate), link-local address is fe80::6e9c:edff:fe7f:fb32
[DUPLICATE] MTU is 1514 (1500 is available to IPv6)
ICMP redirects are disabled
ICMP unreachable are enabled
ND DAD is enabled, number of DAD attempts 1
! Output omitted for brevity
```

Once the source of the duplicate IP address has been found and corrected, the router interface may be safely brought back into service. On an IOS router, the method to restore service is to disable the interface and then enable using the interface commands **shutdown** and **no shutdown**. An IOS XR router may clear the DAD error state with the command **clear ipv6 duplicate addresses interface-type interface-number**. [Example 18-40](#) demonstrates how to restore IPv6 service for the interface.

## Example 18-40 Restoring IPv6 Service

[Click here to view code image](#)

**IOS**

```
interface GigabitEthernet1/0
shut
no shut
```

**IOS XR**

```
clear ipv6 duplicate addresses GigabitEthernet 0/0/0/0
```

## SUMMARY

An IPv6 address is 128 bits in length and is represented in hexadecimal text notation. The enormous address space provided by IPv6 is necessary for allowing the Internet to continue to grow.

An IPv6 address may take three forms: unicast, anycast, or multicast. The IPv4 broadcast address type, along with ARP Layer 3 to Layer 2 address resolution, has been replaced in IPv6 by scope-specific multicast and Neighbor Discovery Protocol. Overall network efficiency is improved in IPv6 because disinterested hosts no longer receive interruptions from broadcasts packets.

IPv6 simplifies network configuration by incorporating multiple IPv4 link-layer discovery protocols such as ARP, ICMP, and DHCP natively into the Neighbor Discovery Protocol operation:

- Router, prefix, and parameter discovery
- Redirect
- Stateless address autoconfiguration (SLAAC)
- IPv6 address resolution
- Neighbor unreachability detection (NUD)
- Duplicate address detection (DAD)

## REFERENCES IN THIS CHAPTER

Cisco. Cisco IOS Software Configuration Guides, <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides, <http://www.cisco.com>

Cisco. IPv6 Addressing White Paper,

[http://www.cisco.com/web/strategy/docs/gov/IPv6\\_WP.pdf](http://www.cisco.com/web/strategy/docs/gov/IPv6_WP.pdf)

IANA. IPv6 Global Unicast Address Assignments,

<http://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xhtml>

IANA. IPv6 Multicast Address Space Registry,

<http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml>

SixXS. IPv6 ULA (Unique Local Address) RFC4193 registration,

<https://www.sixxs.net/tools/grh/ula/>

Huitema, C. and B. Carpenter. RFC 3879, *Deprecating Site Local Addresses*. IETF, <http://www.ietf.org/rfc/rfc3879.txt>, September 2004

Draves, R. and D. Thaler. RFC 4191, *Default Router Preferences and More-Specific Routes*. IETF, <http://www.ietf.org/rfc/rfc4191.txt>, November 2005

Hinden, R. and B. Haberman, RFC 4193, *Unique Local IPv6 Unicast Addresses*. IETF, <http://www.ietf.org/rfc/rfc4193.txt>, October 2005

Hinden, R. and S. Deering. RFC 4291, *IP Version 6 Addressing Architecture*. IETF, <http://www.ietf.org/rfc/rfc4291.txt>, February 2006

Hinden, R. and D. Thaler. RFC 4311, *IPv6 Host-to-Router Load Sharing*. IETF, <http://www.ietf.org/rfc/rfc4311.txt>, November 2005

Narten, T., E. Nordmark, H. Simpson, and H. Soliman. RFC 4861, *Neighbor Discovery in IPv6*. IETF, <http://www.ietf.org/rfc/rfc4861.txt>, September 2007

Thomson, S., T. Narten, and T. Jinmei. RFC 4862, *IPv6 Stateless Address Autoconfiguration*. IETF, <http://www.ietf.org/rfc/rfc4862.txt>, September 2007

Kawamura, S. and M. Kawashima. RFC 5952, *A Recommendation for IPv6 Address Text Representation*. IETF, <http://www.ietf.org/rfc/rfc5952.txt>, August 2010

Jeong, J., S. Park, L. Beloeil, and S. Madanapalli, RFC 6106, *IPv6 Router Advertisement Options for DNS Configuration*. IETF, <http://www.ietf.org/rfc/rfc6106.txt>, November 2010

Narten, T., G. Huston, and L. Roberts, RFC 6177, *IPv6 Address Assignment to End Sites*. IETF, <http://www.ietf.org/rfc/rfc6177.txt>, March 2011

Thaler, D., R. Draves, A. Matsumoto, and T. Chown, RFC 6724, *Default Address Selection for Internet Protocol Version 6 (IPv6)*. IETF, <http://www.ietf.org/rfc/rfc6724.txt>, September 2012

Cotton, M., L. Vegoda, R. Bonica, and B. Haberman, RFC 6890, *Special-Purpose IP Address Registries*. IETF, <http://www.ietf.org/rfc/rfc6890.txt>, April 2013

## Chapter 19. IPv6 Routing

This chapter covers the following topics:

- [Static routing](#)
- [EIGRPv6](#)
- [OSPFv3](#)
- IS-IS
- MP-BGP
- Route redistribution

Conceptually, the process of forwarding IPv6 packets is nearly identical to that of IPv4. The routing protocol topics covered in the previous chapters, such as floating static routes, Open Shortest Path First (OSPF) areas, Enhanced Interior Gateway Routing Protocol (EIGRP) stubs, and Intermediate System-to-Intermediate System (IS-IS) levels, all apply to IPv6 routing. The protocols have simply been redesigned or extended to support the IPv6 address family.

The objective of this chapter is to highlight the key operational differences between IPv4 and IPv6, along with any changes to the configuration syntax and verification commands.

### STATIC ROUTING

Static routing allows for the manual insertion of a route entry into the route table with a default administrative distance (AD) of 1. The IPv4 Routing Information Base (RIB) insertion rules from [Chapter 3](#), “[How a Router Works](#),” such as lowest AD, recursion, and forwarding address reachability, all apply to IPv6 static routing.

One fundamental difference between the IPV4 and IPv6 protocols is that IPv6 does not support the broadcast address type or Address Resolution Protocol (ARP), and therefore route entries for multi-access interfaces require a next-hop forwarding address to ensure proper resolution. Point-to-point (P2) interfaces do not require a forwarding address.

#### Note

A configuration best practice for IPv4 and IPv6 static routes is to use a fully specified static route that includes the outgoing interface, along with the next-hop address. Creating a fully specified static route entry ensures that the route is correctly withdrawn from the RIB if the interface goes down or is disabled, which in turn prevents problems with next-hop reachability recursion.

The global unicast address or link-local address may be used as the next-hop forwarding address in the static route. The outgoing interface is required in the static route entry when using a link-local address as the next-hop.

#### Static Route Configuration

The IOS global configuration command for creating an IPv6 static route is **ipv6 route** *ipv6-prefix/prefix-length* {*ipv6-address* | *interface-type interface-number* [*ipv6-address*]} [*AD*].

In IOS XR, the static route is specified using the command *ipv6-prefix/prefix-length* {*ipv6-*

*address | interface-type interface-number [ipv6-address]}* [AD] under the IPv6 address family, within the router static configuration mode.

[Example 19-1](#) demonstrates how to configure a fully specified static IPv6 route entry for the prefix 2001:db8:0:1::/64 and the default route (::/0) using a link-local forwarding address.

### **Example 19-1** *Static Route Configuration with Link-Local Next Hop*

[Click here to view code image](#)

```
IOS
ipv6 unicast-routing
ipv6 route ::/0 GigabitEthernet0/0 FE80::1
ipv6 route 2001:DB8:0:1::/64 GigabitEthernet0/0 FE80::2
```

```
IOS XR
router static
address-family ipv6 unicast
  ::/0 GigabitEthernet0/0/0/0 fe80::1
  2001:db8:0:2::/64 GigabitEthernet0/0/0/0 fe80::2
```

The IOS command to view the IPv6 route table is **show ipv6 route [static]**, and the IOS XR command is **show route ipv6 [static]**.

[Example 19-2](#) displays the IPv6 static route table.

### **Example 19-2** *IPv6 Static Route Table*

[Click here to view code image](#)

```
R2#show ipv6 route static
! Output omitted for brevity
S   ::/0 [1/0]
    via FE80::1, GigabitEthernet0/0
S   2001:DB8:0:1::/64 [1/0]
    via FE80::2, GigabitEthernet0/0
```

```
RP/0/RSP0/CPU0:XR1#show route ipv6 static
S*  ::/0
    [1/0] via fe80::1, 01:44:41, GigabitEthernet0/0/0/0
S   2001:db8:0:2::/64
    [1/0] via fe80::2, 01:49:11, GigabitEthernet0/0/0/0
```

### **Static Route Reference Chart for IPv6**

[Table 19-1](#) displays the corresponding IPv6 version of the **show** commands that are covered in [Chapter 4](#), “[Static Routing](#).”



OS	Command	Description
IOS	<code>show ipv6 route <i>ipv6-prefix/prefix-length</i></code>	Displays IPv6 route table information
XR	<code>show route ipv6 <i>ipv6-prefix/prefix-length</i></code>	
IOS	<code>show ipv6 cef <i>ipv6-prefix/prefix-length</i> [detail]</code>	Displays the IPv6 hardware programming for a specific IPv6 network prefix
IOS XR	<code>show cef <i>ipv6 ipv6-prefix/prefix-length</i> [detail]</code>	

Table 19-1 show Commands

Table 19-2 lists the corresponding IPv6 version of the configuration commands that are covered in Chapter 4.

OS	Command	Description
IOS	<code>ipv6 route <i>ipv6-prefix/prefix-length</i> {<i>ipv6-address</i>   <i>interface-type interface-number</i> [<i>ipv6-address</i>]} [AD]</code>	Configuration syntax for an IPv6 unicast static route.
XR	<code>router static address-family ipv6 unicast <i>ipv6-prefix/prefix-length</i> {<i>ipv6-address</i>   <i>interface-type interface-number</i> [<i>ipv6-address</i>]} [AD]</code>	* Interface is required when using a link-local next-hop address.

Table 19-2 Configuration Commands

## EIGRPV6

The Enhanced Interior Gateway Routing Protocol supports multiple protocol suites, such as IPv4, IPv6, AppleTalk, and IPX. Protocol-dependent modules (PDMs) provide unique neighbor and topology tables for each protocol. When the IPv6 address family is enabled, the routing protocol is commonly referred to as EIGRPv6.

EIGRP's functional behavior is unchanged between IPv4 and IPv6. The same AD, metrics, timers, and Diffusing Update Algorithm (DUAL) mechanisms are in place to build the route table. Chapter 5, "EIGRP," provides a detailed overview of the EIGRP operation along with its common features. The remainder of this section covers the components of the routing protocol that are unique to IPv6.

### EIGRPv6 Inter-Router Communication

EIGRP packets are identified using the well-known protocol ID 88 for both IPv4 and IPv6. When EIGRPv6 is enabled, the routers communicate with each other using the interface's IPv6 link-local address as the source, and depending on the EIGRP packet type, the destination address may be either a unicast link-local address or the multicast link-local scoped address `ff02::a`.

Table 19-3 displays the source and destination address for the EIGRP packet types.

<b>EIGRP Packet</b>	<b>Source</b>	<b>Destination</b>	<b>Purpose</b>
Hello	Link-local address	ff02::a	Neighbor discovery and keepalive
Acknowledgment	Link-local address	Link-local address	Acknowledges receipt of an update
Query	Link-local address	ff02::a	Request for route information during topology change event
Reply	Link-local address	Link-local address	A response to a query message
Update	Link-local address	Link-local address	Adjacency forming
Update	Link-local address	ff02::a	Topology change

**Table 19-3** *EIGRPv6 Packets*

### **EIGRPv6 Configuration**

There are two methods for configuring IPv6 for EIGRP on IOS routers:

- Classic autonomous system mode
- Named mode

#### **IOS EIGRPv6 Autonomous System Configuration (Classic)**

Classic mode is the original IOS method for enabling IPv6 on EIGRP. In this mode, the routing process is configured using an autonomous system number (ASN).

The steps for configuring EIGRPv6 on an IOS router are as follows:

#### **Step 1. Configure the EIGRPv6 process.**

The routing process is configured with the global configuration command **ipv6 router eigrp as-number**.

#### **Step 2. Assign the router ID.**

The IPv6 address family command **eigrp router-id id** manually assigns the router ID (RID). The RID should be manually assigned to ensure proper operation of the routing process.

The default behavior for EIGRP is to locally assign a RID based on the highest IPv4 loopback address, or if that is not available, the highest IPv4 address. The RID does not need to map to an IPv4 address; the ID value could be any 32-bit unique dotted-decimal identifier. If an IPv4 address is not defined or if the RID is not manually configured, the routing process will not initiate.

#### **Step 3. Enable the process on the interface.**

The interface parameter command **ipv6 eigrp as-number** enables the protocol on the interface.

Nearly all EIGRP IPv4 features are configured in the same manner in IPv6 EIGRP classic mode. The primary difference is that the **ipv6** keyword precedes most IOS commands rather than the **ip** keyword. One noticeable exception is the familiar IPv4 **network** statement within the EIGRP routing configuration mode. The **network** statement does not exist within the EIGRPv6. The protocol must be enabled directly on the interface when using the classic IPv6 EIGRP autonomous system configuration method.

Note

The EIGRPv6 routing process is disabled by default in Cisco IOS Software Version 12.4T. The command **no shut** in the EIGRP router configuration mode will activate the process.

### IOS EIGRPv6 Hierarchical Configuration (Named Mode)

EIGRP named configuration mode is the latest method for configuring the protocol on IOS routers. Named mode provides support for IPv4, IPv6, and VRF (Virtual Routing and Forwarding) all within a single EIGRP instance.

The steps for configuring EIGRP named mode are as follows:

#### Step 1. Configure the EIGRP process.

The EIGRPv6 routing process is configured in global configuration mode with the command **router eigrp process-name**. Unlike classic mode, a name is specified instead of an ASN.

#### Step 2. Define the address family and ASN.

The IPv6 address family and ASN are assigned to the routing process with the command **address-family ipv6 [vrf vrf-name] autonomous-system as-number**.

#### Step 3. Assign the RID.

The IPv6 address family command **eigrp router-id router-id** manually assigns the RID.

Note

The **eigrp upgrade-cli** command was introduced in Cisco IOS XE 3.11S and Cisco IOS 15.4(1)S and may be used to convert classic mode configurations to named mode without causing a network disruption or process restart. The command is applied within the classic IPv6 EIGRP router configuration mode.

EIGRP named mode uses a hierarchical configuration similar to IOS XR to simplify configuration and improve command-line interface (CLI) usability. All the EIGRP-specific interface parameters are configured in the af-interface mode within the IPv6 address family of the named EIGRP process.

Note

Once the IPv6 address family is configured for the EIGRP named process, all the IPv6-enabled interfaces will immediately start participating in routing. To disable the routing process on a specific interface, the interface needs to be shut down (with the **shutdown** command in af-interface configuration mode). Alternatively, routing participation on all interfaces may be disabled by default by applying the **shutdown** command to **af-interface default**.

## IOS XR EIGRPv6 Configuration

IOS XR's IPv6 EIGRP configuration is nearly identical to IPv4. The only difference is that the configuration takes place under the IPv6 address family rather than IPv4.

The steps to configure EIGRPv6 in IOS XR are as follows:

### Step 1. Configure the EIGRP routing process.

The EIGRP routing process and ASN are configured with the command **router eigrp as-number**.

### Step 2. Enable the IPv6 address family.

The IPv6 address family is enabled with the command **address-family ipv6**.

### Step 3. Assign a RID.

The 32-bit RID is manually assigned with the command **router-id router-id**. It is highly recommended that this parameter always be manually assigned to ensure proper operation of the routing process.

The default behavior for EIGRP is to locally assign a RID based on the highest configured IPv4 loopback address, or if that is not available, the highest available IPv4 address. If an IPv4 address is not configured, the system will use the value 0.0.0.0 for an ID.

### Step 4. Enable the process on the interface.

The command **interface interface-type interface-number** enables the protocol on the interface.

## EIGRPv6 Verification

IPv6 uses the same EIGRP verification commands as described [Chapter 5](#). The only modification is that the **ipv6** keyword is included in the command syntax.

[Table 19-4](#) lists the IPv6 version of the **show** commands that are covered in [Chapter 5](#).

OS	Command	Description
IOS	<code>show ipv6 eigrp interface [interface-type interface-number] [detail]</code>	Displays the EIGRPv6 interfaces.
XR	<code>show eigrp ipv6 interface [interface-type interface-number] [detail]</code>	
IOS	<code>show ipv6 eigrp neighbors</code>	Displays the EIGRPv6 neighbors.
IOS	<code>show eigrp ipv6 neighbors</code>	
XR	<code>show ipv6 route eigrp</code>	Displays only EIGRP IPv6 routes in the routing table.
IOS	<code>show route ipv6 eigrp</code>	
XR	<code>show ipv6 eigrp topology [ipv6-prefix/prefix-length]</code>	Displays the current state of the EIGRP topology.
XR	<code>show eigrp ipv6 topology [ipv6-prefix/prefix-length]</code>	Including a specific prefix in the command provides additional calculation information such as total delay, minimum bandwidth, and other K values for path metric calculation.
XR	<code>show ipv6 protocols</code>	Displays the current state of active routing protocol processes.
XR	<code>show protocols ipv6</code>	

Table 19-4 EIGRP Display Commands

Figure 19-1 illustrates a simple EIGRP topology. In the example, EIGRPv6 AS100 is enabled on routers XR1 and R2 to provide connectivity between the networks.

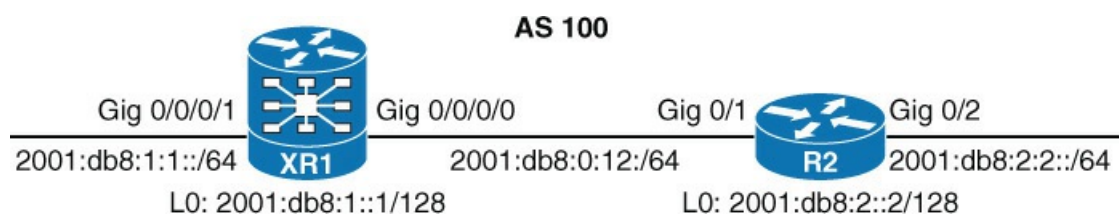


Figure 19-1 Simple EIGRPv6 Topology

Example 19-3 displays the full EIGRPv6 configuration for the topology example. Both EIGRPv6 classic autonomous system and named mode configurations are provided. Notice in IOS classic mode that the routing protocol is applied to each physical interface. In named mode, the protocol is automatically enabled on all interfaces.

### Example 19-3 EIGRPv6 Base Configuration

[Click here to view code image](#)

```

R2 {Classic AS Mode}
ipv6 unicast-routing
ipv6 router eigrp 100
  passive-interface Loopback0
  eigrp router-id 192.168.0.2
!
interface GigabitEthernet0/1
  ipv6 address 2001:DB8:0:12::2/64
  ipv6 address fe80::2 link-local
  ipv6 eigrp 100
!
interface GigabitEthernet0/2
  ipv6 address 2001:DB8:0:2::2/64
  ipv6 address fe80::2 link-local
  ipv6 eigrp 100
!
interface Loopback0
  ipv6 address 2001:DB8:2::2/128
  ipv6 eigrp 100

```

```

R2 {Named Mode}
ipv6 unicast-routing
router eigrp CISCO
!
  address-family ipv6 unicast autonomous-system 100
  !
  af-interface Loopback0
  passive-interface
  exit-af-interface
  !
  topology base
  exit-af-topology
  eigrp router-id 192.168.0.2
  exit-address-family
!

```

```

XR1
router eigrp 100
  address-family ipv6
  router-id 192.168.0.1
  interface Loopback0
  passive-interface
  !
  interface GigabitEthernet0/0/0/0
  !
  interface GigabitEthernet0/0/0/1

```

**Example 19-4** provides verification of the EIGRPv6 neighbor adjacency. Notice that the adjacency uses the link-local addressing.

### **Example 19-4** *EIGRPv6 Neighbor Adjacency*

**[Click here to view code image](#)**

```
R2#show ipv6 eigrp neighbors
EIGRP-IPv6 VR(CISCO) Address-Family Neighbors for AS(100)
H   Address                               Interface           Hold Uptime      SRTT   RTO  Q  Seq
                               (sec)              (ms)              Cnt  Num
0   Link-local address:  Gi0/1              11 00:14:07     21   126  0  5
   FE80::1
```

```
RP/0/0/CPU0:XR1#show eigrp ipv6 neighbors
IPv6-EIGRP Neighbors for AS(100) VRF default
H   Address                               Interface           Hold Uptime      SRTT   RTO  Q  Seq
                               (sec)              (ms)              Cnt  Num
0   Link Local Address:  Gi0/0/0/0         14 00:14:24     1    200  0  3
   fe80::2
```

**Example 19-5** displays route table entries for XR1 and R2. Notice that the IPv6 next-hop forwarding address also uses the link-local address and not the global unicast address of the peer.

### Example 19-5 EIGRPv6 Route Table Entries

[Click here to view code image](#)

```
R2#show ipv6 route eigrp
! Output omitted for brevity
D   2001:DB8:0:1::/64 [90/15360]
   via FE80::1, GigabitEthernet0/1
D   2001:DB8:1::1/128 [90/10752]
   via FE80::1, GigabitEthernet0/1
```

```
RP/0/RSP0/CPU0:XR1#show route ipv6 eigrp
D   2001:db8:0:2::/64
   [90/15360] via fe80::2, 00:42:20, GigabitEthernet0/0/0/0
D   2001:db8:2::2/128
   [90/10880] via fe80::2, 00:05:28, GigabitEthernet0/0/0/0
```

### Summarization

There is no concept of classful or classless routing in IPv6; therefore, autosummarization is not possible. EIGRPv6 summarization for IPv6 is manually configured on a per-interface basis using the same rules as IPv4:

- The summary aggregate prefix is not advertised until a prefix matches it.
- More specific prefixes are suppressed.
- A Null0 route with an AD of 5 is added to the routing table as a loop prevention mechanism.
- IOS routers advertise more specific prefixes using a leak map or with additional **summary-address** statements on the interface.
- IOS XR routers advertise more specific prefixes with additional **summary-address**

statements on the interface.

IOS summarization is configured at the interface level in classic mode using the command **ipv6 summary-address eigrp** *as-number ipv6-prefix/prefix-length* or in named mode with the command **summary-address** *ipv6-prefix/prefix-length* under the af-interface. In IOS XR, IPv6 summarization is applied with the command **summary-address** *ipv6-prefix/prefix-length [AD\_VALUE]* under the interface in router EIGRP configuration mode.

[Example 19-6](#) demonstrates how to configure XR1 to advertise a 2001:db8:1::/48 summary route to R2 and configuring R2 to advertise a 2001:DB8:2::/48 summary route to XR1.

### Example 19-6 EIGRPv6 Summary Configuration

[Click here to view code image](#)

```
R2 {Classic Mode Method}
interface GigabitEthernet0/1
  ipv6 summary-address eigrp 100 2001:DB8:2::/48
```

```
R2 {Named Mode Method}
router eigrp CISCO
!
address-family ipv6 unicast autonomous-system 100
!
af-interface GigabitEthernet0/1
  summary-address 2001:DB8:2::/48
exit-af-interface
!
exit-address-family
```

```
XR1
router eigrp 100
address-family ipv6
interface GigabitEthernet0/0/0/0
  summary-address 2001:db8:1::/48
```

[Example 19-7](#) displays the route table for XR1 and R2. Notice that only the /48 summary prefix is received from the neighbor router and that the more specific /64 and /128 route entries have been suppressed. A Null0 route has been populated on the router for the local /48 summary route advertisement.

### Example 19-7 EIGRPv6 Route Table Entries

[Click here to view code image](#)



```
R2#show ipv6 route eigrp
! Output omitted for brevity
D   2001:DB8:1::/48 [90/2841]
    via FE80::1, GigabitEthernet0/1
D   2001:DB8:2::/48 [5/2816]
    via Null0, directly connected
```

```
RP/0/RSP0/CPU0:XR1#show route ipv6 eigrp
D   2001:db8:1::/48
    [5/1024] via ::, 00:04:15, Null0
D   2001:db8:2::/48
    [90/15360] via fe80::2, 00:00:24, GigabitEthernet0/0/0/0
```

## Default Route

In IOS, a default route can be injected into the EIGRPv6 topology either through route redistribution or through a default route (::/0) summary address at the interface level. When using the summary method, all prefix advertisements, except the ::/0 default route entry, will be suppressed by the router. The redistribution method will not automatically suppress EIGRP network prefixes, but the AD will be set to 170 as an external EIGRP route.

**Example 19-8** demonstrates the two IOS configuration methods for injecting a default route into EIGRPv6: summary address or redistribution.

## Example 19-8 IOS Default Route Injection

[Click here to view code image](#)

```
R2 - Classic (Summary Method)
interface GigabitEthernet0/1
  ipv6 eigrp 100
  ipv6 summary-address eigrp 100 ::/0
```

```
R2 - Classic (Redistribution Method)
ipv6 route ::/0 Null0
ipv6 router eigrp 100
  redistribute static metric 10000 100 255 1 1500
```

```
R2 - Named (Summary Method)
router eigrp CISCO
!
address-family ipv6 unicast autonomous-system 100
  af-interface GigabitEthernet0/1
    summary-address ::/0
```

#### R2 - Named (Redistribution Method)

```
ipv6 route ::/0 Null0
router eigrp CISCO
!
address-family ipv6 unicast autonomous-system 100
topology base
redistribute static metric 10000 100 255 1 1500
```

[Example 19-9](#) demonstrates how to use redistribution on XR1 to inject a default route into EIGRP.

### Example 19-9 IOS XR Default Route Injection

[Click here to view code image](#)

```
XR1
route-policy STATIC-2-EIGRP
  set eigrp-metric 10000 100 255 1 1500
end-policy
!
router static
address-family ipv6 unicast
  ::/0 Null0
!
!
router eigrp 100
address-family ipv6
  router-id 192.168.0.1
  redistribute static route-policy STATIC-2-EIGRP
```

#### Note

Make sure to change the default seed metric of infinity when redistributing routes into EIGRP. EIGRP requires a metric value to perform the diffusing update algorithm. Omitting this step during the configuration will result in the routes not populating the RIB.

### Route Filtering

In IOS, prefix lists are the only method available for matching IPv6 routes in route maps and distribution lists.

[Example 19-10](#) demonstrates how to use a distribute list for filtering the default route ::/0 advertisements from an upstream neighbor connected to interface Gigabit Ethernet 0/1. The associated prefix list BLOCK-DEFAULT first action, sequence 5, is a **deny** statement that filters the exact match default route prefix ::/0. Sequence 10 is a **permit** any match statement that allows a prefix of any length to be received.

### Example 19-10 IOS Distribute List to Filter Default Route

[Click here to view code image](#)

#### R2 {Classic Mode Method}

```
ipv6 router eigrp 100
  distribute-list prefix-list BLOCK-DEFAULT in GigabitEthernet0/1
!
ipv6 prefix-list BLOCK-DEFAULT seq 5 deny ::/0
ipv6 prefix-list BLOCK-DEFAULT seq 10 permit ::/0 le 128
```

#### R2 {Named Mode Method}

```
router eigrp CISCO
!
address-family ipv6 unicast autonomous-system 100
  topology base
    distribute-list prefix-list BLOCK-DEFAULT in GigabitEthernet0/1
  exit-af-topology
exit-address-family
!
ipv6 prefix-list BLOCK-DEFAULT seq 5 deny ::/0
ipv6 prefix-list BLOCK-DEFAULT seq 10 permit ::/0 le 128
```

IOS XR routers do not use distribute lists for filtering EIGRP routes; instead, route policies perform the function.

**Example 19-11** demonstrates how to use a routing protocol language (RPL) inline destination set to drop the ::/0 default route from an upstream neighbor attached to interface Gigabit Ethernet o/o/o/o. The **pass** statement in the RPL allows the reception of all the remaining routes.

#### **Example 19-11** IOS XR RPL Policy to Filter a Default Route

[Click here to view code image](#)

```
XR1
route-policy BLOCK-DEFAULT
  if destination in (::/0) then
    drop
  else
    pass
  endif
end-policy
!
router eigrp 100
  address-family ipv6
  !
  interface GigabitEthernet0/0/0/0
    route-policy BLOCK-DEFAULT in
```

#### **EIGRP Configuration Command Reference Chart for IPv6**

The EIGRP IPv6 configuration for IOS named mode and IOS XR is nearly identical to IPv4. The only modification is the inclusion of the IPv6 address family within the EIGRP configuration process. IOS classic interface commands include the **ipv6** keyword to differentiate between IPv4 and IPv6 address families.

**Table 19-5** lists some common IPv6 configuration commands. Additional EIGRP-specific

commands are covered in Chapter 5.

OS	Command	Description
IOS (Classic)	<code>ipv6 router eigrp as-number</code>	Global configuration command that initializes the EIGRPv6 process for the ASN specified.
IOS (Named)	<code>router eigrp process-name</code> <code>!</code> <code>address-family ipv6 unicast</code> <code>autonomous-system as-number</code>	
XR	<code>router eigrp as-number</code> <code>address-family ipv6</code>	
IOS (Classic)	<code>ipv6 router eigrp as-number</code>	Interface parameter command that enables EIGRPv6 on an interface.
IOS (Named)	<code>af-interface interface-type</code> <code>interface-number</code> <code>[no] shutdown</code>	IOS EIGRPv6 named mode automatically enables the protocol on all interfaces. To disable the protocol, the keyword <code>shutdown</code> can be applied to the target interface.
XR	<code>interface interface-type interface-number</code>	IOS XR routers set the interface under the IPv6 address family within the EIGRP process.
IOS (Classic)	<code>ipv6 authentication mode eigrp</code> <code>as-number md5</code> <code>ipv6 authentication key-chain eigrp</code> <code>as-number key-chain-name</code>	Interface parameter command that enables authentication on an interface and specifies the key chain associated for authentication
IOS (Named)	<code>authentication keychain key-chain-name</code>	IOS named mode and IOS XR routers set the value under the interface within the EIGRP process.
XR		
IOS (Classic)	<code>ipv6 summary-address eigrp</code> <code>as-number ipv6-prefix/prefix-length</code>	Interface parameter command that enables a summary route for EIGRPv6 advertisements out of the associated interface. More-specific prefix updates within this range are suppressed from leaving this interface.
IOS (Named)	<code>summary-address ipv6-prefix/</code> <code>prefix-length [AD_value]</code>	IOS named mode and IOS XR routers set the value under the interface within the EIGRP process.
XR		
IOS (Classic)	<code>distribute-list prefix-list list-name</code> <code>{in   out} interface-number</code>	EIGRPv6 router configuration command that allows for filtering inbound our outbound routes from an update. IPv6 distribute lists require a prefix list.
IOS (Named)	<code>topology base</code> <code>distribute-list prefix-list</code> <code>list-name {in   out} interface-type</code> <code>interface-number</code>	IOS named mode applies the distribute list configuration within the topology base of the IPv6 address family.
XR	<code>route-policy route-policy-name</code> <code>{in   out}</code>	IOS XR routers filter EIGRPv6 routes with a route policy under the target interface within the EIGRP process.

Table 19-5 EIGRPv6 Configuration Commands

## OSPFV3

Open Shortest Path First Protocol Version 3 (OSPFv3) is the latest version of OSPF and includes support for both the IPv4 and IPv6 address families. OSPFv3 is not backward compatible with OSPFv2, but the protocol mechanisms described in Chapter 6, “OSPF,” and Chapter 7, “Advanced OSPF,” are essentially the same.

The primary differences between OSPFv2 and OSPFv3 protocols are as follows:

- **Support for multiple address families:** OSPFv3 supports IPv4 and IPv6 address families.
- **New LSA types:** New LSA types have been created to carry IPv6 prefixes.
- **LSA flooding:** OSPFv3 includes a new Link-State Type field that is used to determine the flooding scope of the LSA, in addition to the handling of unknown LSA types.
- **Packet format:** OSPFv3 runs directly over IPv6, and the number of fields in the packet header have been reduced.
- **Removal of addressing semantics:** The IP prefix information is no longer present in the OSPF packet headers. Instead, it is carried as link-state advertisement (LSA) payload information, making the protocol essentially address family independent, similar to IS-IS. OSPFv3 uses the term *link* rather than *network* because the SPT calculations are per link instead of per subnet.
- **RID:** The RID identifies neighbors, regardless of the network type in OSPFv3. When configuring OSPFv3 on IOS routers, the ID must always be manually assigned in the routing process.
- **Authentication:** Neighbor authentication has been removed from the OSPF protocol and is now performed through IPsec extension headers in the IPv6 packet.
- **Neighbor adjacencies:** OSPFv3 inter-router communication is handled by IPv6 link-local addressing. Neighbors are not automatically detected over non-broadcast multi-access (NBMA) interfaces. The neighbor must be manually specified using the link-local address. IPv6 allows for multiple subnets to be assigned to a single interface, and OSPFv3 allows for neighbor adjacency to form even if the two routers do not share a common subnet.
- **Multiple instances:** OSPFv3 packets include an Instance ID field that may be used to manipulate which routers on a network segment are allowed to form an adjacency.

### Note

RFC 5340 provides in-depth coverage of all the differences between OSPFv2 and OSPFv3.

### OSPFv3 Inter-Router Communication

OSPFv3 packets use protocol ID 89 and routers communicate with each other using the local interface's IPv6 link-local address as the source. Depending on the packet type, the destination address is either a unicast link-local address or the multicast link-local scoped address:

■ **ff02::5**—OSPFv3 AllSPFRouters

■ **ff02::6**—OSPFv3 AllDRouters designated router (DR) router

Every router uses the AllSPFRouters multicast address ff02::5 to send OSPF Hello messages to routers on the same link. The Hello messages are used for neighbor discovery and detecting whether a neighbor relationship is down. The designated router (DR) and backup designated router (BDR) also use this address to send link-state update and flooding acknowledgment messages to all routers.

Non-DR/BDR routers send an update or link-state acknowledgment message to the DR and BDR using the AllDRouters address ff02::6.

OSPFv3 uses the same five packet types and logic as OSPFv2. [Table 19-6](#) displays the packet's name, address, and purpose.

Type	Packet Name	Source	Destination	Purpose
1	Hello	Link-local address	ff02::5 (all routers)	Discover/maintain neighbors
		Link-local address	Link-local address	Initial adjacency forming, immediate Hello
2	Database description	Link-local address	Link-local address	Summarize database contents
3	Link-state request	Link-local address	Link-local address	Database information request
4	Link-state update	Link-local address	Link-local address	Initial adjacency forming, in response to link state request
		Link-local address (from DR)	ff02::5 (all routers)	Database update
		Link-local address (from non-DR)	ff02::6 (DR/BDR)	Database update
5	Link-state acknowledgment	Link-local address	Link-local address	Initial adjacency forming, in response to link state update
		Link-local address (from DR)	ff02::5 (All Routers)	Flooding acknowledgment
		Link-local address (from non-DR)	ff02::6 (DR/BDR)	Flooding acknowledgment

**Table 19-6** OSPFv3 Packet Types

## OSPFv3 Link-State Advertisement

The OSPF link-state database (LSDB) information is organized and advertised differently in Version 3 compared to Version 2. OSPFv3 modifies the structure of the router LSA (Type 1), renames the network summary LSA to interarea prefix LSA, and renames the Autonomous System Boundary Router (ASBR) summary LSA to interarea router LSA. The principal difference is that the router LSA is only responsible for announcing interface parameters such as the interface type (P2P, broadcast, NBMA, point-to-multipoint, and virtual links) and metric (cost).

IP address information is advertised independently by two new LSA types: intra-area prefix LSA and link-local LSA. The OSPF Dijkstra calculation used to determine the shortest path tree (SPT) only examines the router and network LSAs. Advertising the IP address information using new LSA types eliminates the need for OSPF to perform full shortest path first (SPF) tree calculations every time a new address prefix is added or changed on an interface.

Table 19-7 briefly describes each OSPFv3 LSA type.

LS Type	Name	Description
0x2001	Router	Every router generates router LSAs that describe the state and cost of the router's interfaces to the area.
0x2002	Network	A DR generates network LSAs to announce all the routers attached to the link, including itself.
0x2003	Interarea prefix	Area Border Routers (ABRs) generate interarea prefix LSAs to describe routes to IPv6 address prefixes that belong to other areas.
0x2004	Interarea router	ABRs generate interarea router LSAs to announce the address of autonomous system boundary routers in other areas.
0x4005	AS-external	ASBRs advertise AS-external LSAs to announce default routes or routes learned via redistribution from other protocols.
0x2006	Group membership	Deprecated (not currently in use).
0x2007	NSSA	ASBRs that are located in a not so stubby area advertise not-so-stubby area (NSSA) LSAs for routes redistributed into the area.
0x0008	Link	The link LSA maps all the global unicast address prefixes associated with an interface to the link-local interface IP address of the router. The link LSA is shared only between neighbors on the same link.
0x2009	Intra-area prefix	The intra-area-prefix LSA is used to advertise one or more IPv6 prefixes that are associated with a router, stub, or transit network segment.

Table 19-7 OSPFv3 LSA Type

## OSPFv3 LSA Flooding Scope

There were two LSA flooding scopes in OSPFv2: area and autonomous system. OSPFv3 allows for three flooding scopes: link-local scope, area scope, and autonomous system.

The link-local scope is limited to the local link. The area scope contains LSA flooding to the local area. The autonomous system scope floods LSAs throughout the entire OSPF routing domain.

The LS Type field in OSPFv3 has been modified from 8 bits to 16. Figure 19-2 displays the new LS Type field format. The three high-order bits of the new LS Type field allow for the encoding of



flood information. The first bit, U (Unrecognized) bit, indicates how a router should handle an LSA if it is unrecognized. The second and third bits, S (Scope) bits, derive how the LSA should be flooded. The remaining bits of the link-state field indicate the function code of the LSA. For example, a function code of 1 maps to the router LSA, which matches the original OSPFv2 LS type value 1.

Figure 19-2 displays the new LS Type field format.

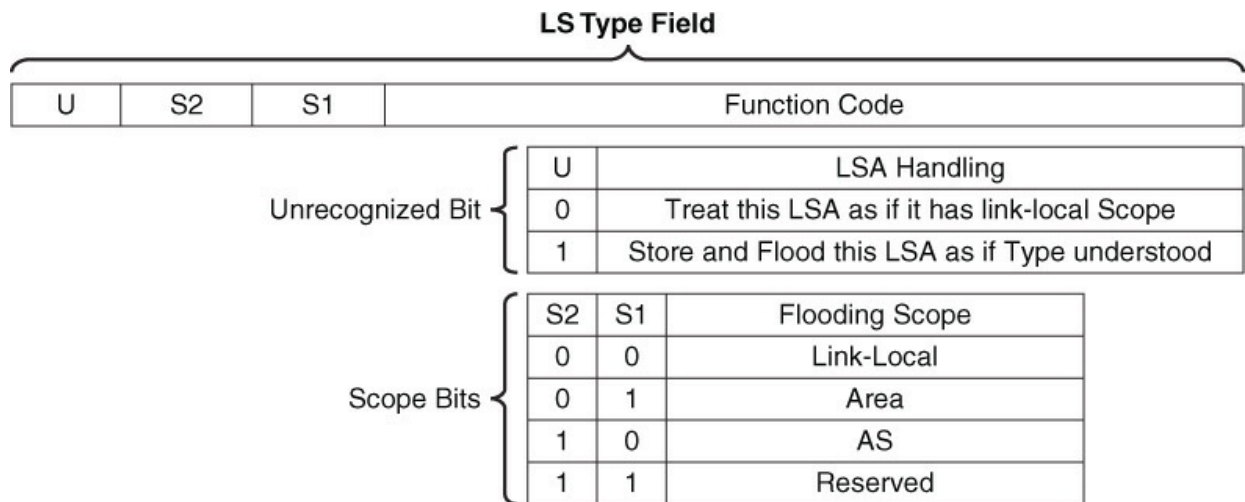


Figure 19-2 LS Type Field

Table 19-8 outlines all nine OSPFv3 LSA types and flooding scopes.

Function Code	LS Type	LSA Name	Flooding Scope
1	0x2001	Router LSA	Area
2	0x2002	Network LSA	Area
3	0x2003	Interarea prefix LSA	Area
4	0x2004	Interarea router LSA	Area
5	0x4005	AS-external LSA	Autonomous system
7	0x2007	NSSA LSA	Area
8	0x0008	Link LSA	Link-local
9	0x2009	Intra-area prefix LSA	Area

Table 19-8 OSPFv3 LSA Type Flooding Scope

### OSPFv3 Configuration

This section introduces the steps required to configure OSPFv3 on a router. A configuration example for a small network topology is provided, along with verification commands. The section concludes with an in-depth examination of the LSDB, which reveals how OSPFv3 creates a view of the network.

### IOS OSPFv3 Configuration

OSPFv3 may be implemented in IOS using either the **ipv6 router ospf** or **router ospfv3** command-line interface (CLI) syntax. The **router ospfv3** syntax was introduced in Version 15.1(3)S and provides additional support for multiple address families. The remainder of this text demonstrates how to configure OSPFv3 using the **router ospfv3** method.

Note

OSPFv3 provides support for the IPv4 address family, but IPv6 link-layer addressing is still required for inter-router communication. The router will prompt an error messages when trying to enable OSPFv3 on an interface that does not have IPv6 already enabled.

To configure IPv6 for OSPFv3 on an IOS router, follow these steps:

**Step 1. Create the routing process.**

The OSPFv3 process is configured with the command **router ospfv3** [*process-id*].

**Step 2. Define the RID.**

The command **router-id** *router-id* assigns a RID to the OSPF process. The RID is a 32-bit value that does not need to match an IPv4 address. It may be any number as long as the value is unique within the OSPF domain.

**Step 3. Assign the address family.**

The address family is defined within the routing process with the command **address-family ipv6 unicast**. The IPv6 address family is enabled by default.

**Step 4. Enable the protocol on an interface.**

The interface command **ospfv3** *process-id* **area** *area-id* **ipv6** enables the protocol and assigns the interface to an area.

**IOS XR OSPFv3 Configuration**

To configure IPv6 for OSPFv3 on an IOS XR router, follow these steps:

**Step 1. Create the routing process.**

The OSPFv3 process is configured with the command **router ospfv3** *process-id*.

**Step 2. Define the RID.**

The 32-bit RID is manually assigned with the command **router-id** *router-id*. It is recommended that this parameter always be manually assigned to ensure proper operation of the routing process.

The default behavior for OSPFv3 is to locally assign a RID based on the highest IPv4 loopback address, or if that is not available, the highest IPv4 address. If an IPv4 address is not configured, the process will not initiate.

**Step 3. Assign the area.**

The command **area** *area-id* defines an area and is used to enter area configuration mode.

**Step 4. Enable the protocol on an interface.**

The command **interface interface-type interface-number** associates the interface to the area.

### OSPFv3 Verification

The **show** verification commands for OSPFv3 are similar to those found in OSPFv2. The only difference is that in IOS the keyword **ospfv3** replaces **ip ospf** and in IOS XR **ospfv3** replaces **ospf** in the command syntax. Table 19-9 lists some of the most common verification commands for reviewing the protocol operation.

OS	Command	Description
IOS XR	<b>show ospfv3 interface [brief]</b>	Displays the OSPFv3 process ID, interface, associated area, interface state, and interface cost
IOS XR	<b>show ospfv3 neighbor</b>	Displays the OSPFv3 neighbor RID, state, priority, dead time, and peer IP address
IOS XR	<b>show ospfv3 database [router   network   interarea   external   adv-router   internal   nssa-external   prefix   self-originated   unknown   database-summary]</b>	Displays the LSAs in the OSPF database
IOS XR	<b>show ospfv3 virtual-links</b>	Displays the status of an OSPF virtual link
IOS XR	<b>show ipv6 route ospf</b> <b>show route ipv6 ospf</b>	Displays only the OSPFv3 routes that are installed into the RIB

Table 19-9 show Commands

Figure 19-3 illustrates a simple OSPFv3 topology. XR2 and R3 are ABR routers for Area 12 and 34. R1 is a member of Area 12 and XR4 is a member of the stub Area 34.

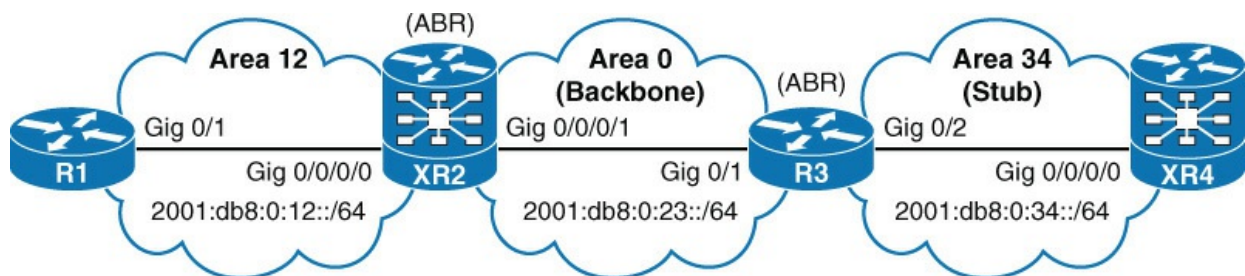


Figure 19-3 OSPFv3 Topology

Example 19-12 displays the OSPFv3-specific configuration for the topology. All routers include a loopback interface, which is set to passive mode to disable routing updates on the software interface. The loopback interface for XR2 and R3 is assigned to Area 0.

### Example 19-12 OSPFv3 Configuration

[Click here to view code image](#)

**R1**

```
interface Loopback0
  ipv6 address 2001:DB8::1/128
  ospfv3 100 ipv6 area 12
!
interface GigabitEthernet0/1
  ipv6 address FE80::1 link-local
  ipv6 address 2001:DB8:0:12::1/64
  ospfv3 100 ipv6 area 12
!
router ospfv3 100
  router-id 192.168.0.1
  !
  address-family ipv6 unicast
    passive-interface Loopback0
  exit-address-family
```

**XR2**

```
interface Loopback0
  ipv6 address 2001:db8::2/128
!
interface GigabitEthernet0/0/0/0
  ipv6 address fe80::2 link-local
  ipv6 address 2001:db8:0:12::2/64
!
interface GigabitEthernet0/0/0/1
  ipv6 address fe80::2 link-local
  ipv6 address 2001:db8:0:23::2/64
!
router ospfv3 100
  router-id 192.168.0.2
  area 0
    interface Loopback0
      passive
    !
    interface GigabitEthernet0/0/0/1
      !
    !
  area 12
    interface GigabitEthernet0/0/0/0
```

**R3**

```
interface Loopback0
ipv6 address 2001:DB8::3/128
 ospfv3 100 ipv6 area 0
!
interface GigabitEthernet0/1
 ipv6 address FE80::3 link-local
 ipv6 address 2001:DB8:0:23::3/64
 ospfv3 100 ipv6 area 0
!
interface GigabitEthernet0/2
 ipv6 address FE80::3 link-local
 ipv6 address 2001:DB8:0:34::3/64
 ospfv3 100 ipv6 area 34
!
router ospfv3 100
 router-id 192.168.0.3
 area 34 stub
!
 address-family ipv6 unicast
  passive-interface Loopback0
 exit-address-family
```

**XR4**

```
interface Loopback0
 ipv6 address 2001:db8::4/128
!
interface GigabitEthernet0/0/0/0
 ipv6 address fe80::4 link-local
 ipv6 address 2001:db8:0:34::4/64
!
interface GigabitEthernet0/0/0/1
 shutdown
!
router ospfv3 100
 router-id 192.168.0.4
 area 34
 stub
 interface Loopback0
  passive
!
 interface GigabitEthernet0/0/0/0
```

**Example 19-13** displays R1's Gigabit Ethernet 0/1 OSPFv3-enabled interface status. Notice how address semantics have been removed. The interface maps to the interface ID value of 3 and not an IP address value like in OSPFv2. In addition, there is some helpful topology information describing the link. The local router is the DR (192.168.0.1), and the adjacent neighbor router is the BDR (192.168.0.2).

**Example 19-13 OSPFv3 Interface**

[Click here to view code image](#)

```

R1#show ospfv3 interface GigabitEthernet0/1
GigabitEthernet0/1 is up, line protocol is up
  Link Local Address FE80::1, Interface ID 3
  Area 12, Process ID 100, Instance ID 0, Router ID 192.168.0.1
  Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 192.168.0.1, local address FE80::1
  Backup Designated router (ID) 192.168.0.2, local address FE80::2
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:01
  Graceful restart helper support enabled
  Index 1/1/1, flood queue length 0
  Next 0x0(0)/0x0(0)/0x0(0)
  Last flood scan length is 0, maximum is 4
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 192.168.0.2 (Backup Designated Router)
  Suppress hello for 0 neighbor(s)

```

The OSPFv3 LSDB creates a shortest path topology tree based on links rather than networks. The next few examples display various LSAs in the OSPFv3 database to help demonstrate this key concept.

The router LSA describes the router's interface state and cost. [Example 19-14](#) displays the output of the command **show ospfv3 database router**. R1 is advertising a router LSA for the local Gigabit Ethernet 0/1 interface (interface ID 3) with a cost of 1. R1 is the DR for the segment, so it populates its own RID in the LSA.

### Example 19-14 OSPFv3 Database Network

[Click here to view code image](#)

```

R1#show ospfv3 database router
      OSPFv3 100 address-family ipv6 (router-id 192.168.0.1)

      Router Link States (Area 12)

      LS age: 763
      Options: (V6-Bit, E-Bit, R-Bit, DC-Bit)
      LS Type: Router Links
      Link State ID: 0
      Advertising Router: 192.168.0.1
      LS Seq Number: 80000002
      Checksum: 0xF9E7
      Length: 40
      Number of Links: 1

      Link connected to: a Transit Network
      Link Metric: 1
      Local Interface ID: 3
      Neighbor (DR) Interface ID: 3
      Neighbor (DR) Router ID: 192.168.0.1
      ! Output omitted for brevity

```

OSPFv3 LSAs include an Options bit field that describes the routers capabilities. [Table 19-10](#) describes the various service options.

Option	Description
V6	The V6-bit indicates that the router participates in IPv6 routing.
E	The E-bit indicates the router is capable of processing external LSAs. A router in a stub area will set the E-bit to clear (0). Neighboring routers will not form an adjacency if they have mismatched E-bit settings.
R	The R-bit indicates that the router will actively participate in forwarding traffic. A router with the R-bit set to clear (0) indicates that it is not to be used as a transit router for forwarding traffic but is still capable of exchanging route information. Clearing the R-bit may be desirable when staging a router or for multihomed hosts that should participate in routing but not forward non-locally addressed packets. The R-bit may be set to clear with the IOS OSPFv3 address family command <i>area area-id no-transit</i> .
DC	The DC-bit is set to indicate that the router is capable of suppressing future Hellos from being sent over the interface. The interface must be configured as a demand circuit for Hello suppression to occur. Demand circuits are typically used on costly low-bandwidth legacy ISDN BRI circuits, which are beyond the scope of this text.
MC	The MC-bit indicates the router is capable of multicast extensions for OSPF (MOSPF). This bit is not used and is only listed for reference. In 2008, RFC 5340 deprecated MOSPF along with the group membership LSA.
N/P	The N-bit indicates that the router supports Type 7 LSAs (NSSA). Neighboring routers will not form an adjacency if they have mismatched N-bit settings. The P-bit setting signals Type 7/5 translation responsibilities. When the bit is set to on (1), the propagation or translation of the Type 7 into a Type 5 LSA should occur on the ABR. If the NSSA ASBR injecting the external route is also an ABR, the P bit will be set to clear (0), indicating that it will perform the translation.

**Table 19-10** OSPFv3 Options Field Bits

[Example 19-15](#) displays a portion of R3's router LSA's LSDB. Notice that the router LSA for Area 0 includes the E-bit, and that the router LSA in Area 34 does not because it is in a stub area.

### **Example 19-15** OSPFv3 Database Network

[Click here to view code image](#)

```

R3#show ospfv3 database router
      OSPFv3 100 address-family ipv6 (router-id 192.168.0.3)

      Router Link States (Area 0)
        LS age: 548
        Options: (V6-Bit, E-Bit, R-Bit, DC-Bit)
        LS Type: Router Links
        ! Output omitted for brevity

      Router Link States (Area 34)
        LS age: 548
        Options: (V6-Bit, R-Bit, DC-Bit)
        LS Type: Router Links
        ! Output omitted for brevity

```

The network LSA describes the known routers on the broadcast interface Gigabit Ethernet 0/1 (interface ID 3). [Example 19-16](#) displays the output of the command **show ospfv3 database network**, which indicates that two routers are present: 192.168.0.1 (R1) and 192.168.0.2 (XR2).

### Example 19-16 OSPFv3 Database Network

[Click here to view code image](#)

```

R1#show ospfv3 database network
      OSPFv3 100 address-family ipv6 (router-id 192.168.0.1)

      Net Link States (Area 12)

        LS age: 1818
        Options: (V6-Bit, E-Bit, R-Bit, DC-Bit)
        LS Type: Network Links
        Link State ID: 3 (Interface ID of Designated Router)
        Advertising Router: 192.168.0.1
        LS Seq Number: 80000002
        Checksum: 0x4B97
        Length: 32
          Attached Router: 192.168.0.1
          Attached Router: 192.168.0.2
        ! Output omitted for brevity

```

The link LSA is responsible for providing actual details for the IPv6 prefixes associated with an interface. [Example 19-17](#) displays the output of the command **show ospfv3 database link**. Notice that the prefix 2001:db8:0:12::/64 is associated with Gigabit Ethernet 0/1 (Interface ID 3) and can be reached using the link-local address fe80::1.

### Example 19-17 OSPFv3 Database Link

[Click here to view code image](#)



```

R1#show ospfv3 database link
      OSPFv3 100 address-family ipv6 (router-id 192.168.0.1)

      Link (Type-8) Link States (Area 12)

LS age: 266
Options: (V6-Bit, E-Bit, R-Bit, DC-Bit)
LS Type: Link-LSA (Interface: GigabitEthernet0/1)
Link State ID: 3 (Interface ID)
Advertising Router: 192.168.0.1
LS Seq Number: 80000003
Checksum: 0xD32
Length: 56
Router Priority: 1
Link Local Address: FE80::1
Number of Prefixes: 1
Prefix Address: 2001:DB8:0:12::
Prefix Length: 64, Options: None
! Output omitted for brevity

```

The router LSA includes bits that describe the router’s role. There is not any prefix information, only information describing the originating router and its connected links. [Table 19-11](#) describes the various bit flags.

Bit	Description
NT	The NT-bit indicates the router is an NSSA border router and will translate LSA Type 7 to LSA Type 5.
x	The x-bit is not currently used. Originally, it was the W-bit for MOSPF, but multi-cast extensions was deprecated in RFC 5340.
V	The V-bit indicates the router is an endpoint for a virtual link.
E	The E-bit indicates the router is an ASBR.
B	The B-bit indicates the router is an ABR.

**Table 19-11** OSPFv3 Router LSA Bits

XR2 has backbone connectivity and is the local ABR for Area 12 in the network topology example. As an ABR, it is responsible for advertising interarea prefix LSAs that describe routes that belong to other areas in the OSPF domain. The command **show ospfv3 database** displays the router’s summary view OSPFv3 database. [Example 19-18](#) displays R1’s database. Notice that XR2’s router LSA bits are set with a B-bit, which indicates that XR2 is an ABR. The advertising RID for all the interarea prefix LSAs originate from 192.168.0.2 (XR2).

### **Example 19-18** Summary View OSPFv3 LSDB

[Click here to view code image](#)

```
R1#show ospfv3 database
```

```
OSPFv3 100 address-family ipv6 (router-id 192.168.0.1)
```

```
Router Link States (Area 12)
```

ADV Router	Age	Seq#	Fragment ID	Link count	Bits
192.168.0.1	744	0x80000004	0	1	None
192.168.0.2	1822	0x80000004	0	1	B

```
Net Link States (Area 12)
```

ADV Router	Age	Seq#	Link ID	Rtr count
192.168.0.1	744	0x80000003	3	2

```
Inter Area Prefix Link States (Area 12)
```

ADV Router	Age	Seq#	Prefix
192.168.0.2	1822	0x80000004	2001:DB8::3/128
192.168.0.2	1822	0x80000004	2001:DB8:0:34::/64
192.168.0.2	1822	0x80000004	2001:DB8::4/128
192.168.0.2	1822	0x80000002	2001:DB8:0:23::/64
192.168.0.2	1822	0x80000002	2001:DB8::2/128

```
Link (Type-8) Link States (Area 12)
```

ADV Router	Age	Seq#	Link ID	Interface
192.168.0.1	997	0x80000003	3	Gi0/1
192.168.0.2	1822	0x80000003	3	Gi0/1

```
Intra Area Prefix Link States (Area 12)
```

ADV Router	Age	Seq#	Link ID	Ref-lstype	Ref-LSID
192.168.0.1	744	0x80000005	0	0x2001	0
192.168.0.1	744	0x80000003	3072	0x2002	3

**Example 19-19** displays the IPv6 route table for routers R1 and XR4. The forwarding address for the routes is the link-local address of the neighboring router. XR4 includes an additional default route entry because it is in a stub area.

### Example 19-19 OSPFv3 Routes Table

[Click here to view code image](#)

```
R1#show ipv6 route ospf
```

```
! Output omitted for brevity
OI 2001:DB8::2/128 [110/1]
    via FE80::2, GigabitEthernet0/1
OI 2001:DB8::3/128 [110/2]
    via FE80::2, GigabitEthernet0/1
OI 2001:DB8::4/128 [110/3]
    via FE80::2, GigabitEthernet0/1
OI 2001:DB8:0:23::/64 [110/2]
    via FE80::2, GigabitEthernet0/1
OI 2001:DB8:0:34::/64 [110/3]
    via FE80::2, GigabitEthernet0/1
```

```

RP/0/0/CPU0:XR4#show route ipv6 ospf
O*IA ::/0
  [110/2] via fe80::3, 00:35:54, GigabitEthernet0/0/0/0
O IA 2001:db8::1/128
  [110/3] via fe80::3, 00:31:43, GigabitEthernet0/0/0/0
O IA 2001:db8::2/128
  [110/2] via fe80::3, 00:31:43, GigabitEthernet0/0/0/0
O IA 2001:db8::3/128
  [110/1] via fe80::3, 00:35:54, GigabitEthernet0/0/0/0
O IA 2001:db8:0:12::/64
  [110/3] via fe80::3, 00:31:43, GigabitEthernet0/0/0/0
O IA 2001:db8:0:23::/64
  [110/2] via fe80::3, 00:35:54, GigabitEthernet0/0/0/0

```

### OSPFv3 Authentication

OSPFv3 does not support neighbor authentication within the protocol itself; instead, the routing protocol utilizes IP Security (IPsec) to provide authentication. IPv6 Authentication Header (AH) or Encapsulated Security Payload (ESP) extension headers may be added to the OSPF packets to provide authentication, integrity, and confidentiality:

- **Authentication Header (AH):** Provides authentication

- **Encapsulating Security Payload (ESP):** Provides authentication and encryption

Figure 19-4 displays the IPv6 IPsec packet format.

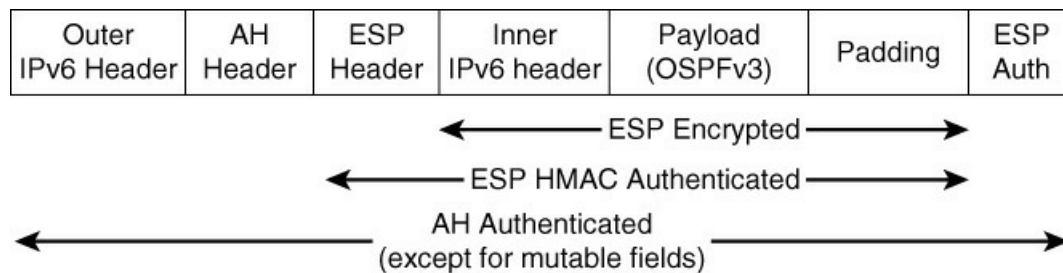


Figure 19-4 IPv6 IPsec Packet Format

#### Note

To configure IPsec, the security software package needs to be enabled on the IOS and IOS XR router.

OSPFv3 authentication supports IPsec AH authentication using command **ospfv3 authentication** or ESP authentication and encryption with the command **ospfv3 encryption**. The configuration may be applied to an interface, virtual link, or an entire area. Area authentication requires that every router in the area perform IPsec authentication in order to form neighbor adjacencies. Interface-level authentication settings preempt area-level settings.

Unlike an IPsec VPN tunnel, OSPFv3 neighbor authentication does not perform Internet Key Exchange (IKE) to negotiate the IPsec security association (SA) values. Therefore, the IPsec Security Parameter Index (SPI) hash algorithm and key must be manually defined when configuring OSPFv3 authentication. IPsec peers cannot reuse the same SPI values. The command **show crypto ipsec sa | include spi** may be used to determine the active IPsec

sessions and currently used SPI values. The SPI verification check is necessary only if virtual private network (VPN) tunnels already exist on the router before enabling OSPFv3 authentication.

The full interface command **ospfv3 encryption {ipsec spi spi esp encryption-algorithm {key-encryption-type key} authentication-algorithm {key-encryption-type key} | null}** encrypts and authenticates the OSPFv3 packet in IOS using ESP. The **null** keyword disables OSPFv3 packet payload encryption and enables only ESP header authentication.

In IOS XR, the OSPFv3 routing process command **encryption {disable | ipsec spi security-parameter esp {3des | aes | [192 | 256] | des | null authentication | {md5 | sha1}}}{clear | password} password** encrypts and authenticates the packets using ESP.

**Example 19-20** demonstrates how to configure encryption and authentication for OSPFv3 packets using ESP. The IOS-XR example demonstrates the router CLI commands as they are entered into the router. The actual key values may look different once applied because IOS XR will not display the cleartext keys in the configuration. The following fabricated values are included in the configuration to establish the IPsec session:

- Security policy index = 500
- Encryption algorithm = aes 192
- Encryption key = 0123456789012345678901234567890123456789012345678901234567
- Authentication algorithm = sha1
- Authentication key = 01234567890123456789012345678901234567890123456789

Note

The fabricated authentication and encryption key values in the example are for demonstration purposes. A real deployment should not use such predictable values. Ideally, the key strings should have high entropy (randomness). Random number generator tools such as openssl rand (random) can be leveraged to create the random strings.

### **Example 19-20** OSPFv3 Interface Authentication and Encryption

**Click here to view code image**

```
IOS
interface GigabitEthernet0/1
ospfv3 encryption ipsec spi 500 esp aes-cbc 192 012345678901234567890123456789012345
678901234567
    sha1 0123456789012345678901234567890123456789
! The ospfv3 encryption rolls over to two lines in the example, but it is only one
! single cli command.
```

**IOS XR**

```
router ospfv3 100
  area 0
    interface GigabitEthernet0/0/0/1
      encryption ipsec spi 500 esp aes 192 01234567890123456789012345678901234
567
      authentication sha1 0123456789012345678901234567890123456789
```

! The ospfv3 encryption rolls over to two lines in the example, but it is entered as one long command.

! Once the configuration is committed the running configuration will display the password encrypted

**Example 19-21** demonstrates how to configure area authentication and encryption using the same IPsec settings.

**Example 19-21 OSPFv3 Area Authentication and Encryption**

[Click here to view code image](#)

**IOS**

```
router ospfv3 100
  area 0 encryption ipsec spi 500 esp aes-cbc 192 01234567890123456789012345678901234
5678901234567
  sha1 0123456789012345678901234567890123456789
```

! The ospfv3 encryption rolls over to two lines in the example, but it is entered as one long command.

! The running configuration will display the password encrypted

**IOS XR**

```
router ospfv3 100
  router-id 100.0.0.1
  area 0
    encryption ipsec spi 500 esp aes 192 012345678901234567890123456789012345
67
    authentication sha1 0123456789012345678901234567890123456789
```

! The ospfv3 encryption rolls over to two lines in the example, but it is entered as one long command.

! The running configuration will display the password encrypted

**Example 19-22** displays the output for the command **show ospfv3 interface** [*interface-type interface-number*]. The **show** command can be used to verify that authentication and encryption is enabled on the interface and that a secure connection has formed with the neighbor.

**Example 19-22 OSPFv3 IPsec Verification**

[Click here to view code image](#)

```
R2#show ospfv3 interface
GigabitEthernet0/1 is up, line protocol is up
  Link Local Address FE80::2, Interface ID 3
  Area 0, Process ID 100, Instance ID 0, Router ID 100.0.0.2
  Network Type BROADCAST, Cost: 1
  AES-CBC-192 encryption SHA-1 auth SPI 500, secure socket UP (errors: 0)
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 100.0.0.2, local address FE80::2
  Backup Designated router (ID) 100.0.0.1, local address FE80::1
! Output omitted for brevity
```

```
RP/0/RSP0/CPU0:XR1#show ospfv3 interface
GigabitEthernet0/0/0/1 is up, line protocol is up, ipsec is up
  Link Local address fe80::1, Interface ID 4
  Area 0, Process ID 100, Instance ID 0, Router ID 100.0.0.1
  Network Type BROADCAST, Cost: 1
  ESP Encryption AES-192, Authentication SHA1, SPI 500
  Transmit Delay is 1 sec, State BDR, Priority 1
  Designated Router (ID) 100.0.0.2, local address fe80::2
  Backup Designated router (ID) 100.0.0.1, local address fe80::1
! Output omitted for brevity
```

### OSPFv3 Multiple Instances

Adjacent routers do not need to be in the same network to form a neighbor relationship because adjacencies are formed using link-local IPv6 addressing. It may be desirable to prevent neighbor adjacencies from forming on links that are shared by multiple routers. OSPFv3 packets include an Instance ID field. Routers will form only neighbor relationships with other routers that have the same instance ID. An instance ID only has local interface significance. The default instance ID value is 0, and an IOS router may set a value from 0 to 31; an IOS XR router may configure a value from 0 to 255.

The interface command **ospfv3 process-id area area-ID [instance instance-id]** sets the instance ID value for an IOS router. For IOS XR routers the instance ID is set under the interface in the router configuration mode with the command **instance instance-id**.

Figure 19-5 illustrates a network where four routers share the same physical network. The networks should be isolated from each other; so in this case, the instance ID 12 is assigned to routers XR1 and R2, while R3 and XR4 use the instance ID 34.

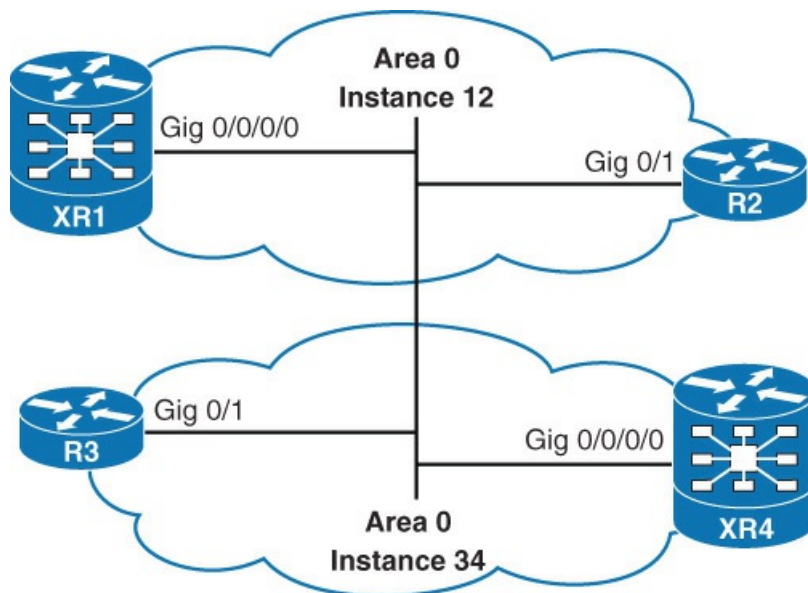


Figure 19-5 OSPFv3 Multiple Instances

Example 19-23 demonstrates how to configure the instance ID to match the topology in Figure 19-5.

### Example 19-23 OSPFv3 Multiple-Instance Configuration

[Click here to view code image](#)

```

XR1
router ospfv3 100
  router-id 192.168.0.1
  area 0
    interface GigabitEthernet0/0/0/0
      instance 12
  
```

```

R2
interface GigabitEthernet0/1
  ospfv3 100 ipv6 area 0 instance 12
!
router ospfv3 100
  router-id 192.168.0.2
  !
  address-family ipv6 unicast
  exit-address-family
  
```

```

R3
interface GigabitEthernet0/1
  ospfv3 100 ipv6 area 0 instance 34
!
router ospfv3 100
  router-id 192.168.0.3
  !
  address-family ipv6 unicast
  exit-address-family
  
```

**XR4**

```

router ospfv3 100
  router-id 192.168.0.4
  area 0
  interface GigabitEthernet0/0/0/0
    instance 34

```

**OSPFv3 Configuration Command Reference Chart for IPv6**

The OSPFv3 IPv6 configuration for IOS XR maintains the same CLI hierarchy as IPv4 OSPFv2.

OSPFv3 on IOS does not include the **network** statement for enabling the protocol. Instead, the protocol is enabled directly on the physical interface. The keyword **ospfv3** replaces **ip ospf** in the configuration syntax.

Table 19-12 lists some common IPv6 configuration commands. Additional OSPF-specific commands are covered in Chapters 6 and 7.

<b>OS</b>	<b>Command</b>	<b>Description</b>
IOS XR	<code>router ospfv3 <i>process-id</i></code>	Global configuration command that initializes the OSPF process.
IOS XR	<code>ospfv3 <i>process-id</i> area <i>area-id</i></code> <code>interface <i>interface-type</i></code> <code><i>interface-number</i></code>	IOS interface configuration command that enables OSPFv3 on a specific interface. IOS XR routers assign the interface under the specific area within the OSPFv3 configuration mode.
IOS	<code>area <i>area-id</i> authentication ipsec spi <i>spi</i> authentication-algorithm</code> <code>[<i>key-encryption-type</i>] <i>key</i></code>	Interface parameter command that enables OSPFv3 packet IPsec AH authentication for a specific interface.
XR	<code>authentication {ipsec spi <i>spi-value</i></code> <code>{md5   sha1} [clear   password]</code> <code><i>password</i>   disable}</code>	IOS XR nodes set the value within the OSPFv3 configuration mode under the area or specific interface.
IOS	<code>ospfv3 authentication {ipsec <i>spi</i></code> <code>{md5   sha1} [<i>key-encryption-type</i> <i>key</i>]</code> <code>  null</code>	Interface parameter command that enables IPsec AH packet authentication for an entire area.
IOS	<code>area <i>area-id</i> encryption ipsec spi <i>spi</i> esp encryption-algorithm</code> <code>[[<i>key-encryption-type</i>] <i>key</i>]</code> <code><i>authentication-algorithm</i></code> <code>[<i>key-encryption-type</i>]</code>	OSPFv3 configuration mode command that enables IPsec ESP packet authentication and encryption for an entire area. IOS XR nodes set the value within the OSPFv3 configuration mode under the area or specific interface.
XR	<code>encryption {disable   ipsec spi</code> <code><i>security-parameter</i> esp {3des   aes  </code> <code>[192   256]   des   null authentication  </code> <code>{md5   sha1 } } {clear   password}</code> <code><i>password</i></code>	



IOS	<code>ospfv3 encryption {ipsec spi spi esp encryption-algorithm {key-encryption-type key} authentication-algorithm {key-encryption-type key}   null}</code>	Interface parameter command that enables IPsec ESP authentication and encryption for OSPFv3 packets for a specific interface.
IOS	<code>ospfv3 process-id area area-ID [instance instance-id]</code>	Interface parameter command that defines an OSPFv3 instance for the specific interface.
XR	<code>instance instance-id</code>	IOS XR nodes set the value within the OSPFv3 process under the target interface.
IOS	<code>area area-id range ipv6-prefix/prefix-length [advertise   not-advertise] [cost metric]</code>	OSPFv3 IPv6 address family configuration mode command that allows for interarea OSPF summarization on an ABR.
XR	<code>range ipv6-prefix/prefix-length [advertise   not-advertise] [cost metric]</code>	IOS XR nodes set the value within the OSPFv3 process under the target area originating the summarization.
IOS XR	<code>summary-address ipv6-prefix/prefix-length</code>	OSPFv3 IPv6 address family configuration mode command that allows external OSPF summarization on an ABR.  IOS XR nodes apply the command in OSPFv3 configuration mode.
IOS XR	<code>default-information originate [always] [metric metric-value] [metric-type type-value]</code>	OSPFv3 IPv6 address family configuration mode command to advertise a default route into OSPF. The <code>always</code> keyword bypasses the check to ensure that the default route is installed in the RIB.  IOS XR nodes apply the command in OSPFv3 configuration mode.

IOS	<code>area <i>area-id</i> stub</code>	OSPFv3 IPv6 address family configuration mode command that configures the area as a stub.  IOS XR configuration is applied under the target area within OSPFv3 configuration mode.
XR	<code>stub</code>	
IOS	<code>area <i>area-id</i> stub no-summary</code>	OSPFv3 IPv6 address family configuration mode command that configures the area as a totally stubby area.  IOS XR configuration is applied under the target area within OSPFv3 configuration mode.
XR	<code>stub no-summary</code>	
IOS	<code>area <i>area-id</i> nssa [default-information-originate]</code>	OSPFv3 IPv6 address family configuration mode command that configures the area as an NSSA. The optional default-information-originate keyword is required on the ABR for a default route to be advertised.  IOS XR configuration is applied under the target area within OSPFv3 configuration mode.
XR	<code>nssa [default-information-originate]</code>	
IOS	<code>area <i>area-id</i> nssa [default-information-originate] no-summary</code>	OSPFv3 IPv6 address family configuration mode command that configures the area as a totally NSSA when configured on an ABR. The optional default-information-originate keyword is required on the ABR for a default route to be advertised.  IOS XR configuration is applied under the target area within OSPFv3 configuration mode.
XR	<code>nssa [default-information-originate] no-summary</code>	

**Table 19-12** Configuration Commands

## INTEGRATED IS-IS FOR IPV6

RFC 5308 extends IS-IS support for IPv6 in essentially the same manner as RFC 1195 did for IPv4. IS-IS Type-Length-Values (TLVs) provide a simple extension of the protocol by adding two TLVs to support IPv6, versus requiring a redesign of the packet format fields like in OSPFv3.

IS-IS handling of IPv6 is different from IPv4 in that it introduces:

- New TLVs
- New multitopology mode option

Chapter 8, “IS-IS,” and Chapter 9, “Advanced IS-IS,” provide a detailed overview of the IS-IS protocol operation and its common features. The remainder of this section covers the protocol operation and router configuration features that are unique for the IPv6 address family.

### IS-IS Inter-Router Communication

One common misconception is that IS-IS protocol data units (PDUs) are encapsulated at Layer 3. Unlike the other routing protocols covered in this book, the packets are encapsulated directly at Layer 2 of the OSI model.

- IS-IS is *not* encapsulated in IPv6 or IPv4.

- IS-IS PDUs are encapsulated directly in the data link layer using the OSI Ethernet type value 0xFEFE.

### IS-IS Type-Length-Value

The IPv6 Reachability TLV (TLV 236) describes the IPv6 prefix reachability information, such as the metric, up/down bit, and whether the route was learned via redistribution. The up/down bit indicates whether the prefix was advertised from a higher level to a lower level. For example, a value of 1 indicates up and that the prefix was advertised from a level 2 to level 1 area. If the prefix was learned via redistribution then the external origination bit is set to 1.

The IPv6 Interface Address TLV (TLV 232) contains IPv6 addressing. Hello PDUs (packets) include the link-local IPv6 addresses assigned to the interface and LSPs packets contain the non-link-local addresses for the intermediate system (router).

RFC 5120 introduced a third IPv6 TLV, Multi Topology Reachable IPv6 Prefix (TLV 237), which is required to support multiple independent topologies and independent SPF calculations.

### IS-IS Topology Modes

Cisco routers provide IS-IS support for IPv6 using one of three topology configuration modes: single topology, multitopology, multitopology transition:

- **Single topology:** In single-topology mode, IPv4 and IPv6 are considered part of the same logical topology table and therefore share a single SPF computation for calculating the best path. To avoid reachability issues, it is critical that all interfaces include both IPv6 and IPv4 addressing.

Single-topology mode is the default IPv6 mode for IOS routers.

- **Multitopology:** Multitopology mode allows for the separation of IPv4 and IPv6 into independent topology tables. Address family specific metrics can be assigned to an interface allowing for diverse network paths and SPF calculations for each protocol. Multitopology mode avoids reachability issues inherent with single-topology mode because a 1:1 correlation of IPv6 and IPv4 addressing is not required.

Multitopology mode is the default IPv6 mode for IOS XR routers.

- **Single-topology transition:** Single-topology transition is a hybrid mode and is targeted at deployments that use legacy equipment that already has IPv6 enabled in single-topology mode but would like to add support for IPv4 without disrupting the IPv6 network that is already in place. Single-topology IS-IS and multitopology IS-IS use different TLVs to exchange information. A router that does not support multitopology mode will not be able to interpret IPv6 multitopology TLVs, which could lead to a network outage during a migration. To ensure this does not occur, the new equipment can be configured using single-topology transition mode. In this mode, the router will send both single and multitopology TLVs. Even though both TLVs are being distributed throughout the network, the routers still operate in single-topology mode and perform one SPF calculation. Once the migration is complete and all the legacy routers are removed, the IS-IS configuration can then be transitioned to multitopology mode.

## IS-IS Configuration

The design goals of the network dictate the topology mode configuration of the routers. Single-topology mode is supported by a wider range of legacy routers and is less resource intensive, whereas multitopology allows for network design flexibility with the use of IPv6-specific metrics and does not require all interfaces to have both IP protocols enabled. Single-topology transition mode may be used in rare instances that IPv6 is already deployed on legacy equipment and the goal is to gracefully transition to a dual-stack environment of IPv4 and IPv6 using multitopology mode. The new equipment can be configured using single-topology transition mode to ensure IPv6 TLV compatibility with the legacy equipment. Once the legacy equipment is decommissioned, the routers can be reconfigured to use multitopology mode without causing network disruption due to TLV interpretation RIB failures.

### Note

Multitopology mode is the recommended IS-IS mode of operation for dual-stack network deployments because it avoids the stringent requirement of congruent IPv4 and IPv6 network topologies.

## IOS Base Configuration

The IOS configuration for IPv6 single-topology mode is nearly identical to the configuration for IPv4. The only difference is that the **ipv6** keyword is used when enabling the protocol at the interface level:

### Step 1. Create the routing process.

The IS-IS process is configured with the command **router isis** [*instance-id*].

### Step 2. Define the IS-IS Network Entity Title (NET).

Every IS-IS router in an area requires a unique NET ID. The NET ID is assigned within the IS-IS routing configuration mode with the command **net network-entity-title**. See [Chapter 8](#) for more information on the NET ID format.

### Step 3. Enable protocol on an interface.

The interface command **ipv6 router isis** [*instance-id*] enables the protocol on the interface.

## IOS XR Base Configuration

Multitopology mode is the default mode of operation when enabling the IPv6 address family within IS-IS. The steps for enabling IPv6 in IS-IS on IOS XR are as follows:

### Step 1. Create the routing process.

The IS-IS process is configured with the command **router isis** [*instance-id*].

### Step 2. Define the IS-IS NET.

Every IS-IS router in an area requires a unique NET ID. The NET ID is assigned within the IS-IS routing configuration mode with the command **net network-entity-title**.

### Step 3. Enable the IPv6 address family.

The IPv6 address family is enabled within the IS-IS routing process with the command **address-family ipv6 unicast**.

#### Step 4. Enable the protocol on an interface.

Within the IS-IS process configuration, enter the IS-IS interface configuration submode with the command **interface** *interface-type interface-number* and assign the IPv6 address family with the command **address-family ipv6 unicast**.

Routers that do not have matching topology modes configured will form IS-IS network adjacencies but will encounter problems processing the IPv6 routes because they will not have matching TLV formats.

Figure 19-6 illustrates two dual-stack routers that are in different topology modes. The routers can form an IS-IS adjacency, but only the IPv4 routes are correctly processed. The two routers are exchanging different types of IPv6 TLVs, so the route entries are not processed properly.

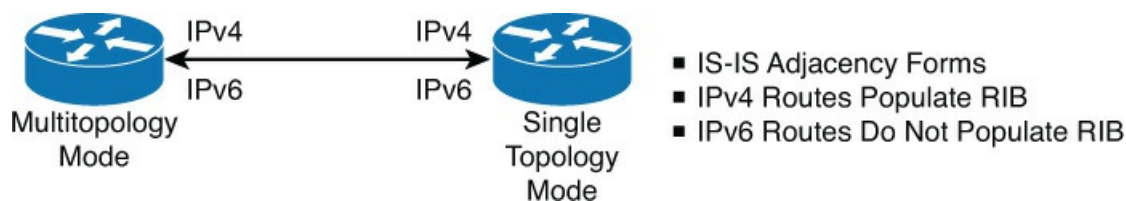


Figure 19-6 Mismatched IS-IS Topology Modes

The default topology modes differ between IOS and IOS XR operating systems:

- **IOS:** Single-topology mode
- **IOS XR:** Multitopology mode

The topology mode needs to be defined to ensure IPv6 route TLV compatibility between IOS and IOS XR.

#### IOS Topology Mode Configuration

The following configuration steps change the default IOS single-topology mode to multitopology:

##### Step 1. Enter IS-IS router configuration mode.

Enter the IS-IS routing process configuration mode with the command **router isis** [*instance-id*].

##### Step 2. Enable metric support.

Enable the router to generate and support new style TLVs with the command **metric-style** [**wide** | **transition**]. The **wide** keyword is required for multitopology deployments. The **transition** keyword sends and accepts both styles of TLVs.

##### Step 3. Specify the IPv6 address family.

Enter the integrated IS-IS-specific IPv6 address family configuration mode with the command **address-family ipv6 unicast**.

##### Step 4. Define the topology mode.

The command **multitopology** [**transition**] enables multitopology or with the optional **transition** keyword multitopology transition mode.

### IOS XR Topology Mode Configuration

The following configuration steps change the IOS XR router's operating mode to single-topology mode:

#### Step 1. Enter IS-IS router configuration mode.

Enter the IS-IS routing process configuration mode with the command **router isis** *instance-id*.

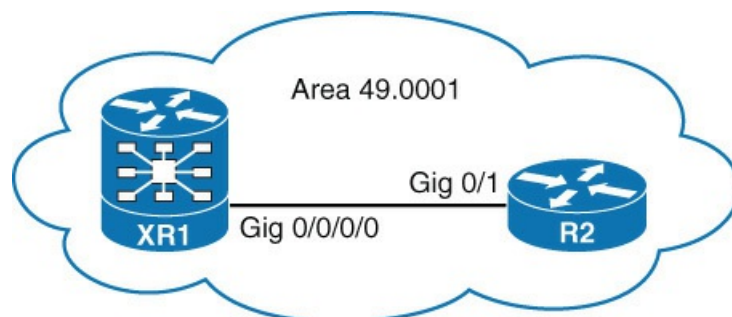
#### Step 2. Specify the IPv6 address family.

Enter the integrated IS-IS-specific IPv6 address family configuration mode with the command **address-family ipv6 unicast**.

#### Step 3. Define the topology mode.

The command **single-topology** changes the IS-IS operating mode to single-topology mode.

[Figure 19-7](#) illustrates two adjacent IS-IS router, XR1 and R2. The two routers are both members of Area 1. IPv6 and IS-IS are enabled on Loopback 0 and the Ethernet interfaces. To ensure IPv6 TLV compatibility, the routers are configured with matching IS-IS topology modes.



**Figure 19-7** Basic IS-IS Adjacency

[Examples 19-24](#), [19-25](#), and [19-26](#) display the various topology modes supported between the routers.

#### Example 19-24 Single-Topology Mode

[Click here to view code image](#)

```
IOS
router isis AREA1
  net 49.0001.0000.0000.0002.00
  !
interface GigabitEthernet0/1
  ipv6 router isis AREA1
  !
interface Loopback0
  ipv6 router isis AREA1
```

**IOS XR**

```
router isis AREA1
 net 49.0001.0000.0000.0001.00
 address-family ipv6 unicast
  single-topology
!
interface Loopback0
 address-family ipv6 unicast
!
!
interface GigabitEthernet0/0/0/0
 address-family ipv6 unicast
```

**Example 19-25 Multitopology Mode**

[Click here to view code image](#)

**IOS**

```
router isis AREA1
 net 49.0001.0000.0000.0002.00
 metric-style wide
!
 address-family ipv6
  multitopology
 exit-address-family
!
interface GigabitEthernet0/1
 ipv6 router isis AREA1
!
interface Loopback0
 ipv6 router isis AREA1
```

**IOS XR**

```
router isis AREA1
 net 49.0001.0000.0000.0001.00
 address-family ipv6 unicast
!
interface Loopback0
 address-family ipv6 unicast
!
!
interface GigabitEthernet0/0/0/0
 address-family ipv6 unicast
```

**Example 19-26 Single-Topology Transition Mode**

[Click here to view code image](#)

#### IOS

```
router isis AREA1
 net 49.0001.0000.0000.0002.00
 metric-style transition
 !
 address-family ipv6
  multitopology transition
 exit-address-family
 !
 interface GigabitEthernet0/1
  ipv6 router isis AREA1
 !
 interface Loopback0
  ipv6 router isis AREA1
```

#### IOS XR

```
router isis AREA1
 net 49.0001.0000.0000.0001.00
 address-family ipv6 unicast
  metric-style transition
  single-topology
 !
 interface Loopback0
  address-family ipv6 unicast
 !
 !
 interface GigabitEthernet0/0/0/0
  address-family ipv6 unicast
```

A router configured for single-topology transition mode will properly share IPv6 routes with a neighbor configured for single or multitopology mode because narrow-and wide-style TLVs are generated and supported to ensure compatibility with all the routers in the topology. The command **show isis database detail** displays the IPv6 routes in the IS-IS topology database.

Example 19-27 demonstrates that router R2 is advertising both IPv6 TLV styles because it is configured to use single-topology transition mode. Notice that the prefix 2001:db8:0:12::/64 is listed twice in the database.

#### **Example 19-27** IS-IS Database

[Click here to view code image](#)



```

R2#show isis database detail
! Output omitted for brevity
R2.00-00          * 0x00000016   0x5F22          1135           0/0/0
  Area Address: 49.0001
  Topology:      IPv4 (0x0)
                 IPv6 (0x2)
  NLPID:        0x8E
  Hostname: R2
  IPv6 Address: 2001:DB8::2
  Metric: 10     IPv6 2001:DB8:0:12::/64
  Metric: 10     IPv6 2001:DB8:0:23::/64
  Metric: 10     IPv6 2001:DB8::2/128
  Metric: 10     IPv6 (MT-IPv6) 2001:DB8:0:12::/64
  Metric: 10     IPv6 (MT-IPv6) 2001:DB8:0:23::/64
  Metric: 10     IPv6 (MT-IPv6) 2001:DB8::2/128
  Metric: 10     IS 0000.0000.0001.01
  Metric: 10     IS-Extended 0000.0000.0001.01
! Output omitted for brevity

```

## Verification

The base operation of IS-IS is covered in [Chapters 8 and 9](#). The only modification necessary to support IPv6 is that the **ipv6** keyword is rather than the **ip** in certain commands. [Table 19-13](#) lists the most common verification commands for reviewing the protocol operation.

OS	Command	Description
IOS	<code>show clns neighbor [detail]</code>	Displays the status of neighbor routers running IS
	<code>show isis neighbors [detail]</code>	
XR	<code>show isis neighbor [detail]</code>	
IOS	<code>show isis database [detail]</code>	Displays the IS-IS LSDB
	XR	
IOS	<code>show clns interface</code>	Displays the active IS-IS interfaces along with applied configuration parameters, such as metric value
	XR	
IOS	<code>show ipv6 protocols</code>	Displays routing protocol configuration information, such as enabled interfaces, redistribution, and AD
	XR	

**Table 19-13** IS-IS Verification Commands

The majority of the complexities associated with IPv6 for IS-IS relate to understanding the expected behavior of the topology modes.

[Figure 19-8](#) illustrates a dual-stack network using single-topology mode. To demonstrate the constraints of using single-topology mode, the transit link between routers R1 and R2 does not have IPv6 addressing configured. Because IS-IS single-topology mode performs a single SPF best path calculation, the routers do not differentiate between the IPv4 and IPv6 address families. The routers believe the link between R1 and R2 is the best path for both protocols.

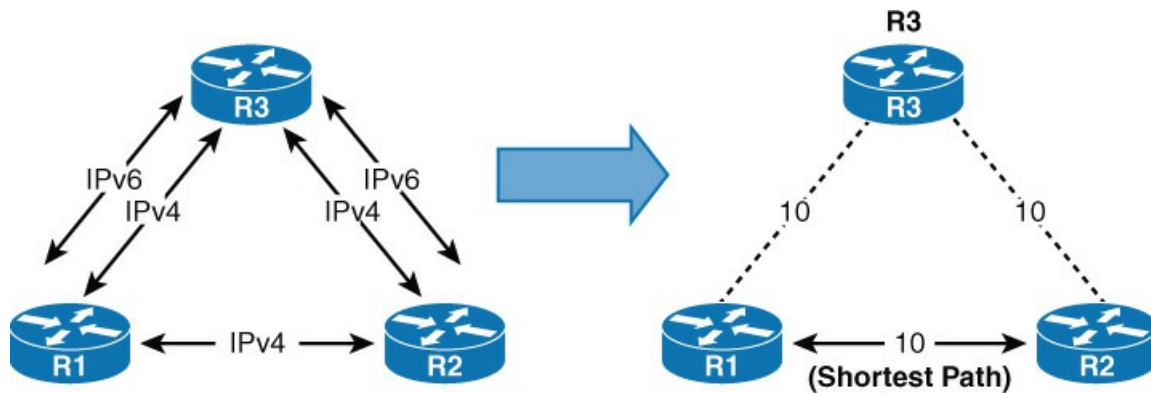


Figure 19-8 Single-Topology IS-IS

The command **show isis database detail** displays the networks in the IS-IS topology. [Example 19-28](#) displays R1's link-state packet database (LSPDB) entries for R2's IPv4 and IPv6 networks. Notice that all of R2's IPv6 routes are visible in the database.

### Example 19-28 IS-IS Database

[Click here to view code image](#)

```
R1#show isis database detail
! Output omitted for brevity
R2.00-00          0x00000038   0xB447      1152        0/0/0
  Area Address: 49.0001
  NLPID:         0xCC 0x8E
  Hostname: R2
  IP Address:    10.0.23.2
  IPv6 Address: 2001:DB8::2
  Metric: 10    IS R2.00
  Metric: 10    IS R3.00
  Metric: 10    IP 10.0.12.0 255.255.255.0
  Metric: 10    IP 10.0.23.0 255.255.255.0
  Metric: 10    IPv6 2001:DB8:23::/64
  Metric: 10    IPv6 2001:DB8::2/128
! Output omitted for brevity
```

R1's LSPDB includes all the IPv6 prefixes in the network. Because the best path is via a transit link without IPv6 addressing, the R2 loopback prefix 2001:db8::2/128 is unreachable. [Example 19-29](#) displays R1's IPv6 route table. Only R3's networks are reachable and included in the RIB.

### Example 19-29 IPv6 Route Table: Partial Connectivity

[Click here to view code image](#)

```

R1#show ipv6 route isis
IPv6 Routing Table - default - 6 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, R - RIP, H - NHRP, I1 - ISIS L1
       I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary, D - EIGRP
       EX - EIGRP external, ND - ND Default, NDp - ND Prefix, DCE - Destination
       NDr - Redirect, O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1
       OE2 - OSPF ext 2, ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, l - LISP
I1 2001:DB8::3/128 [115/20]
    via FE80::3, GigabitEthernet2/0
I1 2001:DB8:23::/64 [115/20]
    via FE80::3, GigabitEthernet2/0
! The route to R2's loopback address 2001:db8::2/128 is missing because there is not
! a valid IPv6 forwarding address between R1<->R2 link.

```

The only way to resolve the connectivity problem is to either configure IPv6 on all interfaces in the topology or change the router configuration to use the recommended multitopology mode.

Figure 19-9 displays the same topology but with multitopology mode enabled.

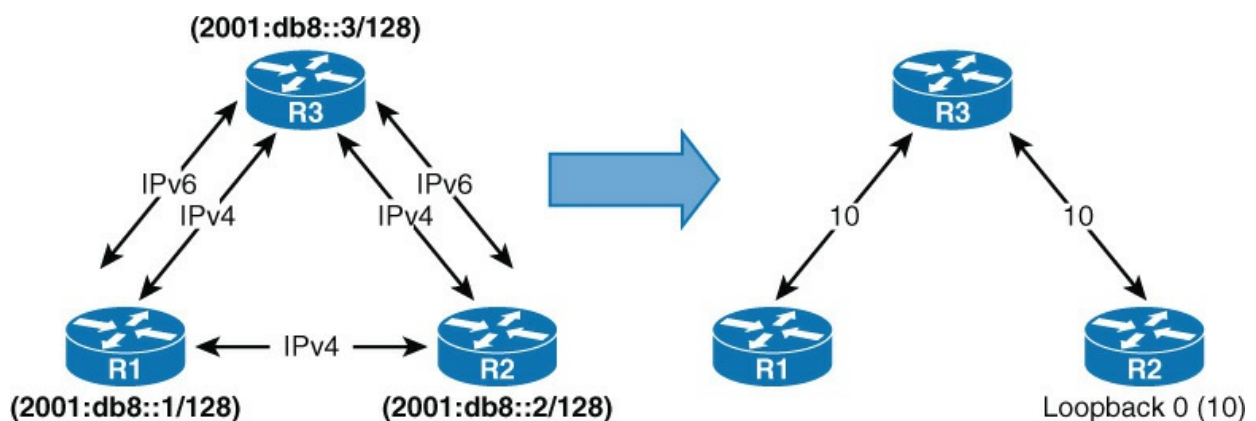


Figure 19-9 Multitopology IPv6 Routing Topology

With multitopology mode enabled, R1, R2, and R3 build an independent IPv6 LSPDB. The IPv4-only path between R1 and R2 is no longer an option, and the routers correctly determine that the only viable IPv6 path between the two routers is via R3.

Example 19-30 displays R1's route table with R3 as the next-hop forwarding address. Notice that the metric is 30 to reach the route 2001:db8::2/128. Like IPv4, each IPv6 interface along the path is counted as a hop (metric 10), including R2's loopback.

### Example 19-30 IPv6 Route Table: Full Connectivity

[Click here to view code image](#)

```

R1#show ipv6 route isis
IPv6 Routing Table - default - 9 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, R - RIP, H - NHRP, I1 - ISIS L1
       I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary, D - EIGRP
       EX - EIGRP external, ND - ND Default, NDp - ND Prefix, DCE - Destination
       NDr - Redirect, O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1
       OE2 - OSPF ext 2, ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, l - LISP
I1 2001:DB8::2/128 [115/30]
    via FE80::3, GigabitEthernet2/0
I1 2001:DB8::3/128 [115/20]
    via FE80::3, GigabitEthernet2/0
I1 2001:DB8:0:23::/64 [115/20]
    via FE80::3, GigabitEthernet2/0

```

### IS-IS Configuration Reference Chart for IPv6

It may be desirable to configure IPv6-specific topology information such as metric, AD, and maximum paths. Fundamentally, no difference exists between IPv4 and IPv6 when using these features. The only difference is the CLI syntax for applying the configuration. IOS interface configuration commands replace **ip isis** with **ipv6** keyword to differentiate between IPv4 and IPv6 protocols.

Table 19-14 lists some common IPv6 configuration commands. Additional IS-IS specific commands are covered in Chapters 8 and 9.

OS	Command	Description
IOS XR	<code>router isis [instance-id]</code>	Global configuration command that initializes the IS-IS process.  * IOS routers can use the null instance if not specified.
IOS XR	<code>net network-entiry-title</code>	IS-IS router configuration command that identifies the IS-IS NET for a router.
IOS XR	<code>ipv6 router isis [instance-id]</code> <code>interface interface-type</code> <code>interface-number</code> <code>address-family ipv6 unicast</code>	Interface parameter command that enables IS-IS on the interfaces.  * IOS routers can use the null instance if not specified.  IOS XR nodes enable the IPv6 protocol under the interface in IS-IS router configuration mode.
IOS XR	<code>default-information originate</code>	IS-IS IPv6 address family router configuration mode command that advertises a default route in level 2 (L2) LSPs.
IOS XR	<code>metric-style {wide   transition} [level-1   level-2   level-1-2]</code> <code>metric-style {narrow   wide   transition} [level {1 2}]</code>	IS-IS process configuration command that changes the metric-style for an IS-IS router. Metric style wide is required for IPv6 multitopology mode.  IOS XR nodes configuration resides under IPv6 address family within IS-IS router configuration mode.
IOS XR	<code>multitopology [transition]</code> <code>single-topology</code>	IS-IS IPv6 address family router configuration command that defines the IS-IS topology mode. IOS default mode of operation is single-topology mode.  IOS XR nodes configuration resides under IPv6 address family within IS-IS router configuration mode. The default mode of operation is multi topology mode.
IOS XR	<code>isis ipv6 metric {1-16777214   maximum} [level-1   level-2]</code> <code>metric {1-16777214   maximum} [level {1 2}]</code>	Interface parameter command that sets the IPv6 interface metric. Metrics can use different values for different levels.  IOS XR nodes configuration resides under IPv6 address-family within the interface IS-IS router configuration mode.
IOS XR	<code>summary-address prefix/prefix-length [level-1   level-2   level-1-2] [metric 1-4294967295]</code> <code>summary-prefix prefix/prefix-length {[level {1 2}]}</code>	IS-IS IPv6 address family router configuration mode command for summarizing network prefixes as they are advertised into a level.

Table 19-14 IS-IS Configuration Commands

## MULTIPROTOCOL BGP FOR IPV6

Multiprotocol extensions for BGPv4 (MP-BGP) enables BGP to carry network layer reachability information (NLRI) for multiple protocols, such as IPv4, IPv6, and Multiprotocol Label Switching (MPLS) Layer 3 virtual private networks (L3VPNs).

RFC 4760 defines the following new features:

- New address family identifier (AFI) model
- New BGPv4 optional and nontransitive attributes:
  - Multiprotocol reachable NLRI
  - Multiprotocol unreachable NLRI

The new multiprotocol reachable NLRI attribute describes IPv6 route information, while the multiprotocol unreachable NLRI withdraws the IPv6 route from service. The attributes are optional and nontransitive so that if an older router does not understand the attributes the information can just be ignored.

All the same underlying IPv4 path vector routing protocol features and rules described in the previous BGP chapters also apply to MP-BGP for IPv6.

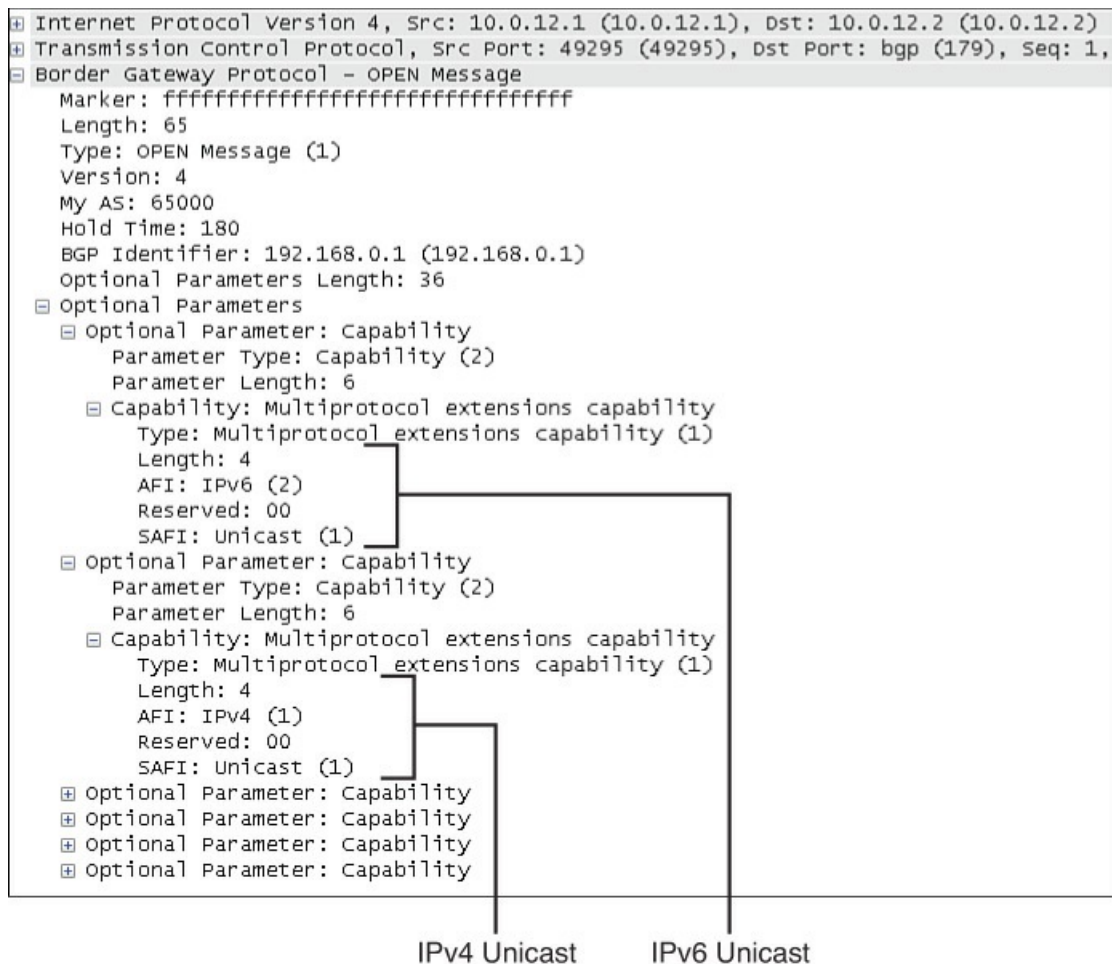
#### **Inter-Router Communication**

MP-BGP for IPv6 continues to use the same well-known TCP port 179 for session peering as BGP for IPv4. During the initial open message negotiation, the BGP peer routers exchange capabilities. The MP-BGP extensions include an address family identifier (AFI) that describes the supported protocols, along with subsequent address family identifier (SAFI) attribute fields that describe whether the prefix applies to the unicast or multicast routing table:

■ **IPv4 unicast:** AFI:1, SAFI:1

■ **IPv6 unicast:** AFI:2, SAFI:1

Figure 19-10 is an open message packet from a MP-BGP session negotiation. The packet capture demonstrates that the BGP session is negotiating support for exchanging both IPv4 and IPv6 unicast address families.



**Figure 19-10** Multiprotocol Extensions Capability

The packet capture demonstrates an important aspect of BGP. The protocol used to establish the BGP session is independent of the AFI/SAFI route advertisements. The TCP session used by BGP is a Layer 4 protocol and may use either an IPv4 or IPv6 address to form a session adjacency and exchange routes:

- IPv4 addressing
- Public IP addressing
- Private RFC 1918 addressing
- IPv6 addressing
- IPv6 link-local addressing
- IPv6 unique local addressing
- IPv6 unique global addressing

#### Note

Unique global unicast addressing is the recommended method for BGP peering to avoid operational complexity. BGP peering using the link-local address may introduce risk if the address is not manually assigned to an interface. A hardware failure or cabling move will change the MAC address, resulting in a new link-local address. This will cause the session to fail because the stateless address autoconfiguration will generate a new IP.

Advertising IPv6 prefixes over an IPv4 BGP session is covered in detail later in this chapter.

### BGP Configuration

This section introduces the required steps for routing IPv6 using MP-BGP. An configuration example for a small network topology of external Border Gateway Protocol (eBGP) and internal BGP (iBGP) peers is provided, along with the necessary verification commands to confirm the proper operation of the routing protocol.

### IOS Base Configuration

Cisco IOS extends support for IPv6 through a new multiprotocol AFI model. AFI independent parameters, such as a neighbor IP address and remote AS, are configured in the BGP parent configuration mode in the same method described in [Chapter 10](#). Address family-specific features however, are now configured within the IPv6 address family.

This includes parameters such as network advertisement, redistribution, route policies, and neighbor activation:

#### Step 1. Define the ASN.

The command **router bgp *as-number*** enables the BGP process and assigns an ASN.

#### Step 2. Assign the RID.

The command **bgp router-id *router-id*** manually assigns a RID. It is recommended that this parameter always be manually assigned to ensure proper operation of the routing process.

The MP-BGP RID is a 32-bit dotted-decimal value. The default behavior for BGP is to locally assign a RID based on the highest IPv4 loopback address, or if that is not available, the highest IPv4 address. If an IPv4 address is not configured, the process will not initiate.

#### Step 3. Add a neighbor router.

The IP address and ASN of the neighbor router is assigned with the command **neighbor {*ip-address* | *ipv6-address* [%*interface-id*] | *peer-group-name*} remote-as *autonomous-system-number***.

The optional % keyword is required when using link-local addressing to provide additional information as to which interface the LLA peering is associated.

**Example: neighbor fe80::4%gigabitethernet0/2 remote-as 65300**

#### Step 4. Assign the IPv6 address family.

The command **address-family IPv6 unicast** enters the address family configuration mode.

#### Step 5. Inject routes into MP-BGP (optional).



Within the IPv6 address family configuration mode, IPv6 prefixes may be injected into the BGP table using the commands **network**, **redistribute**, or **aggregate-address**.

### Step 6. Activate a neighbor.

The default behavior for BGP is to automatically advertise IPv4 unicast routes to neighbors. For the other address families, including IPv6, the activation of the AFI needs to be enabled for each neighbor router with the command **neighbor** {*ip-address* | *peer-group-name* | *ipv6-address*} **activate**.

#### Note

IPv4 unicast routing capability is advertised by default in IOS unless the neighbor is specifically shutdown within the IPv4 address family or globally within the BGP process with the command **no bgp default ipv4-unicast**.

### IOS XR Base Configuration

IOS XR uses the exact same configuration model as IPv4:

### Step 1. Define the ASN.

The command **router bgp** *as-number* enables the BGP process and assigns an ASN number.

### Step 2. Assign the RID.

The command **bgp router-id** *router-id* manually assigns an ID. It is recommended that this parameter always be manually assigned to ensure proper operation of the routing process.

The default IOS XR behavior for selecting the RID is to choose the highest IPv4 loopback address. IOS XR will not generate a RID from a physical interface IPv4 address. If an IPv4 loopback address is not configured or if an ID is not manually defined, the router will fail to form a BGP session.

### Step 3. Assign the IPv6 address family.

The command **address-family IPv6 unicast** enters the IPv6 address family configuration mode.

### Step 4. Inject routes into MP-BGP (optional).

Within the IPv6 address family configuration mode, IPv6 prefixes may be injected into the BGP table using the commands **network**, **redistribute**, or **aggregate-address**.

### Step 5. Define a neighbor.

The IP address of the neighbor is assigned with the command **neighbor ip-address** {*ip-address* | *ipv6-address*}.

### Step 6. Assign the ASN.

The ASN for the neighbor router is assigned with the command **remote-as** *as-number*.

\* eBGP neighbors require an inbound and output route policy. If no policy is configured, all routes are rejected for the neighbor.

### Step 7. Assign the address family.

The neighbor configuration command **address-family ipv6 unicast** enables IPv6 routing with the neighbor.

#### BGP Verification

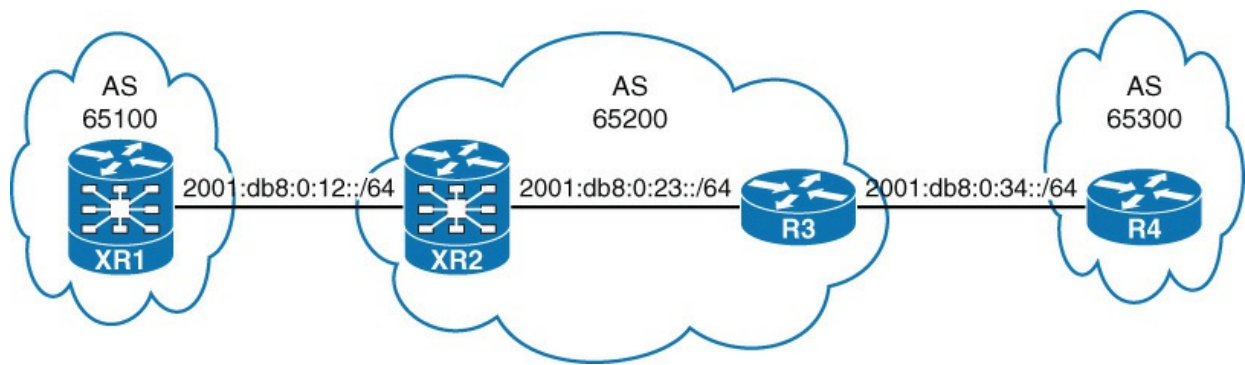
When working with multiple protocols, the specific address family needs to be specified in the **show** command. For example, the **ipv6** keyword is required to view IPv6 prefix information in BGP. Without the keyword, the router assumes the address family information request is for IPv4. The **ipv6** keyword is also required when clearing an adjacency or refreshing routes. For example, the command **clear bgp ipv6 unicast \* soft out** can be used to manually push an IPv6 routing update without tearing down and disrupting the BGP session.

Table 19-15 lists the most common **show** commands for reviewing the protocol operation.

OS	Command	Description
IOS XR	<b>show bgp ipv6 unicast summary</b>	Displays a summary view of all BGP neighbor sessions. A useful command for quickly determining the neighbor status and uptime.
IOS XR	<b>show bgp ipv6 unicast neighbors</b> [ <i>ipv6-address   ipv4-address</i> ] [ <i>received-routes   routes  </i> <i>flap-statistics   advertised-routes  </i> <i>dampened-routes</i> ]	Displays the BGP neighbor status, including negotiated capabilities and the details on the number of exchanged routes.
IOS XR	<b>show bgp ipv6 unicast</b> [ <i>ipv6-prefix</i> ]	Displays the IPv6 prefix in the BGP routing table.
IOS XR	<b>show ipv6 route bgp</b> <b>show route ipv6 bgp</b>	Displays the IPv6 BGP routes in the global router table.

Table 19-15 BGP Verification Commands

Figure 19-11 illustrates a network that is exchanging IPv6 routes using MP-BGP. In the example, XR1 and XR2 are eBGP peers. Similar to IPv4, the IOS XR routers need to include a route policy that permits prefix exchange over the eBGP peering. XR2 and R3 are iBGP peers. The next-hop-self feature ensures that the prefixes are advertised with the iBGP router's local IPv6 session peering address to avoid any reachability issues with the external routes. Finally, R3 and R4 are IOS routers. The IPv6 neighbor sessions require activation within the IPv6 unicast address family.



**Figure 19-11** IPv6 MP-BGP Topology

**Example 19-31** demonstrates how to configure BGP on the routers in the example topology.

### **Example 19-31** Router Configurations

[Click here to view code image](#)

```

XR1
route-policy ALLOW-ALL
  pass
end-policy
!
router bgp 65100
  bgp router-id 192.168.0.1
  address-family ipv6 unicast
    network 2001:db8::1/128
    network 2001:db8:0:12::/64
  !
  neighbor 2001:db8:0:12::2
    remote-as 65200
  address-family ipv6 unicast
    route-policy ALLOW-ALL in
    route-policy ALLOW-ALL out

```

**XR2**

```
route-policy PASS-ALL
  pass
end-policy
!
router bgp 65200
  bgp router-id 192.168.0.2
  address-family ipv6 unicast
    network 2001:db8::2/128
    network 2001:db8:0:12::/64
    network 2001:db8:0:23::/64
  !
  neighbor 2001:db8:0:12::1
    remote-as 65100
    address-family ipv6 unicast
      route-policy PASS-ALL in
      route-policy PASS-ALL out
  !
  !
  neighbor 2001:db8:0:23::3
    remote-as 65200
    address-family ipv6 unicast
      next-hop-self
```

**R3**

```
router bgp 65200
  bgp router-id 192.168.0.3
  bgp log-neighbor-changes
  neighbor 2001:DB8:0:23::2 remote-as 65200
  neighbor 2001:DB8:0:34::4 remote-as 65300
  !
  address-family ipv4
    no neighbor 2001:DB8:0:23::2 activate
    no neighbor 2001:DB8:0:34::4 activate
  exit-address-family
  !
  address-family ipv6
    network 2001:DB8::3/128
    network 2001:DB8:0:34::/64
    neighbor 2001:DB8:0:23::2 activate
    neighbor 2001:DB8:0:23::2 next-hop-self
    neighbor 2001:DB8:0:34::4 activate
  exit-address-family
```

#### R4

```
router bgp 65300
  bgp router-id 192.168.0.4
  bgp log-neighbor-changes
  neighbor 2001:DB8:0:34::3 remote-as 65200
  !
  address-family ipv4
    no neighbor 2001:DB8:0:34::3 activate
  exit-address-family
  !
  address-family ipv6
    network 2001:DB8::4/128
    network 2001:DB8:0:34::/64
    neighbor 2001:DB8:0:34::3 activate
  exit-address-family
```

Routers exchange AFI capabilities during the initial BGP session negotiation. The command **show bgp ipv6 unicast neighbors *ip-address* [detail]** displays detailed information on whether the IPv6 capabilities were negotiated successfully. The command **show bgp ipv6 unicast summary** displays a status summary of the sessions, including the number of routes that have been exchanged and the session uptime.

[Example 19-32](#) highlights the IPv6 AFI neighbor status for XR2. Notice that the two neighbor adjacencies have been up for 1 hour and 54 minutes. Neighbor 2001:db8:0:12::1 is advertising two routes, and neighbor 2001:db8:0:23::3 is sending three routes.

#### Example 19-32 IPv6 BGP Neighbor Status

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show bgp ipv6 unicast summary
BGP router identifier 192.168.0.2, local AS number 65200
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0xe0800000 RD version: 9
BGP main routing table version 9
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.

Process          RcvTblVer  bRIB/RIB  LabelVer  ImportVer  SendTblVer  StandbyVer
Speaker          9          9          9          9          9          9

Neighbor        Spk      AS  MsgRcvd  MsgSent  TblVer  InQ  OutQ  Up/Down  St/PfxRcd
2001:db8:0:12::1  0 65100    118    120      9     0    0 01:54:16    2
2001:db8:0:23::3  0 65200    131    118      9     0    0 01:54:16    3
```

[Example 19-33](#) displays the IPv6 unicast BGP table for router XR2. Notice that some of the routes include the unspecified address (::) as the next hop. The unspecified address indicates

that the local router is generating the prefix for the BGP table. The weight value 32768 also indicates that the prefix is locally originated by the router.

### Example 19-33 IPv6 Unicast BGP Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show bgp ipv6 unicast
BGP router identifier 192.168.0.2, local AS number 65200
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0xe0800000 RD version: 9
BGP main routing table version 9
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
               i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop          Metric LocPrf Weight Path
*> 2001:db8::1/128   2001:db8:0:12::1
                                     0           0 65100 i
*> 2001:db8::2/128   ::
                                     0           32768 i
*>i2001:db8::3/128   2001:db8:0:23::3
                                     0    100     0 i
*>i2001:db8::4/128   2001:db8:0:23::3
                                     0    100     0 65300 i
*> 2001:db8:0:12::/64 ::
*                               2001:db8:0:12::1
                                     0           0 65100 i
*> 2001:db8:0:23::/64 ::
*>i2001:db8:0:34::/64 2001:db8:0:23::3
                                     0    100     0 i

Processed 7 prefixes, 8 paths
```

Example 19-34 displays the IPv6 BGP route entries for XR2. Notice that there is a mixture of link-local and global addresses listed for the next-hop forwarding address.

### Example 19-34 IPv6 Route Table

[Click here to view code image](#)

```
RP/0/0/CPU0:XR2#show route ipv6 bgp
B   2001:db8::1/128
    [20/0] via fe80::1, 01:49:08, GigabitEthernet0/0/0/0
B   2001:db8::3/128
    [200/0] via 2001:db8:0:23::3, 01:49:08
B   2001:db8::4/128
    [200/0] via 2001:db8:0:23::3, 01:49:08
B   2001:db8:0:34::/64
    [200/0] via 2001:db8:0:23::3, 01:49:08
```

When the routers exchange routes, the multiprotocol reach NLRI attribute provides the BGP next-hop information for an IPv6 prefix. The attribute includes the global address and optionally

the link-local address. The link-local address is included in the advertisement only when the BGP neighbor routers share the same subnet.

Table 19-16 summarizes the rules the router uses for selecting which IPv6 next-hop forwarding address populates the route table.

BGP Peer Configuration	Advertised Next-hop Address (MP_NLRI_Attribute)	Next Hop Listed in Route Table (RIB)
IPv6 global address directly connected (eBGP)	IPv6 global address IPv6 LLA	IPv6 LLA.
IPv6 global address not directly connected (eBGP)	IPv6 global address	IPv6 global address.
IPv6 global address (iBGP)	IPv6 global address	IPv6 global address.
IPv6 LLA	IPv6 LLA	IPv6 LLA.
IPv4 address (legacy IOS/XR behavior)	IPv4-mapped IPv6 address (::FFFF.<IPv4>)	Route map policy must be configured to change the next hop manually.
IPv4 address (modern IOS/XR behavior) IOS-XR 3.8 and later IOS versions 12.0(32)SY09, 12.0(33)S6, 15.1(1)S, 15.1(4)M, 15(2)1T, and later	<ol style="list-style-type: none"> <li>1. If a route map is applied, use the next hop given in the route map.</li> <li>2. If a route map is not mentioned               <ol style="list-style-type: none"> <li>a. If it is directly connected peering, pick up a v6 address (global and link-local).</li> <li>b. If it is loopback peering (update-source configured), pick up a v6 address from that interface (global and link-local).</li> <li>c. Fall back to the default behavior of a v4-mapped v6 address.</li> </ol> </li> </ol>	

Table 19-16 Next-Hop Address Selection

### IPv6 over IPv4 BGP Sessions

BGP can exchange routes using either an IPv4 or IPv6 TCP session. In a typical deployment, IPv4 routes are exchanged using a dedicated IPv4 session and IPv6 routes are exchanged with a dedicated IPv6 session, as illustrated in Figure 19-12.

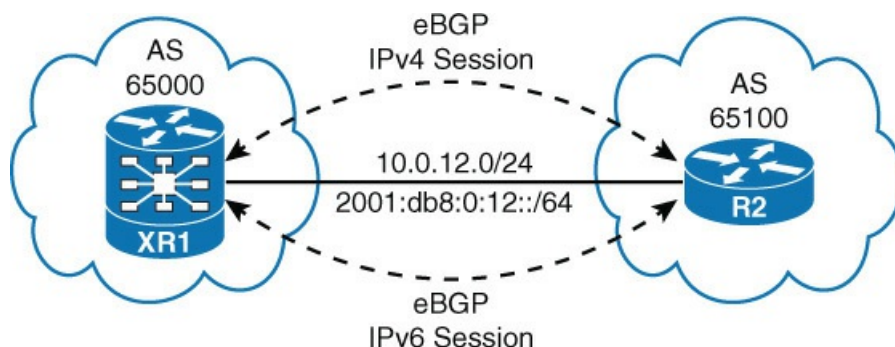
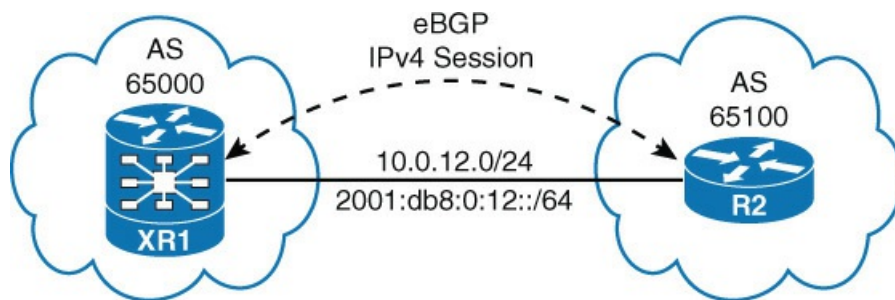


Figure 19-12 Independent BGP Sessions

It is also possible to share IPv4 and IPv6 routes using a single IPv4 BGP session. Figure 19-13 illustrates using one IPv4 session to share both IPv4 and IPv6 prefix information.



**Figure 19-13** Single BGP Session

Note

It is highly recommended that IPv6 have its own independent BGP session. Intermixing IPv4 and IPv6 routes over a single IPv4 BGP session unnecessarily complicates the topology and places the IPv6 route exchange at risk if there are instabilities with the IPv4 network.

On legacy versions of IOS and IOS XR software, the IPv6 routes advertised over an IPv4 BGP session are assigned an IPv4-mapped IPv6 address (::ffff:10.1.2.1) for the next hop. This is not a valid forwarding address, so the IPv6 route does not populate the RIB. To correct the problem, a route policy specifying the NLRI next-hop address is required.

**Example 19-35** displays the router configuration for routers XR1 and R2. Notice that each router includes a route policy to set the next-hop address for the IPv6 advertisements.

**Example 19-35** IPv6 over IPv4 BGP Configuration

[Click here to view code image](#)



**XR1**

```
route-policy PASS
  pass
end-policy
!
route-policy SET-HOP
  set next-hop 2001:db8:0:12::1
  pass
!
router bgp 65000
  address-family ipv4 unicast
    network 10.0.12.0/24
    network 192.168.0.1/32
  !
  address-family ipv6 unicast
    network 2001:db8::1/128
    network 2001:db8:0:12::/64
  !
  neighbor 10.0.12.2
    remote-as 65001
    address-family ipv4 unicast
      route-policy PASS in
      route-policy PASS out
  !
  address-family ipv6 unicast
    route-policy PASS in
    route-policy SET-HOP out
```

**R2**

```
route-map SET-HOP permit 10
  set ipv6 next-hop 2001:DB8:0:12::2
!
router bgp 65001
  bgp log-neighbor-changes
  neighbor 10.0.12.1 remote-as 65000
  !
  address-family ipv4
    network 10.0.12.0 mask 255.255.255.0
    network 192.168.0.2 mask 255.255.255.255
    neighbor 10.0.12.1 activate
  exit-address-family
  !
  address-family ipv6
    network 2001:DB8::2/128
    network 2001:DB8:0:12::/64
    neighbor 10.0.12.1 activate
    neighbor 10.0.12.1 route-map SET-HOP out
  exit-address-family
```

**Example 19-36** displays XR1's IPv6 neighbor summary. The output validates that the IPv6 routes are being shared over an IPv4 BGPv4 TCP connection.

**Example 19-36 IPv6 Address Family BGP Peering Sessions**

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show bgp ipv6 unicast summary
BGP router identifier 192.168.0.1, local AS number 65000
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0xe0800000 RD version: 8
BGP main routing table version 8
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.
```

Process	RcvTblVer	bRIB/RIB	LabelVer	ImportVer	SendTblVer	StandbyVer
Speaker	8	8	8	8	8	8

Neighbor	Spk	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	St/PfxRcd
10.0.12.2	0	65001	33	28	8	0	0	00:37:40	2

### BGP Configuration Command Reference Chart for IPv6

In IOS, the key difference between IPv4 and IPv6 BGP is that address family-specific commands, such as network advertisements and neighbor route policies, are defined under the IPv6 address family.

[Table 19-17](#) lists some common IPv6 configuration commands. Additional BGP-specific features are covered in [Chapters 14, “Advanced BGP,”](#) and [15, “BGP Best Path Selection.”](#)

OS	Command	Description
IOS XR	<code>clear bgp ipv6 unicast [*   <i>autonomous-system-number</i>   <i>ipv4-address</i>   <i>ipv6-address</i>   <i>peer-group-name</i>] [soft] [in   out]</code>	The command resets the BGP session for all neighbors (*) or for a specific neighbor. The soft keyword only triggers a route refresh, but does not reset peering.
IOS XR	<code>router bgp <i>as-number</i></code>	Global configuration command that initializes the BGP process.
IOS XR	<code>bgp router-id <i>router-id</i></code>	BGP router configuration command that manually defines the RID.
IOS	<code>neighbor {<i>ip-address</i>   <i>ipv6-address</i> [%<i>interface-id</i>]   <i>peer-group-name</i>} remote-as <i>as-number</i></code>	BGP router configuration command that defines the neighbor peer address for forming the BGP session.
IOS XR	<code>address-family ipv6 unicast</code>	BGP router configuration command that specifies the IPv6 address family, and enters address family configuration mode.
IOS XR	<code>network <i>ipv6-prefix/prefix-length</i></code>	BGP IPv6 address family configuration mode command to advertise a network prefix.
IOS XR	<code>aggregate-address <i>ipv6-prefix/prefix-length</i> [as-set] [as-confed-set] [summary-only] [route-policy <i>route-policy-name</i>]</code>	BGP IPv6 address family configuration mode command to create a summary route advertisement. The summary-only keyword suppresses advertisements of more specific routes.
IOS	<code>neighbor {<i>ip-address</i>   <i>peer-group-name</i>   <i>ipv6-address</i>[%]} activate</code>	BGP IPv6 address family configuration mode command that enables the BGP session.
XR	<code>neighbor {<i>ip-address</i>   <i>ipv6-address</i>}</code>	BGP configuration command that defines a neighbor peering and enters neighbor configuration mode.
XR	<code>remote-as <i>as-number</i></code>	BGP neighbor configuration mode command that defines the remote autonomous system of the peer router.
XR	<code>address-family ipv6 unicast</code>	BGP neighbor configuration mode command that enables IPv6 capabilities for the session and enters IPv6 neighbor configuration mode.
XR	<code>route-policy <i>route-policy-name</i> {in   out}</code>	BGP IPv6 neighbor configuration mode command to apply a route policy to a session.
IOS	<code>neighbor {<i>ip-address</i>   <i>peer-group-name</i>   <i>ipv6-address</i> [%]} route-map <i>map-name</i> {in   out}</code>	BGP IPv6 address family configuration mode command to apply a route map to a session.
IOS	<code>neighbor {<i>ip-address</i>   <i>peer-group-name</i>} default-originate [route-map <i>map-name</i>]</code>	BGP IPv6 address family mode command to advertise a default route ::/0 to a neighbor.
XR	<code>default-originate [route-policy <i>route-policy-name</i>]</code>	BGP IPv6 neighbor configuration mode command to advertise a default route ::/0.

Table 19-17 BGP Configuration Commands

## IPV6 ROUTE REDISTRIBUTION

The IPv4 route protocol redistribution rules covered in Chapter 11, “Route Maps and Route Policy,” also apply to IPv6 route redistribution. The only major modification is that the IOS operating system no longer redistributes the connected subnets on the interfaces over which the protocol is enabled. An IOS router will only redistribute route entries that exactly match the redistributed protocol in the route table.

Note

IOS XR routers use the same redistribution logic for IPv4 and IPv6.

Figure 19-14 illustrates a network where IOS router R2 is redistributing routes between EIGRPv6 and OSPFv3. The local interface prefixes are not included in the dynamic route redistribution because the route table reports them as connected.

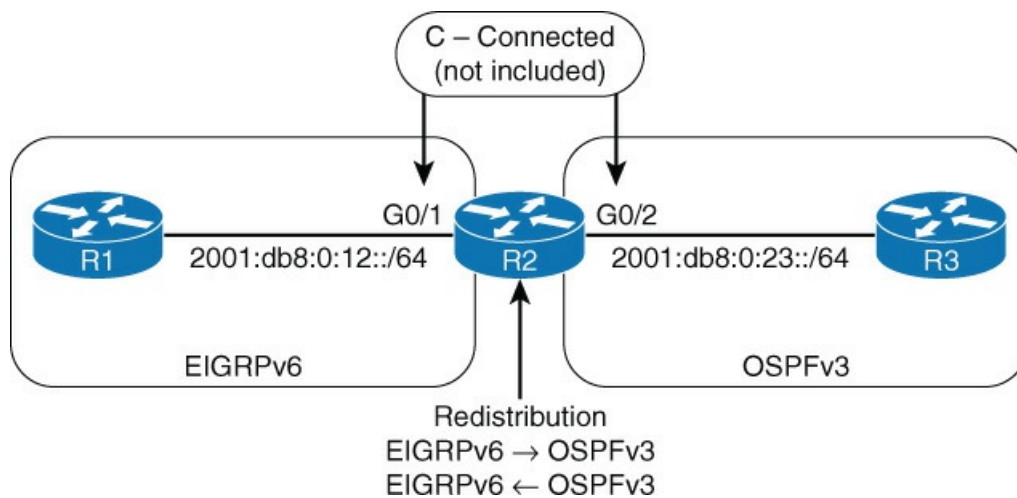


Figure 19-14 IOS IPv6 Route Redistribution

Example 19-37 displays R1’s view of the network. Notice that the router is learning R3’s loopback prefix 2001:db8::3/128 but is not receiving information for the OSPFv3 prefix 2001:db8:0:23::/64.

### Example 19-37 R1’s Route Table

[Click here to view code image](#)

```
R1#show ipv6 route eigrp
IPv6 Routing Table - default - 6 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDR - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, la - LISP alt
       lr - LISP site-registrations, ld - LISP dyn-eid, a - Application
D   2001:DB8::2/128 [90/10880]
    via FE80::2, GigabitEthernet0/1
EX  2001:DB8::3/128 [170/568320]
    via FE80::2, GigabitEthernet0/1
```

[Example 19-38](#) displays R2's route table. The route entries 2001:DB8:0:12::/64 and 2001:DB8:0:23::/64 are reported as *connected*, so they are not considered an exact match and will not be included in the IPv6 dynamic route redistribution.

### Example 19-38 R2's Route Table

[Click here to view code image](#)

```
R2#show ipv6 route
IPv6 Routing Table - default - 8 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, la - LISP alt
       lr - LISP site-registrations, ld - LISP dyn-eid, a - Application
D 2001:DB8::1/128 [90/10880]
   via FE80::1, GigabitEthernet0/1
LC 2001:DB8::2/128 [0/0]
   via Loopback0, receive
O 2001:DB8::3/128 [110/1]
   via FE80::3, GigabitEthernet0/2
C 2001:DB8:0:12::/64 [0/0]
   via GigabitEthernet0/1, directly connected
L 2001:DB8:0:12::2/128 [0/0]
   via GigabitEthernet0/1, receive
C 2001:DB8:0:23::/64 [0/0]
   via GigabitEthernet0/2, directly connected
L 2001:DB8:0:23::2/128 [0/0]
   via GigabitEthernet0/2, receive
L FF00::/8 [0/0]
   via Null0, receive
```

The key word **include-connected** can be used with the redistribution command to include the locally connected prefixes in the dynamic routing protocol redistribution. [Example 19-39](#) displays R2's routing protocol configuration using the **include-connected** command.

Also, notice that IPv6 for EIGRP follows the same rules as IPv4 and requires a metric value to be set when injecting routes into EIGRP.

### Example 19-39 R2's Configuration

[Click here to view code image](#)

## R2

```
router eigrp NET100
!
address-family ipv6 unicast autonomous-system 100
!
af-interface GigabitEthernet0/2
shutdown !EIGRP is not enabled on the interface connecting to R3
exit-af-interface
!
topology base
redistribute ospf 200 metric 10000 10 255 1 1500 include-connected
exit-af-topology
eigrp router-id 192.168.0.2
exit-address-family
!
router ospfv3 200
router-id 192.168.0.2
!
address-family ipv6 unicast
redistribute eigrp 100 include-connected
exit-address-family
!
interface GigabitEthernet0/2
ipv6 address FE80::2 link-local
ipv6 address 2001:DB8:0:23::2/64
ospfv3 200 ipv6 area 0
```

### Note

The **include-connected** keyword only injects prefixes for IOS interfaces that have a dynamic routing protocol enabled. To inject networks for interfaces without a dynamic protocol enabled, the **redistribute connected** command is still required.

Now that the included subnets configuration is applied to R2, the locally attached OSPF network 2001:db8:0:23::/64 is advertised to R1. [Example 19-40](#) displays R1's updated route table. Notice that the router now has a full view of the network.

### Example 19-40 R1's Route Table

[Click here to view code image](#)

```

R1#show ipv6 route eigrp
IPv6 Routing Table - default - 7 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDR - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, la - LISP alt
       lr - LISP site-registrations, ld - LISP dyn-eid, a - Application
D   2001:DB8::2/128 [90/10880]
    via FE80::2, GigabitEthernet0/1
EX  2001:DB8::3/128 [170/568320]
    via FE80::2, GigabitEthernet0/1
EX  2001:DB8:0:23::/64 [170/568320]
    via FE80::2, GigabitEthernet0/1

```

## SUMMARY

This chapter highlighted the key operational differences between IPv6 and IPv4 routing. Conceptually, there is not a significant difference in the routing protocol behavior between these technologies. Most of the routing protocol changes are scaling and operational improvements.

IPv6 routing protocols generally communicate using link-local addressing. The exceptions are IS-IS, which communicates directly at the data link layer, and BGP, which can be manually configured to a non-link-local address to form a peering session. Figure 19-15 illustrates that global unicast addressing is not required on the transit links between R1, R2, and R3 to share routing information. The network has full connectivity between each router's loopback interface.



Figure 19-15 Link-Local Transit

IPv6 routing uses the following rules:

- Routing protocols generally communicate using link-local addressing.
- Hard-coding link-local addressing simplifies network troubleshooting.
- IPv6 redistribution does not automatically include connected interfaces.
- Prefix lists or prefix sets are necessary for performing route filtering.

## REFERENCES IN THIS CHAPTER

Cisco. Cisco IOS Software Configuration Guides, <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides, <http://www.cisco.com>

Bates, T., R. Chandra, D. Katz, and Y. Rekhter. RFC 4760, *Multiprotocol Extensions for BGP-4*. IETF, <http://www.ietf.org/rfc/rfc4760.txt>, January 2007

Narten, T., E. Nordmark, H Simpson, and H. Soliman. RFC 4861, *Neighbor Discovery in IPv6*. IETF, <http://www.ietf.org/rfc/rfc4861.txt>, September 2007

Przygienda, T. and N. Sheth. RFC 5120, *M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)*. IETF, <http://www.ietf.org/rfc/rfc5120.txt>, February 2008

Hopps, C. RFC 5308, *Routing IPv6 with IS-IS*. IETF, <http://www.ietf.org/rfc/rfc5308.txt>, October 2008

Coltun, R., D. Ferguson, J. Moy, and A. Lindem, RFC 5340, *OSPF for IPv6*. IETF, <http://www.ietf.org/rfc/rfc5340.txt>, July 2008



## Chapter 20. IPv6 Multicast Routing

This chapter covers the following topics:

- Multicast Listener Discovery
- IPv6 multicast address mapping into a MAC Address
- Protocol Independent Multicast
- PIM Sparse Mode
- Embedded RP
- Reverse path forwarding
- IPv6 multicast boundary scope
- PIM Source Specific Multicast

Multicast routing conserves network bandwidth by allowing a host to send a single data stream to a group of listening clients instead of generating multiple unicast streams for each client. Similar to unicast, IPv6 multicast's primary benefit over IPv4 is improved scalability by extending the number of available multicast group addresses.

### IPV6 MULTICAST ROUTING OVERVIEW

IPv6 multicast routing retains many of IPv4's familiar features and protocol mechanics. IPv6 and IPv4 both use the same PIMv2 protocol to build the multicast distribution tree. [Chapters 16, "Multicast,"](#) and [17, "Advanced Multicast,"](#) provide comprehensive coverage of these core concepts.

The most notable changes and enhancements to IPv6 multicast are as follows:

- **MLD:** The Multicast Listener Discovery (MLD) protocol has replaced Internet Group Management Protocol (IGMP).
- **Embedded RP:** The rendezvous point (RP) address can be encoded in the multicast address.
- **Scope:** An IPv6 multicast address contains a multicast reachability scope for containing the range of the multicast stream.
- **Auto-RP:** Auto-RP is not supported.
- **PIM-DM:** PIM Dense Mode is not supported.
- **DVMRP:** Distance Vector Multicast Routing Protocol (DVMRP) is not supported.
- **MSDP:** IPv6 multicast does not include support for Multicast Source Discovery Protocol (MSDP).

Table 20-1 compares the common multicast features in IPv4 to IPv6.

Service	IPv4 Solution	IPv6 Solution
Addressing range	32-bit, Class D 224.0.0.0 – 239.255.255.255	128-bit (112-bit group) ff::/8
Routing	Protocol independent, all IGPs and MBGP	Protocol independent, all IGPs and MBGP with v6mcast SAFI
Multicast protocols	PIM-DM, PIM-SM, PIM-SSM, bidir-PIM	PIM-SM, PIM-SSM, bidir-PIM
Group management	IGMPv1, v2, v3	MLDv1, v2
Domain control	Boundary, border	Scope identifier
Interdomain solutions	MSDP across independent PIM domains	Single RP within globally shared domains

Table 20-1 IPv4 and IPv6 Multicast Comparison

Figure 20-1 illustrates the two key components of an IPv6 Multicast network: PIMv2 and Multicast Listener Discovery (MLD) protocol. Configuring IPv6 multicast routing on a Cisco router automatically enables the Protocol Independent Multicast (PIM) and MLD protocol on all IPv6 interfaces, in addition to multicast data forwarding.

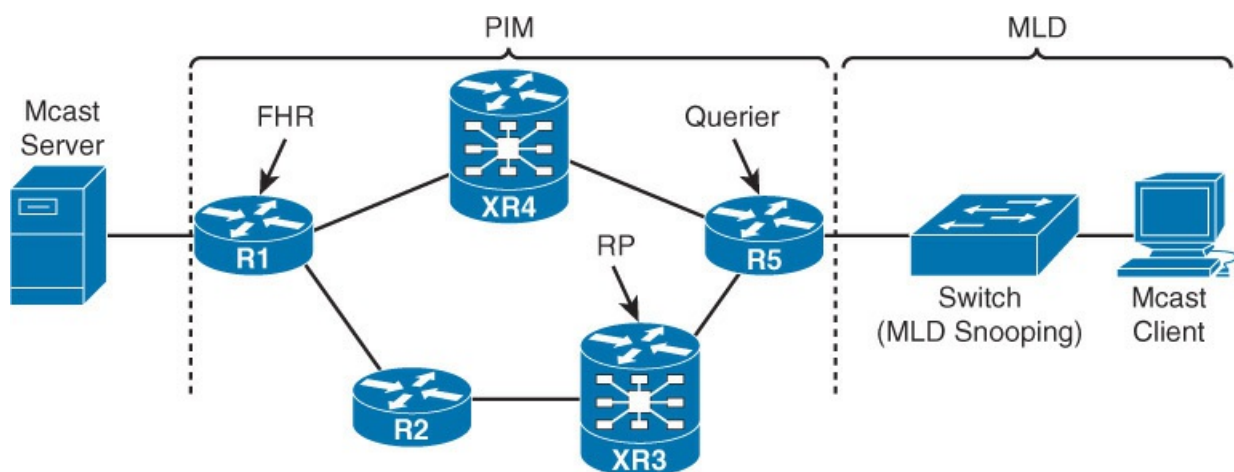


Figure 20-1 IPv6 Multicast

### IPv6 Multicast Address Mapping into MAC Address

A Layer 3 IPv6 multicast address is 128 bits long, whereas an Ethernet address is only 48 bits long. RFC 2464 provides a guideline for mapping IPv6 multicast addresses to Ethernet addresses to accommodate for the difference in bit length. The Ethernet MAC address's first four hexadecimal values should begin with 3333, and the last eight hexadecimal values should match the last eight values of the corresponding IPv6 address.

**Example:** IPv6 to MAC address mapping for OSPF Hello packet

- **IPv6 multicast address:** ff02::5 (compressed format)
- **IPv6 multicast address:** ff02:0000:0000:0000:0000:0000:0000:0005

## ■ MAC address: 33:33:00:00:00:05

Figure 20-2 illustrates an IPv6 multicast to Ethernet MAC address mapping.

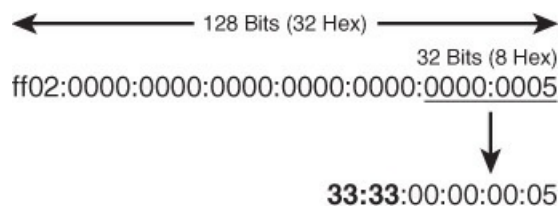


Figure 20-2 MAC Address Mapping

### Note

The address length of an Ethernet MAC address is smaller than an IPv6 address, so it is possible, though unlikely, for more than one multicast address to map to the same MAC hardware address. Assigning the group ID from the low-order 32 bits of the IPv6 multicast address, and setting the remaining group ID bits to 0, will ensure that each group ID maps to a unique MAC address. Chapter 18, "IPv6 Addressing," provides in-depth coverage of the IPv6 addressing format.

## Enabling Multicast Routing

IPv6 multicast routing is enabled in IOS with the global command **ipv6 multicast-routing**. In IOS XR, the address family is enabled by specifying **address-family ipv6** under the multicast-routing configuration mode, along with the participating interfaces. It is common to enable all interfaces with the **address-family** command **interface all enable**.

PIM and MLD are automatically enabled on all IPv6 interfaces once IPv6 multicast routing is enabled.

Example 20-1 demonstrates how to enable IPv6 multicast routing.

### Example 20-1 Enabling Multicast

[Click here to view code image](#)

```
IOS
ipv6 multicast-routing
```

```
IOS XR
multicast-routing
address-family ipv6
interface all enable
```

### Note

To configure multicast, the multicast software package needs to be loaded on the IOS XR router.

## Multicast Listener Discovery

An IPv6 host signals to the local multicast router that it would like to receive a multicast stream for a specific group address using the MLD protocol.

There are currently two versions of MLD: v1 and v2.

■ **MLDv1** is comparable to IGMPv2 for IPv4.

■ **MLDv2** is the default version used by Cisco routers and is the equivalent of IGMPv3 for IPv4. MLDv2 adds support for Source Specific Multicast (SSM) and is backward compatible with MLDv1 hosts.

Note

MLD snooping replaces IGMP snooping and allows switches to determine exactly which interfaces should receive the multicast stream for a specific group address.

Table 20-2 defines three types of MLD messages: report, done, and query. Notice that MLDv2 does not require done messages, but MLDv2 routers will process them if sent from an MLDv1 host.

Version	Message Type	Description
MLDv1 MLDv2	Report	A host sends a report message when it wants to receive a multicast stream or in response to a query sent by a router.
MLDv1	Done (Leaves)	When a host is finished listening to a multicast address it sends a done message.
MLDv1 MLDv2	Query	A router sends a query message to determine whether there are listeners present on the link.

Table 20-2 MLD Message Types

Enhancements to the MLDv2 report message eliminate the need for the done message. RFC 3810 describes MLDv2 as using two filter modes to determine whether a host is actively listening to a multicast stream.

Traditional multicast, otherwise known as *any-source multicast*, uses the exclude filter mode. In this mode, a host sends a report message to the router that describes the multicast group addresses that it is listening too. In exclude mode, streams are accepted from all source addresses except those listed in the source list:

■ The *exclude* record (type 2) indicates the host wants to receive the multicast stream for a group.

■ The *change to include* record (type 3) may be used to indicate the host is finished listening to the multicast stream or wants to change filter modes to include SSM. The record type will include the group address with 0 sources when the host is finished listening to the stream.

SSM utilizes the include filter mode. Hosts send MLD report messages that describe the group address and include the multicast streaming source address. In include mode, streams are accepted only from source addresses listed in the source list:

■ The *include* record (type 1) indicates the host is using the include filter mode for the specific multicast group address and source.

- The *allow new sources* record (type 5) indicates the host wants to receive a multicast stream for a group from additional specific source address.
- The *block old sources* record (type 6) indicates the host is finished listening to the multicast stream from the specific source address.
- The *change to exclude* record (type 4) may be used to switch filter modes to exclude.

Similar to the Neighbor Discovery Protocol, MLD is a special ICMPv6 packet. The IPv6 router alert extension header ensures that the router examines the packet. The MLD packets are sourced from the interface's link-local address. The destination address of the packet is dependent on the version of MLD and the MLD message type. MLDv2 packets include a multicast address Record field that indicates the multicast group address of interest, while MLDv1 will send a report (join) using the target multicast group address.

Table 20-3 displays the MLD destination addresses.

Version	Message Type	Destination address
MLDv1	Report	Multicast group address
MLDv1	Done	ff02::2 (all routers)
MLDv1	General query	ff02::1 (all nodes)
MLDv2	Report	ff02::16
MLDv2	General query	ff02::1 (all nodes)
MLDv2	Address-specific query	Multicast group address

Table 20-3 MLD Destination Address

A host will send a MLD report message that allows the router to learn which interfaces have multicast group address listeners. The local multicast router keeps a list of all the group addresses that have listeners on its attached interface, along with a timer for each of those addresses. The router will send query messages to determine whether hosts are actively listening to a multicast stream, ensuring that the hosts are still active. Each host uses a randomized response timer to prevent a flood of report response messages. If a report message is received before the timer expires, the report response message is suppressed.

When there are multiple routers on a segment, the routers perform a querier election. The router with the lowest interface IPv6 address is selected as the querier. The nonquerier router simply monitors the link. If the initial querier router stops responding after 255 seconds, the nonquerier router takes over. In addition, if a router detects that it has the lowest IPv6 address, it will perform preemption and take over the role of querier.

There are two types of MLD router query messages:

- **General query:** Sent by the router to the all-nodes multicast address (ff02::1) every 125 seconds and is used to learn which multicast addresses have active listeners.



```
RP/0/0/CPU0:XR3#show mld groups
MLD Connected Group Membership

Loopback0

Group Address : ff02::2
Last Reporter : fe80::7488:bdff:febd:2581
  Uptime : 02:37:07
  Expires : never
Group Address : ff02::d
Last Reporter : fe80::7488:bdff:febd:2581
  Uptime : 02:37:07
  Expires : never
Group Address : ff02::16
Last Reporter : fe80::7488:bdff:febd:2581
  Uptime : 02:37:07
  Expires : never
! Output omitted for brevity
```

MLD querying may be disabled with the interface parameter command **no ipv6 mld router** on IOS nodes, or with the IOS XR command **router disable** with the **router mld** configuration, as shown in [Example 20-3](#).

### Example 20-3 Disabling MLD

[Click here to view code image](#)

```
IOS
interface GigabitEthernet0/0
  no ipv6 mld router
```

```
IOS XR
router mld
  interface GigabitEthernet0/0/0/0
    router disable
```

Routers do not maintain state information for every multicast listener client by default, and only verify that a single active multicast listener is present. This behavior can be changed by enabling the explicit tracking feature in MLDv2. Explicit tracking works similarly to the IGMPv3 fast-leave mechanism and reduces the time a multicast stream is transmitted on the interface after the last active host leaves the multicast group.

Explicit tracking is enabled on IOS nodes with the interface parameter command **ipv6 mld explicit-tracking** [*access-list*]. In IOS XR, tracking is enabled within the **router mld** configuration mode under the desired interface with the command **explicit-tracking** [*access-list*].

[Example 20-4](#) demonstrates how to configure MLD explicit tracking.

## Example 20-4 Enabling Explicit Tracking

[Click here to view code image](#)

```
IOS
interface GigabitEthernet0/3
  ipv6 mld explicit-tracking
```

```
IOS XR
router mld
  interface GigabitEthernet0/0/0/0
    explicit-tracking
```

A router can join a specific group address to ensure that a multicast stream is always requested for a network segment. The static MLD join feature is a useful troubleshooting tool for verifying that the multicast distribution tree is forming properly.

In IOS, the MLD static join is configured with the interface parameter command **ipv6 mld join-group** [*group-address*]. For IOS XR the static join is configured in the router MLD configuration mode under the interface with the command **join-group** [*group-address*].

[Example 20-5](#) demonstrates how to configure a MLD static join for the group address ff1e::100.

## Example 20-5 MLD Static Join

[Click here to view code image](#)

```
IOS
interface Loopback0
  ipv6 mld join-group ff1e::100
```

```
IOS XR
router mld
  interface Loopback0
    join-group ff1e::100
```

## Protocol Independent Multicast

IPv6 routers use PIMv2 for signaling how to build the multicast distribution tree (MDT). The PIMv2 commands do not show up in the configuration because they are automatically enabled by default when multicast is configured. PIM may be disabled on a per-interface basis with the IOS command **no ipv6 pim** or in IOS XR PIM IPv6 router configuration mode using the **disable** keyword under the interface.

[Example 20-6](#) demonstrates how to disable IPv6 PIM on an interface.

## Example 20-6 Disabling PIM

[Click here to view code image](#)



**IOS**

```
Interface GigabitEthernet0/0
no ipv6 pim
```

**IOS XR**

```
router pim
address-family ipv6
interface GigabitEthernet0/0/0/0
disable
```

**PIM Sparse Mode**

PIM-SM is the default mode of operation for IPv6 multicast. IPv6 PIM-SM includes the same PIM “pull” mechanism described in [Chapters 16](#) and [17](#), such as the shortest-path tree (S,G), shared tree (\*,G), and bidir-PIM shared trees. IPv6 multicast does not include Dense Mode or the Auto-RP announcement “push” mechanism.

IPv6 PIM-SM multicast uses the same PIMv2 message types as IPv4. The source address is always the interface link-local address, and the destination address may be unicast or multicast depending on the message type.

[Table 20-5](#) lists the PIMv2 message types along with the source and destination IPv6 address.

Message Type	Source	Destination
0 = Hello	Link-local	ff02::d (all PIM routers)
1 = Register	Link-local	RP Address (unicast)
2 = Register Stop	Link-local	First-hop router (unicast)
3 = Join / Prune	Link-local	ff02::d (all PIM routers)
4 = Bootstrap	Link-local	ff02::d (all PIM routers)
5 = Assert	Link-local	ff02::d (all PIM routers)
8 = Candidate RP Advertisement	Link-local	BSR address (unicast)

**Table 20-5** PIMv2 Message Types

Similar to IPv4 multicast, IPv6 routers send PIMv2 hello messages every 30 seconds for neighbor discovery. When multiple routers are on the same segment the router with the highest designated router (DR), priority is elected as the DR. If the priority is the same, preference goes to the router with the highest IPv6 address. The DR router is responsible for performing important PIM actions such as a Join or Prune based on the presence of MLD clients on the local segment. The DR hold time is 3.5 times the hello interval or 105 seconds. If there are no hellos after this interval, a new DR is elected.

Every multicast router automatically creates a unidirectional PIM tunnel to a RP once it learns the RP address. There may be multiple tunnels depending on the number of RPs present in the network. The first-hop router (FHR) DR adjacent to the multicast sender uses the tunnel to send PIM register messages to the RP. PIM register-stop messages from the RP are sent directly to the registering FHR and do not traverse a PIM tunnel.

The tunnel will remain in an active up/up state even when there are no active multicast streams. The tunnel will remain active as long as there is a valid RPF path for the RP.

The status of a PIM tunnel is viewed in IOS with the command **show ipv6 pim tunnel** [*tunnel-id*], and the equivalent command in IOS XR is **show pim ipv6 tunnel info** {*tunnel-id* | *all*}.

**Example 20-7** displays the PIM tunnel information for the FHR and the RP. Notice that two encapsulation tunnels are present on the FHR representing the two known RPs. The first tunnel, tunnel 0, is for encapsulation register messages for local embedded RP multicast group traffic. The second tunnel is for registering all other group traffic to the RP 2001:db8::2. The RP router 2001:db8::2 also includes two encapsulation tunnels just like the FHR router, in addition to a receive tunnel for decapsulating incoming register messages from all of the FHR DR routers in the network.

### Example 20-7 IPv6 PIM Tunnel

[Click here to view code image](#)

```
FHR##show ipv6 pim tunnel
Tunnel0*
  Type   : PIM Encap
  RP     : Embedded RP Tunnel
  Source : 2001:DB8:0:14::1
Tunnel1*
  Type   : PIM Encap
  RP     : 2001:DB8::2
  Source : 2001:DB8::1
```

```
RP#show ipv6 pim tunnel
Tunnel0*
  Type   : PIM Encap
  RP     : Embedded RP Tunnel
  Source : 2001:DB8::2
Tunnel1*
  Type   : PIM Encap
  RP     : 2001:DB8::2*
  Source : 2001:DB8:0:23::2
Tunnel2*
  Type   : PIM Decap
  RP     : 2001:DB8::2*
  Source : -
```

There are three methods available in PIM-SM for defining an RP:

- Static RP
- Bootstrap Router (BSR)
- Embedded RP

## Static RP

The static RP PIM-SM method requires that every router in the multicast network include a manual entry of the RP address.

The IOS command that defines the RP is **ipv6 pim rp-address *ipv6-address***. In IOS XR, the RP is specified using the command **rp-address *ipv6-address*** under the ipv6 address family within the router PIM configuration mode.

Example 20-8 demonstrates how to configure a router with the static RP address 2001:db8::2.

### Example 20-8 Static RP Configuration

[Click here to view code image](#)

#### IOS

```
interface Loopback0
  ipv6 address 2001:db8::2/128
!
ipv6 pim rp-address 2001:db8::2
```

#### IOS XR

```
interface Loopback0
  ipv6 address 2001:db8::4/128
!
router pim
  address-family ipv6
    rp-address 2001:db8::2
```

#### Note

Statically defining an RP may be suitable in small environments, but can quickly lead to an administrative burden as the number of multicast routers increases. Static RP can also lead to fault tolerance issues unless *Anycast-RP* is used. With *Anycast-RP*, multiple routers are configured with the same loopback RP address so that if one router goes down the topology converges to use the closest router based on routing metrics.

## Bootstrap Router

The PIM bootstrap router (BSR) method allows for dynamically advertising RP information throughout the PIM domain. It is a more fault-tolerant solution than the static RP method because the RP configuration is automatically pushed to every router based on changing network conditions.

An IPv6 BSR advertises the location of the RP to the rest of the network using a PIM bootstrap message in the same manner as IPv4. The PIM bootstrap message is sent out every 60 seconds to the destination address ff02::d. The router will replicate only the message when the multicast-enabled interfaces have a PIM neighbor.

An election process determines the role of PIM BSR and RP router. The BSR candidate router with the highest assigned priority value is elected the BSR, while the RP candidate router with the lowest is selected.

Table 20-6 lists the priority values for BSR and RP election. Notice that the default candidate BSR priority value is higher for the IOS XR operating system. IOS XR routers do not permit a

BSR priority value of 0. If an IOS and IOS XR router are both deployed with default settings, the IOS XR router will be elected because it has the highest default priority value.

	<b>IOS Default Value</b>	<b>IOS XR Default Value</b>	<b>Election Criteria</b>	<b>Tie Breaker</b>
<b>Candidate RP</b>	192	192	Lowest	Highest RP address
<b>Candidate BSR</b>	0	1	Highest	Highest BSR address

**Table 20-6** Default BSR Priority Values and Election Criteria

If the priority is the same the router with the highest address is selected the BSR or RP candidate winner. An IPv6 multicast router can be elected a BSR, RP, or both.

The IOS global configuration command to advertise the BSR candidate address is **ipv6 pim bsr candidate bsr ipv6-address**. In IOS XR, the command **bsr candidate-bsr ipv6-address** is applied under the IPv6 address family within the router PIM configuration mode.

The IOS global configuration command to advertise the RP candidate address is **ipv6 pim bsr candidate rp ipv6-address**. In IOS XR, the command **bsr candidate-rp ipv6-address** is applied under the IPv6 address family within the router PIM configuration mode.

**Example 20-9** demonstrates how to configure BSR and RP candidacy using R2 and XR4's local loopback addresses.

### **Example 20-9** BSR Example

[Click here to view code image](#)

```
R2
interface Loopback0
  ipv6 address 2001:db8::2/128
!
ipv6 pim bsr candidate bsr 2001:DB8::2
ipv6 pim bsr candidate rp 2001:DB8::2
```

```
XR4
interface Loopback0
  ipv6 address 2001:db8::4/128
!
router pim
  address-family ipv6
    bsr candidate-bsr 2001:db8::4
    bsr candidate-rp 2001:db8::4
```

IOS XR autopopulates the default RP hash value, priority, and bootstrap message interval value in the global router configuration. Cisco IOS Software will populate only the router configuration with values that are nondefault. **Example 20-10** demonstrates the autopopulated keywords from

the configuration provided in [Example 20-9](#).

### Example 20-10 IOS XR Router PIM Configuration

[Click here to view code image](#)

```
RP/0/0/CPU0:XR3#show running-config router pim

router pim
address-family ipv6
  bsr candidate-bsr 2001:db8::4 hash-mask-len 126 priority 1
  bsr candidate-rp 2001:db8::4 priority 192 interval 60
```

The results of the BSR election are viewed with the IOS command **show ipv6 pim bsr election**, and in IOS XR with the command **show pim ipv6 bsr election**.

[Example 20-11](#) displays the result of the BSR election. As expected, XR4 is elected the BSR because its default priority value (1) is higher than R2's default priority value (0).

### Example 20-11 BSR Election

[Click here to view code image](#)

```
R5#show ipv6 pim bsr election
PIMv2 BSR information

BSR Election Information
Scope Range List: ff00::/8
  BSR Address: 2001:DB8::4
  Uptime: 00:01:28, BSR Priority: 1, Hash mask length: 126
  RPF: FE80::4,GigabitEthernet0/1
  BS Timer: 00:01:55
```

```
RP/0/0/CPU0:XR3#show pim ipv6 bsr election
PIM BSR Election State

Cand/Elect-State      Uptime      BS-Timer      BSR              C-BSR
No-Info/Accept-Pref  00:02:13   00:01:11     2001:db8::4 [1, 126]  :: [0,
```

The elected BSR router maintains a list of candidate RPs and is responsible for advertising the RP to the rest of the PIM network domain. [Examples 20-12](#) and [20-13](#) display the RP information. XR4 and R2 have the same priority value. XR4 is elected the RP for the entire multicast network `ff00::/8` because it has the highest IP address.

### Example 20-12 BSR RP Cache

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show pim ipv6 bsr rp-cache
Sat Mar  8 17:05:25.267 UTC
PIM BSR Candidate RP Cache

Group(s) ff00::/8, RP count 2
RP-addr      Proto    Priority  Holdtime(s)    Uptime    Expires
2001:db8::4  SM      192      150            00:13:29  00:02:00
2001:db8::2  SM      192      150            00:13:29  00:02:04
```

## Example 20-13 Candidate RP

[Click here to view code image](#)

```
RP/0/0/CPU0:XR4#show pim ipv6 bsr candidate-rp
Sat Mar  8 17:05:48.305 UTC
PIM BSR Candidate RP Info

Cand-RP      mode    scope    priority  uptime  group-list
2001:db8::4  SM      16       192       00:13:52  ff00::/8
```

### Note

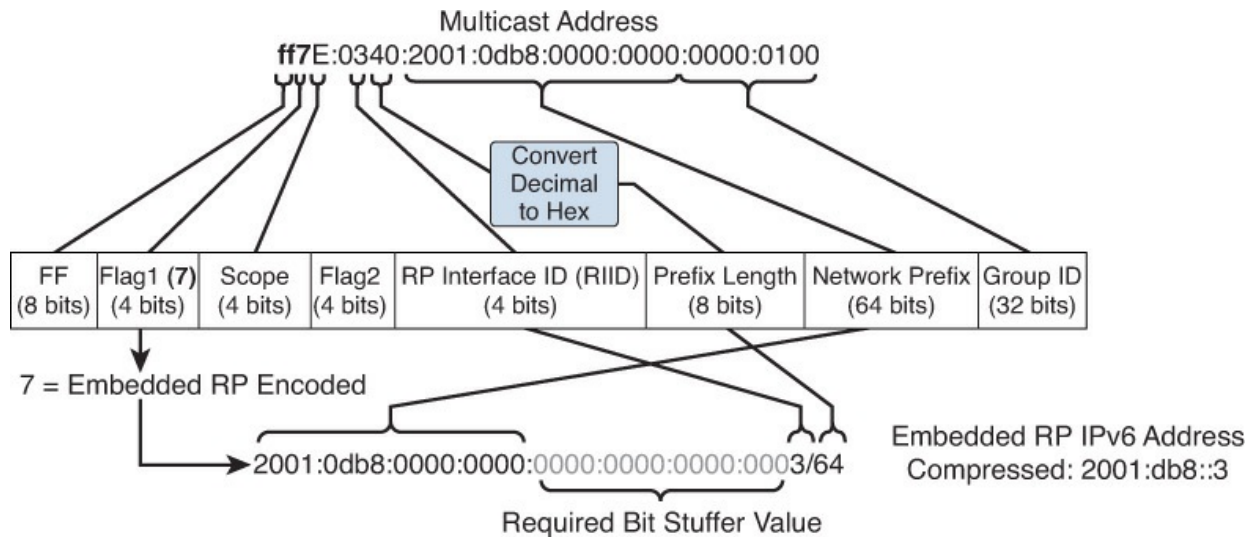
Only the BSR router contains data for the candidate RP and RP cache; therefore, the verification commands in [Examples 20-12](#) and [20-13](#) will not work on the other routers in the PIM domain.

## Embedded RP

Embedded RP encodes the RP address directly in the multicast address. Embedded RP addressing eliminates the need for static RP assignments or the usage of MSDP for interdomain multicast routing because the routers can identify the RP address by simply examining the multicast address. Embedded RP does not support bidirectional-PIM and has limited fault tolerance because only one RP address is encoded.

[Figure 20-3](#) illustrates how router XR3's loopback address 2001:db8::3 from [Figure 20-1](#) is encoded in the multicast address ff7e:340:2001:db8::100 as an embedded RP.

- Multicast address = ff7e:340:2001:db8::100
- Embedded RP = 2001:db8::3/64



**Figure 20-3** Embedded RP Address

The key fields in the multicast address are Flag1, RIID, Prefix Length, Network Prefix, and Group ID:

- Flag Field 1 = 7 (embedded RP present)
- Scope = E (global reachability)
- Flag Field 2 = 0 (reserved for future use)
- RIID = 3
- Prefix Length = /64
- Network Prefix = 2001:0db8:0000:0000
- Group ID = 000:0100

The steps for identifying the embedded RP's address are as follows:

**Step 1. Confirm that the multicast address begins with the value ff7.**

An embedded RP is encoded in the multicast address when the Flags Field 1 is set to 0111 (7).  
Embedded RP present = Yes

Note

In September 2014, RFC7371 modified the multicast flag bits processing rules so that multicast addresses beginning with ff7 or fff may be used to indicate an Embedded RP is present. IOS and IOS XR multicast implementations currently only detect embedded RP addresses when the multicast address begins with ff7.

**Step 2. The RP Interface Identifier (RIID) field describes the RP's IPv6 interface ID address.**

Regardless of the prefix length, the size of the router ID field prevents the interface ID from

including a value larger than a single hex value (nibble). The remaining bits in the interface ID are bit stuffed with 0s when deriving the final RP address. In [Figure 20-3](#) the embedded RP's interface ID is 3.

Partial – RP IPv6 address = X:X:X:X:X:X:X::3

### **Step 3. The Prefix Length field determines the high-order bits of the RP address.**

The address is notated in hexadecimal. To determine the actual prefix length the value needs to be converted to decimal. In the example, the Prefix Length field is 40, which converts to a prefix length of /64.

Partial – RP IPv6 address = X:X:X:X::3/64

### **Step 4. The Network Prefix field describes the subnet prefix.**

In the example, the Network Prefix field is populated with a value of 2001:0db8:0000:0000. The maximum supported prefix length for an embedded RP is 64 bits. A longer value would be too large to encode in the 128 bit multicast address.

Complete – RP IPv6 address = 2001:db8::3/64

#### Note

Using an embedded RP address reduces the available number of group addresses. The embedded RP address reduces the Group ID field from 112 bits to 32 bits. This is not a significant problem because 32 bits is still an extremely large value equal to the size of the entire IPv4 Internet.

The only configuration requirement for embedded RP is on the actual RP router. The router needs to be configured to perform the RP role for the multicast group. For IOS, the embedded RP configuration is applied to the RP router in exactly the same manner as defining a static RP.

The IOS XR embedded RP configuration requires two steps:

### **Step 1. Create IPv6 ACL for supported multicast groups.**

The IPv6 access list needs to match the multicast group ranges supported by the RP. The match criteria can be configured for all groups using a /96 prefix or for one specific multicast group using a /128 prefix.

### **Step 2. Define embedded RP address.**

The static RP address is enabled with the command **embedded-rp ipv6-address access-list** under the address family IPv6 within the **router pim** process.

[Example 20-14](#) demonstrates how to configure the embedded RP router. The IOS R2 router is configured to support the ff7e:240:2001:db8::/96 multicast group range, while the XR3 router is configured to support the ff7e:340:2001:db8::/96 multicast groups range.



## Example 20-14 Embedded RP Configuration

[Click here to view code image](#)

```
R2
interface Loopback0
  ipv6 address 2001:db8::2/128
  !
  ipv6 access-list MCAST
  permit ipv6 any FF7E:240:2001:DB8::/96
  !
  ipv6 pim rp-address 2001:DB8::2 MCAST
```

```
XR3
interface Loopback0
  ipv6 address 2001:db8::3/128
  !
  ipv6 access-list MCAST
  10 permit ipv6 any ff7e:340:2001:db8::/96
  !
  router pim
  address-family ipv6
  embedded-rp 2001:db8::3 MCAST
```

### Note

The multicast group range filter ACL applied to the IOS router in [Example 20-14](#) is optional. Applying a group range filter is a recommended best practice to ensure that only administratively defined multicast groups use the router as an RP.

The IOS command **show ipv6 pim group-map** and the IOS XR command **show pim ipv6 group-map** can be used to view the RP multicast group mappings. [Example 20-15](#) displays the example group mappings for the IOS and IOS XR routers.

## Example 20-15 Viewing Embedded RP Information

[Click here to view code image](#)

```
R2#show ipv6 pim group-map
! Output omitted for brevity
IP PIM Group Mapping Table
(* indicates group mappings being used)
FF7E:240:2001:DB8::/96*
  SM, RP: 2001:DB8::2
  RPF: Tu2,2001:DB8::2 (us)
  Info source: Static
  Uptime: 00:01:05, Groups: 1s
```

```

RP/0/0/CPU0:XR3#show pim ipv6 group-map
! Output omitted for brevity
IP PIM Group Mapping Table
(* indicates group mappings being used)
(+ indicates BSR group mappings active in MRIB)
Group Range                               Proto Client  Groups
ff7e:340:2001:db8::/96*                  SM          embed-cfg  1
  RP: 2001:db8::3
  RPF: De6tunnel1,2001:db8::3 (us)

```

### IPv6 Multicast Verification Commands

A vast majority of the IPv6 multicast verification commands are the same as the IPv4 commands. The key difference is that the **IPv6** command-line interface (CLI) keyword is required when performing the **show** command.

Table 20-7 provides a brief list of helpful IPv6 multicast verification commands. The majority of the commands have a similar format to IPv4.

OS	Command	Description
IOS	<code>show ipv6 mld interface [interface-type interface-number]</code>	Displays information for MLD interfaces.
XR	<code>show mld interface [interface-type interface-number]</code>	
IOS	<code>show ipv6 mld groups</code>	Displays information on MLD groups that are directly connected to the router.
XR	<code>show mld groups</code>	
IOS	<code>show ipv6 mld traffic</code>	Displays traffic counters for MLD packets.
XR	<code>show mld traffic</code>	
IOS	<code>show ipv6 pim interface</code>	Displays information for PIM interfaces.
XR	<code>show pim ipv6 interface</code>	
IOS	<code>show ipv6 pim neighbor</code>	Displays information for discovered PIM neighbors.
XR	<code>show pim ipv6 neighbor</code>	
IOS	<code>show ipv6 pim bsr election</code>	Displays PIM candidate election information for the BSR.
XR	<code>show pim ipv6 bsr election</code>	
IOS	<code>show ipv6 pim bsr candidate-rp</code>	Displays PIM candidate RP information for the BSR.
XR	<code>show pim ipv6 bsr candidate-rp</code>	
IOS	<code>show ipv6 pim bsr rp-cache</code>	Displays the PIM RP cache information for the BSR. This command will provide only data when issued from the BSR router.
XR	<code>show pim ipv6 bsr rp-cache</code>	
IOS	<code>show ipv6 pim tunnel</code>	Displays information about the PIM register encapsulation and decapsulation tunnels and a useful command for determining RP status.
XR	<code>show pim ipv6 tunnel info {interface   all}</code>	

IOS	<code>show ipv6 pim range-list</code>	Displays information on multicast groups, PIM operating mode, and how the specified range list was discovered.
XR	<code>show pim ipv6 range-list</code>	
IOS	<code>show ipv6 pim group-map</code>	Displays multicast group range to PIM mode mapping information along with RP assignment.
XR	<code>show pim ipv6 group-map</code>	
IOS	<code>show ipv6 pim topology</code> <code>show ipv6 mroute</code>	Displays the PIM topology table information. This is an extremely useful command for reviewing the multicast distribution tree state.
XR	<code>show pim ipv6 topology</code>	
IOS	<code>show ipv6 mroute active</code>	Displays active multicast streams on the router along with traffic rates.
XR	<code>show mfib route rate</code>	
IOS	<code>show ipv6 mrrib route</code>	Displays the multicast RIB.
XR	<code>show mrrib ipv6 route</code>	
IOS	<code>show ipv6 pim traffic</code>	Displays traffic counters for PIM packets.
XR	<code>show pim ipv6 traffic</code>	
IOS	<code>show ipv6 rpf ipv6-address</code>	A command to check the reverse path forwarding information for a unicast source address.
XR	<code>Show pim ipv6 rpf ipv6-address</code>	

Table 20-7 show Commands

Figure 20-4 illustrates a small IPv6 PIM-SM multicast network. A media-streaming server connected to R1's local LAN is streaming traffic to the multicast address ff1e::100. R5 has a client that has signaled that it would like to receive the multicast stream.

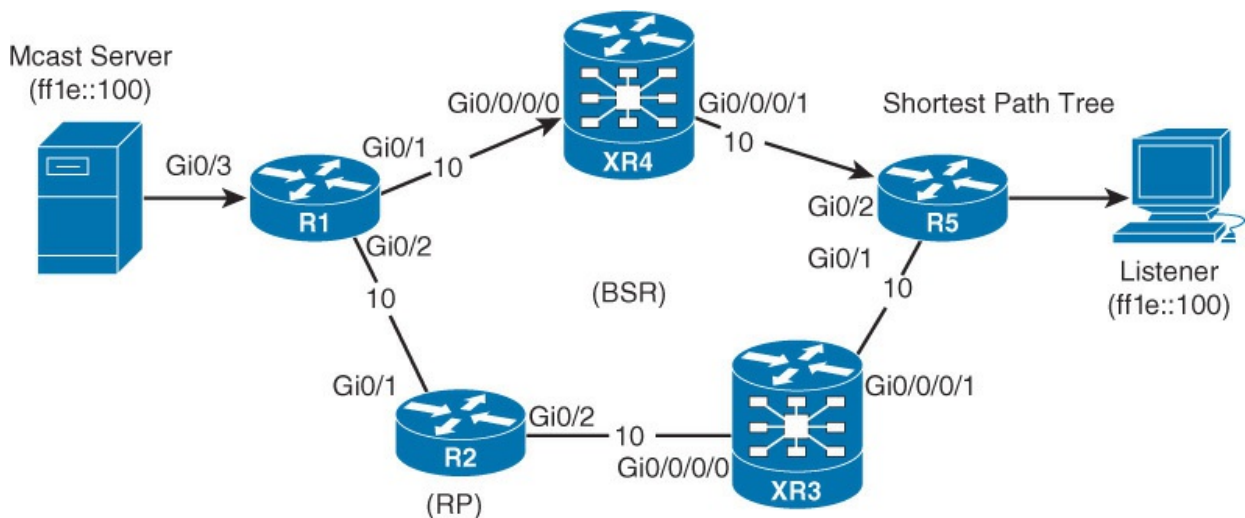


Figure 20-4 IPv6 Multicast Domain

Example 20-16 displays the key multicast configuration components for the example network.

### Example 20-16 Multicast Configuration

[Click here to view code image](#)

**R1**

```
ipv6 multicast-routing
```

**R2**

```
interface Loopback0
  ipv6 address 2001:db8::2/128
!
ipv6 multicast-routing
ipv6 pim bsr candidate rp 2001:DB8::2
```

**XR3**

```
mcast-routing
address-family ipv6
  rate-per-route
interface all enable
```

**XR4**

```
interface Loopback0
  ipv6 address 2001:db8::4/128
!
mcast-routing
address-family ipv6

  interface all enable
  accounting per-prefix
!
router pim
address-family ipv6
  bsr candidate-bsr 2001:db8::4
```

**R5**

```
ipv6 multicast-routing
```

The IOS command to check which router is the active RP for the multicast group is **show ipv6 pim group-map ipv6-multicast-address**, and the IOS XR command is **show pim ipv6 group-map ipv6-multicast-address**.

**Example 20-17** displays the active PIM group mapping for the multicast address ff1e::100. The assigned RP address is 2001:db8::2 (R2), operating in PIM-SM, and the RP is learned via BSR from 2001:db8::4 (XR4).

**Example 20-17 PIM Group Mapping**

[Click here to view code image](#)

```

R5#show ipv6 pim group-map FF1E::100
IP PIM Group Mapping Table
(* indicates group mappings being used)

FF00::/8*
  SM, RP: 2001:DB8::2
  RPF: Gi0/2,FE80::3
  Info source: BSR From: 2001:DB8::4 (00:02:23), Priority: 192
  Uptime: 03:21:47, Groups: 1

```

```

RP/0/0/CPU0:XR4#show pim ipv6 group-map FF1E::100
Sun Mar  9 00:47:36.047 UTC

IP PIM Group Mapping Table
(* indicates group mappings being used)
(+ indicates BSR group mappings active in MRIB)

Group Range                                Proto Client  Groups
ff00::/8*                                  SM           bsr+         0
  RP: 2001:db8::2
  RPF: Gi0/0/0/0,fe80::1)
RP/0/0/CPU0:XR4#

```

To view the multicast distribution tree there are two commands in IOS: **show ipv6 pim topology ipv6-multicast-address** or **show ipv6 mroute ipv6-multicast-address**. IOS XR only supports one command: **show pim IPv6 topology ipv6-multicast-address**.

**Example 20-18** displays the PIM topology table for ff1e::100 on XR4 and R5. R5 is reporting that a (S,G) shortest-path tree has been set up and that the reverse-path forwarding check expects multicast packets to be received on interface Gigabit Ethernet 0/1 (XR3). The outgoing interface for R5 is Gigabit Ethernet 0/3. XR3 topology also has an (S,G) entry and the RPF check expects incoming packets on interface Gigabit Ethernet 0/0/0/0 (R1). The packets will be replicated on interface Gigabit Ethernet 0/0/0/1 toward R5.

### **Example 20-18** PIM Topology Table

[Click here to view code image](#)

```
R5#show ipv6 mroute ff1e::100
```

```
Multicast Routing Table
```

```
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group,  
C - Connected, L - Local, I - Received Source Specific Host Report,  
P - Pruned, R - RP-bit set, F - Register flag, T - SPT-bit set,  
J - Join SPT, Y - Joined MDT-data group,  
y - Sending to MDT-data group  
g - BGP signal originated, G - BGP Signal received,  
N - BGP Shared-Tree Prune received, n - BGP C-Mroute suppressed,  
q - BGP Src-Active originated, Q - BGP Src-Active received  
E - Extranet
```

```
Timers: Uptime/Expires
```

```
Interface state: Interface, State
```

```
(* , FF1E::100), 06:49:08/never, RP 2001:DB8::2, flags: SCJ
```

```
Incoming interface: GigabitEthernet0/2
```

```
RPF nbr: FE80::3
```

```
Immediate Outgoing interface list:
```

```
GigabitEthernet0/3, Forward, 06:49:08/never
```

```
(2001:DB8:0:1::10, FF1E::100), 00:06:00/00:01:02, flags: SJT
```

```
Incoming interface: GigabitEthernet0/1
```

```
RPF nbr: FE80::4
```

```
Inherited Outgoing interface list:
```

```
GigabitEthernet0/3, Forward, 06:49:08/never
```

```
RP/0/0/CPU0:XR4#show pim ipv6 topology ff1e::100
```

```
IP PIM Multicast Topology Table
```

```
Entry state: (*S,G) [RPT/SPT] Protocol Uptime Info
```

```
Entry flags: KAT - Keep Alive Timer, AA - Assume Alive, PA - Probe Alive
```

```
RA - Really Alive, IA - Inherit Alive, LH - Last Hop
```

```
DSS - Don't Signal Sources, RR - Register Received
```

```
SR - Sending Registers, SNR - Sending Null Registers
```

```
E - MSDP External, EX - Extranet
```

```
MFA - Mofrr Active, MFP - Mofrr Primary, MFB - Mofrr Backup
```

```
DCC - Don't Check Connected, ME - MDT Encap, MD - MDT Decap
```

```
MT - Crossed Data MDT threshold, MA - Data MDT Assigned
```

```
SAJ - BGP Source Active Joined, SAR - BGP Source Active Received,
```

```
SAS - BGP Source Active Sent, IM - Inband mLDP
```

```
Interface state: Name, Uptime, Fwd, Info
```

```
Interface flags: LI - Local Interest, LD - Local Dissinterest,
```

```
II - Internal Interest, ID - Internal Dissinterest,
```

```
LH - Last Hop, AS - Assert, AB - Admin Boundary, EX - Extranet,
```

```
BGP - BGP C-Multicast Join, BP - BGP Source Active Prune,
```

```
MVS - MVPN Safi Learned, MV6S - MVPN IPv6 Safi Learned
```

```
(2001:db8:0:1::10, ff1e::100) SPT
```

```
SM Up: 00:05:44 JP: Join(00:00:14) Flags:
```

```
RPF: GigabitEthernet0/0/0/0, fe80::1
```

```
GigabitEthernet0/0/0/1 00:05:44 fwd Join(00:02:48)
```

The IOS command to view the active multicast streams is **show ipv6 mroute active**. The most similar command in IOS XR is **show mfib ipv6 route rate ipv6-multicast-address**.

IOS XR routers require the command **rate-per-route** under the address-family in multicast routing configuration mode to gather statistics for the command **show mfib ipv6 route rate**.

[Example 20-19](#) demonstrates how to enable IPv6 multicast stream statistics.

### Example 20-19 Enabling (S,G) Rate Calculations

[Click here to view code image](#)

```
multicast-routing
address-family ipv6
  rate-per-route
```

[Example 20-20](#) displays the active multicast flows for R5. The media server multicast server is streaming to group ff1e::100 and is consuming 75 Kbps of bandwidth.

### Example 20-20 Active Traffic Flows

[Click here to view code image](#)

```
R5#show ipv6 mroute active
Active Multicast Sources - sending >= 4 kbps
Default
Group: FF1E::100
  Source: 2001:DB8:0:1::10,
  SW Rate: 6 pps/75 kbps (1sec), 75 kbps (last 441 sec)
RP/0/RSP0/CPU0:XR4#show mfib ipv6 route rate FF1E::100

IP Multicast Forwarding Rates
(Source Address, Group Address)
  Incoming rate: pps (Incoming node)
  Outgoing rate:
  Node: (Outgoing node) : pps

(*,ff1e::100)
  Incoming rate : 0 pps
  Outgoing rate :

(2001:db8:0:12::1,ff1e::100)
  Incoming rate : 6 pps (0/0/CPU0)
  Outgoing rate : 6 pps (0/0/CPU0)
```

### Reverse Path Forwarding

The IPv6 concepts of reverse path forwarding (RPF) checks are identical to IPv4 multicast routing. The incoming interface of the multicast stream is selected based on the routing information in the unicast routing table for the multicast server's source address or based on the RP address for shared trees. If a multicast packet is received on an RPF interface, then the packet is forwarded out the outgoing interface list (OIL). If the multicast packet is received on a non RPF interface then the packet is dropped by the router.

Note

Older versions of Cisco IOS Software did not consider Border Gateway Protocol (BGP) unicast routes (SAFI = 1) during the RPF check. You can use the IOS command `ipv6 multicast rpf use-bgp` to enable RPF checks for BGP unicast routes.

In some topologies, it may be desirable to have asymmetrical unicast and multicast routing topologies. Utilizing static multicast routes or the multicast address family in MP-BGP allows for the modification of the multicast topology.

To configure a multicast static route in IOS the **multicast** keyword is required at the end of the **route** statement. In IOS XR, the multicast route is applied under the IPv6 multicast address family in the router static configuration mode.

Example 20-21 demonstrates how to create a static multicast route for the prefix 2001:db8:0:1::/64.

### Example 20-21 Static Mroute

[Click here to view code image](#)

**IOS**

```
ipv6 route 2001:DB8:0:1::/64 GigabitEthernet0/1 2001:DB8:0:35::3 multicast
```

**IOS XR**

```
router static
address-family ipv6 multicast
2001:db8:0:1::/64 GigabitEthernet0/0/0/0
```

Figure 20-5 illustrates how the multicast stream has been modified after applying a static multicast route 2001:db8:0:1::/64 to R5. The result of the static route is that R5 creates (S,G) source trees for the multicast group ff1e::100 using the network path R1-R2-XR3-R5.

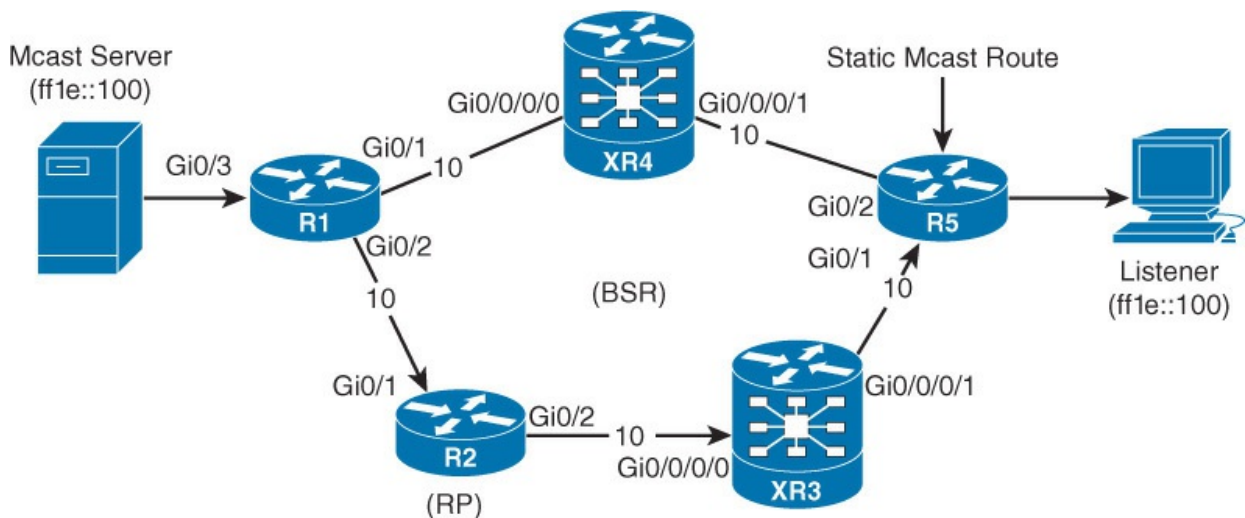


Figure 20-5 Multicast Topology

Example 20-22 demonstrates that the RPF check for the multicast server's address



2001:db8:0:1::100/64 is interface Gigabit Ethernet 0/1, which connects directly to XR3.

### Example 20-22 Reverse-Path Forwarding

[Click here to view code image](#)

```
R5#show ipv6 rpf 2001:db8:0:1::
RPF information for 2001:DB8:0:1::
  RPF interface: GigabitEthernet0/1
  RPF neighbor: 2001:DB8:0:35::3
  RPF route/mask: 2001:DB8:0:1::/64
  RPF type: Mroute
  RPF recursion count: 0
  Metric preference: 1
  Metric: 0
```

### Multicast Boundary Scope

The multicast address Scope field indicates whether the router should forward the multicast traffic. Figure 20-6 illustrates that the Scope field is represented by the fourth hexadecimal nibble value in the multicast address.

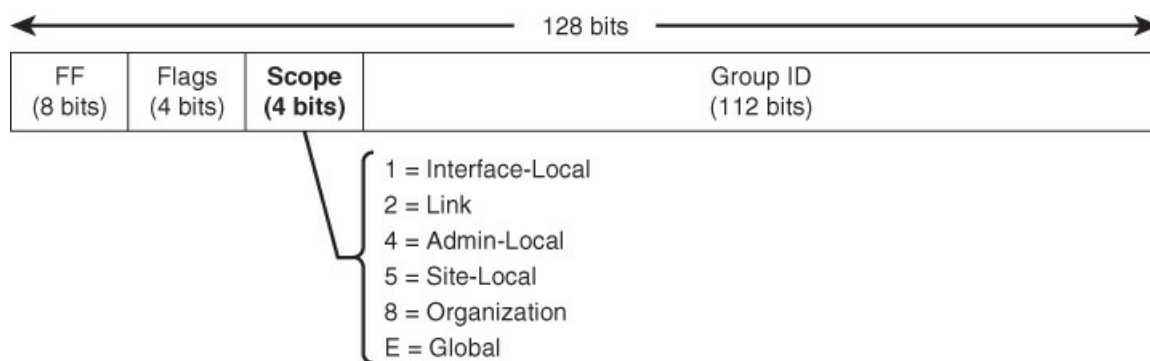


Figure 20-6 Scope Field

The IOS interface parameter command for configuring a multicast scope boundary is **ipv6 multicast boundary scope scope-value**.

#### Note

IOS routers will block traffic streams that belong to scopes that are less than or equal to the one configured.

The IOS XR multicast boundary configuration requires two steps:

#### Step 1. Create IPv6 ACL for the denied multicast groups:

The IPv6 access list includes a specific list of deny clauses for matching the multicast address ranges that are to be administratively blocked. The last line of the ACL is a **permit** statement to allow the other multicast group traffic.

#### Step 2. Assign Multicast Boundary:

The multicast boundary is applied with the command **boundary ipv6-address access-list under the interface** in IPv6 multicast configuration mode.

Example 20-23 demonstrates how to configure a site-local (5) scope boundary to block multicast streams from traversing an interface.

### **Example 20-23** *Multicast Scope Boundary*

[Click here to view code image](#)

```
IOS
interface GigabitEthernet0/1
ipv6 multicast boundary scope site-local
```

```
IOS XR
ipv6 access-list SITE-BOUNDARY
 10 deny ipv6 any ff05::/16
 20 deny ipv6 any ff15::/16
 30 deny ipv6 any ff35::/16
 40 deny ipv6 any ff75::/16
 50 permit ipv6 any any
!
multicast-routing
address-family ipv6
  interface GigabitEthernet0/0/0/1
    boundary SITE-BOUNDARY
```

#### **PIM Source Specific Multicast**

PIM Source Specific Multicast (PIM-SSM) allows for a multicast listener to signal the specific source address channel (S,G) that it would like join. IOS and IOS XR routers automatically support SSM and MLDv2 once IPv6 multicast routing is enabled.

The address prefixes `ff3X::/32` and `ffbX::/32` are reserved for SSM, where *X* is the Scope Identifier field. Currently, the only valid SSM prefix range according to RFCs 3306 and 4607 is `ff3X::/96`. There is no special configuration required to support the MLDv2 SSM group mappings. Cisco routers automatically create all the `ff3X::/32` SSM group mappings by default.

Unfortunately, many host applications do not yet support MLDv2. Cisco routers support statically mapping MLDv1 report requests to match a DNS-based or a static SSM group to source address mapping. DNS-based mapping is the IOS default mode of operation once the SSM mapping feature is enabled.

The IOS XR MLDv1 SSM mapping requires two steps:

#### **Step 1. Enable SSM mapping.**

The global configuration command **ipv6 mld ssm-map enable** activates the MLDv1 SSM group mapping feature.

#### **Step 2. Define the source address for the multicast group.**

The global configuration mode command **ipv6 mld ssm-map query dns** is automatically enabled upon completion of step one. A DNS server is required for DNS-based resolution to

succeed. The command **ip name-server** [*ipv6-address* | *ipv4 address*] defines a DNS server. The global configuration mode command **ipv6 mld ssm-map static** *access-list source-address* may be used to create a manual group to source SSM mapping. The access list matches the group addresses that are to use the designated source address.

The IOS XR MLDv1 SSM mapping requires one step:

### Step 1. Define the source address for the multicast group.

The MLD configuration mode command **ssm map static** *source-address access-list* creates a manual group to source SSM mapping. The access list matches the group addresses that are to use the designated source address.

[Example 20-24](#) provides full static SSM mapping configuration for all of the multicast streams using an SSM group address in the ff3e::/96 range. The streams will receive multicast traffic from the server with IPv6 address 2001:db8:0:1::100.

### Example 20-24 Static MLDv1 PIM-SSM Mapping

[Click here to view code image](#)

```
IOS
ipv6 access-list SSM-GROUP
  permit ipv6 any FF3E::/96
!
ipv6 mld ssm-map enable
ipv6 mld ssm-map static SSM-GROUP 2001:DB8:0:1::100
no ipv6 mld ssm-map query dns
```

```
XR
ipv6 access-list SSM-GROUP
  10 permit ipv6 any ff3e::/96
!
router mld
  ssm map static 2001:db8:0:1::100 SSM-GROUP
```

## SUMMARY

This chapter highlighted the key operational differences between IPv4 and IPv6 multicast routing. IPv6 multicast implementation is similar to IPv4 but includes enhancements such as simplified configuration, embedded RP, and increased scaling due to the larger IPv6 address length.

IPv6 multicast routing key concepts include the following:

- IPv6 uses PIMv2 Sparse Mode to create the multicast distribution tree.
- Configuring IPv6 multicast routing automatically enables PIM and MLD on all IPv6 interfaces.
- IPv6 MLDv2 is similar to IPv4's IGMPv3 protocol and is the default mode of operation on Cisco routers.

- IPv6 multicast boundaries are controlled through scope management.
- IPv6 allows for embedding an RP address in the multicast address.
- PIM SSM is enabled by default.

## REFERENCES IN THIS CHAPTER

Cisco. Cisco IOS Software Configuration Guides, <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides, <http://www.cisco.com>

Popoviciu, Ciprian, Eric Levy-Abegnoli, and Patrick Grossetete. *Deploying IPv6 Networks*. Indianapolis: Cisco Press, 2006.

Cisco. IPv6 Multicast Deployment and Configuration Guide, [http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/ipv6-multicast/product\\_data\\_sheet0900aecd80320fb8.pdf](http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/ipv6-multicast/product_data_sheet0900aecd80320fb8.pdf)

Crawford, M. RFC 2464, Transmission of IPv6 Packets over Ethernet Networks. IETF, <http://www.ietf.org/rfc/rfc2464.txt>, December 1998

Deering, S., W. Fenner, and B. Haberman. RFC 2710, *Multicast Listener Discovery (MLD) for IPv6*, IETF, <http://www.ietf.org/rfc/rfc2710.txt>, October 1999

Deering, S., W. Fenner, and B. Haberman. RFC 3810, *Multicast Listener Discovery Version 2 (MLDv2) for IPv6*. IETF, <http://www.ietf.org/rfc/rfc3810.txt>, June 2004

Savola, P. and B. Haberman. RFC 3956, *Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address*. IETF, <http://www.ietf.org/rfc/rfc3956.txt>, November 2004

Holbrook, H. and B. Cain. RFC 4607, *Source-Specific Multicast for IP*. IETF, <http://www.ietf.org/rfc/rfc4607.txt>, August 2006

Boucadair, M. and S. Venaas. RFC 7371, *Updates to the IPv6 Multicast Addressing Architecture*. IETF, <http://www.ietf.org/rfc/rfc7371.txt>, September 2014

## Index

### SYMBOLS

- \* (asterisks) query modifier, [482](#)
- [ ] (brackets) query modifier, [479](#)
- ^] (caret in brackets) query modifier, [480](#)
- ^ (carets) query modifier, [478](#)
- \$ (dollar signs) query modifier, [478-479](#)
- (hyphen) query modifier, [479-480](#)
- () (parentheses) query modifier, [480](#)
- . (period) query modifier, [481](#)
- | (pipe) query modifier, [480](#)
- + (plus sign) query modifier, [481](#)
- ? (question mark) query modifier, [481-482](#)
- \_ (underscores) query modifier, [477-478](#)
- 2-Way state (OSPF neighbors), [196](#)
- 10.55.200.33/12 address range calculations, [45](#)
- 172.16.2.3/23 address range calculations, [45](#)
- 192.168.100.0/24 subnet challenge, [46](#)
  - available hosts, [47](#)
  - available subnets, [47](#)
  - binary bit values, [47](#)
  - final address allocation scheme, [49](#)
  - LAN 1/LAN 2 subnet assignments, [47-48](#)
  - LAN 3 assignments, [48](#)
  - 192.168.100.5/26 address range calculations, [45](#)

## **A**

**abbreviations (IPv6 addresses), 895-896**

**ABF (access-list-based forwarding), 521**

configuring, 523-525

overview, 521

**ABRs (Area Border Routers), 242**

totally stubby areas, 289

Type 3 LSAs, 259

**accumulated interior gateway protocol. See AIGP**

**ACEs (access control entries), 467**

**ACLs (access control lists), 467**

ABF, 521

*configuring, 523-525*

*overview, 521*

ACEs, 467

AS\_Path, 484

categories, 468

extended

*BGP network selection, 470*

*defining, 469*

*IGP network selection, 469-470*

standard, 468-469

*ACL to network entries, 469-468*

*defining, 468*

**actions (route maps), 492**

**active neighbor field (OSPF Hello packet), [194](#)**

**Active state (BGP neighbors), [416](#)**

**AD (administrative distance), [73-76](#), [91](#)**

default values, [526](#)

floating static routes, verifying, [104-105](#)

modifying, [527](#)

*BGP*, [532-534](#)

*EIGRP*, [528-529](#)

*IS-IS*, [531-532](#)

*OSPF*, [529-531](#)

redistribution routing loop prevention, [601-603](#)

suboptimal routing, [527](#), [584](#)

**Add-Path feature (BGP), [739-742](#)**

**address families**

EIGRP configurations

*instances*, [185-186](#)

*interfaces*, [186-188](#)

*topology*, [188](#)

identifiers (AFIs)

*BGP*, [410](#)

*listing of, website*, [812](#)

*MBGP*, [812](#)

**address-family command, [77](#)**

**address-family ipv6 command, [954](#)**

**address-family ipv6 unicast autonomous-system as-number command, [954](#)**

**address-family ipv6 unicast command, [988](#), [1001](#)**

## **addresses**

anycast, [906-908](#)

*duplicate address detection, disabling, [908](#)*

*topology, [906-908](#)*

*updated topology, [907](#)*

broadcast

*calculating, [43](#)*

*defined, [37](#)*

destination, [1012](#)

host

*all-os, [37](#)*

*available, listing, [47](#)*

*first usable, calculating, [43](#)*

*last usable, calculating, [43](#)*

IP. *See* [IP addresses](#)

IPv6

*abbreviations, [895-896](#)*

*bit length values, [894](#)*

*case, [894](#)*

*components, [894](#)*

*hexadecimal to binary conversion, [896-898](#)*

*hextets, [894](#)*

*network boundaries, [894](#)*

*unicast, [894](#)*



loopback, [911](#)

MAC

*defined, [752](#)*

*multicast, [752-753](#)*

multicast

*Administratively Scoped Block, [751](#)*

*GLOP Block, [751](#)*

*IANA assigned, [749](#)*

*Internetwork Control Block, [750](#)*

*IPv6, [908-911](#)*

*Layer 2, [752-753](#)*

*Local Network Control Block, [750](#)*

*reserved, [750](#)*

*Source-Specific Multicast Block, [751](#)*

NET, [322-323](#)

*common private, [322](#)*

*expanded structure, [322](#)*

*guidelines, [323](#)*

*minimal format, [322](#)*

*multiple, [323](#)*

next-hop

*eBGP, [444-445](#)*

*MBGP for IPv6, [997](#)*

OSPF multicast, [193](#)

resolution, [934-936](#)

unicast

*link-local (LLA), 905-906*

*unique local unicast (ULA), 904-905*

unspecified, 911

**Adj-RIB-Out table neighbor-specific view, 429-430**

**adjacencies (neighbors)**

EIGRP, 128

*advertised networks update packet, 131*

*hello packet capture, 129*

*INIT flag and acknowledgment update packet, 130*

*INIT flag set update packet, 129*

*route prefixes update packet, 130*

*sample topology, 129*

*verification, 139*

EIGRPv6 verification, 949

IS-IS, 318

*broadcast, 333*

*compatibility chart, 386*

*P2P, 338-339*

*verification, 346-347*

OSPF, 197-202

*DBD packet, 199*

*debug perspective example, 201-202*

*Hello packet with neighbors detected, 198*

*Hello packet with no neighbors detected, 197*

*hub-and-spoke topology, [228](#)*

*LSR, [200](#)*

*LSU, [200](#)*

*multi-area, [304-308](#)*

*network type compatibility, [231-234](#)*

*parameters, verifying, [198-199](#)*

*process, [201](#)*

*simple topology, [197](#)*

*verifying, [209-210](#)*

OSPFv2 versus OSPFv3, [957](#)

**Adjacency Information Base (AIB), [87](#)**

**admin mode (IOS XR), [14](#)**

**administrative distance. *See* [AD](#)**

**Administratively Scoped Block, [751](#)**

**administratively scoped multicast boundaries, [863-864](#)**

**advertisements**

BGP

*conditional, [645-647](#)*

*default, [643-644](#)*

*default per neighbor, [644-645](#)*

candidate RP (C-RP-Adv), [775](#)

NDP RA messages, [913](#)

**af-interface command, [185](#)**

**af-interface interface-type interface-number command, [954](#)**

**AFI (Authority and Format Identifier), [320](#)**

## **AFIs (address family identifiers)**

BGP, [410](#)

listing of, website, [812](#)

MBGP, [812](#)

**age (OSPF LSAs), [251](#)**

**aggregate-addressipv6-prefix/prefix-length command, [1001](#)**

**aggregation. *See also* [summarization](#)**

addresses, [629-631](#)

AS\_Set, [639-641](#)

atomic, [637-639](#)

IPv6, [902-903](#)

link aggregation (LAG), [722](#)

selective AS\_Set, [641-643](#)

**AIB (Adjacency Information Base), [87](#)**

**AIGP (accumulated interior gateway protocol), [686-694](#)**

configuration, [689](#)

guidelines, [688](#)

MED conversion, [693-694](#)

metric modifications, [688](#)

neighbor sessions metrics verification, [690](#)

PAs, exchanging, [687](#)

processing logic, [692-719](#)

## **algorithms**

BGP best path, [672](#)

*AIGP*. *See* [AIGP](#)

*AS\_Path length, [694-698](#)*

*attribute classifications, [673](#)*

*attributes list, [672](#)*

*eBGP over iBGP, [714-717](#)*

*local preference, [679-682](#)*

*locally originated via network or aggregate advertisement, [683-686](#)*

*lowest IGP metric, [718-720](#)*

*lowest neighbor address, [722](#)*

*MED. See [MED](#), BGP best path selection*

*minimum cluster list length, [721](#)*

*oldest EBGP, [720](#)*

*origin types, [700-703](#)*

*RIDs, [720-721](#)*

*weight, [673-679](#)*

**BSR RP hash, [845-846](#)**

**Dijkstra shortest path first, [70](#)**

**distance vector, [69](#)**

**enhanced distance vector, [70](#)**

**link-state, [70-71](#)**

**path vector, [71-72](#)**

**RPF. See [RPF](#)**

**allowing ASNs, [658-660](#)**

**and keyword (conjunction operator), [41-42](#), [504](#)**

**any source multicast (ASM), [850](#)**

**anycast IPv6 addresses, [906-908](#)**

duplicate address detection, disabling, [908](#)

RPs, [847-849](#)

topology, [906-908](#)

updated topology, [907](#)

## **architecture**

centralized versus distributed forwarding, [86](#)

IOS, [1](#)

*kernel*, [2](#)

*memory management*, [2](#)

*run to completion scheduler*, [2](#)

*software packaging*, [2-4](#)

IOS XE, [4](#)

*kernel*, [4](#)

*memory management*, [4](#)

*scheduling*, [4](#)

IOS XR, [5](#)

*kernel*, [5](#)

*memory management*, [6](#)

*scheduling*, [5](#)

*software packaging*, [6-7](#)

IS-IS, [317](#)

multicast, [745](#)

OSPF, [192-193](#)

prefix matching, [471](#)

route policies, [499-504](#)

*advanced conditional statement nesting, [502](#)*

*bad RPL design, [503](#)*

*conditional match and action statement, [499](#)*

*conditional match with if-else logic, [500](#)*

*conditional match with if-elseif logic, [501](#)*

*conditional match with if-elseif-else logic, [501](#)*

*conditional statement nesting, [502](#)*

*good RPL design, [503](#)*

*if-elseif-else route-policy, [502](#)*

*inline policy set expansion, [500](#)*

*inline prefix filtering, [499](#)*

*nested conditional match, [503](#)*

*prefix set filtering, [500](#)*

*RFC 1918 route map with if-else logic, [501](#)*

RPL, [496](#)

**archiving configurations, [11-13](#)**

**area area-id authentication ipsec spi spi authentication-algorithm commands, [975](#)**

**area area-id encryption ipsec spi spi esp encryption-algorithm command, [975](#)**

**area area-id nssa command, [975](#)**

**area area-id range ipv6-prefix/prefix-length command, [975](#)**

**area area-id stub command, [975](#)**

**area area-id stub no-summary command, [975](#)**

**Area Border Routers. See [ABRs](#)**

**area ID field (OSPF Hello packet), [194](#)**

**areas (IS-IS), [318-319](#)**

## **areas (OSPF)**

ABRs, [242](#)

backbone, [242](#)

defined, [241](#)

disadvantages, [242](#)

failed route advertisement, [242](#)

filtering, [543-546](#)

IDs, [245-246](#)

multi-area adjacencies, [304-308](#)

*configuration*, [305-306](#)

*inefficient topologies*, [304](#)

*nonbroadcast interface verification*, [304](#)

*OSPF neighborhood*, [306-307](#)

*verification*, [306-308](#)

multi-area topology, [243-245](#)

NSSAs, [268-269](#)

OSPFv3 authentication/encryption, [972](#)

route types

*external*, [247-248](#)

*intra-area/interarea*, [246-247](#)

stubby, [286](#)

*NSSAs*, [292-295](#)

*stub areas*, [286-289](#)

*totally*, [289-292](#)

*totally NSSAs*, [295-298](#)



*types, 286*

successful route advertisement, 243

summarization, 276-280

Type 1 LSA flooding, 252

**ASBRs (Autonomous System Boundary Routers), 247, 265-268**

**AS-external LSAs, 959**

**ASM (any source multicast), 850**

**ASNs (autonomous system numbers), 408-409**

allowing, 658-660

local, 660-664

private, removing, 656-658

**AS\_Paths, 488**

ACL (IOS), 484

BGP best path algorithm, 694-698

*configuration, 696*

*processing logic, 698*

IOS XR selection options, 484

*is-local, 485*

*length, 485-486*

*neighbor-is, 487*

*originates-from option, 487*

*passes-through, 486-487*

*unique-length, 486*

**Assert PIM control message, 764**

**AS\_Set, 641-643**

**as-set keyword, [639-641](#)**

**asterisk (\*) query modifier, [482](#)**

**asynchronous mode (BFD)**

**atomic aggregate attribute, [637-639](#)**

**attached bits (LSPs), [331](#)**

**attachment points, [516](#)**

**Attempt state (OSPF neighbors), [196](#)**

## **attributes**

atomic aggregate, [637-639](#)

BGP best path, [409](#)

*AIGP*. See [AIGP](#)

*AS\_Path length*, [694-698](#)

*classifications*, [673](#)

*eBGP over iBGP*, [714-717](#)

*listing of*, [672](#)

*locally originated via networks or aggregate advertisement*, [683-686](#)

*local preference*, [679-682](#)

*lowest IGP metric*, [718-720](#)

*lowest neighbor address*, [722](#)

*MED*. See [MED](#), *BGP best path selection*

*minimum cluster list length*, [721](#)

*oldest EBGP*, [720](#)

*origin*, [700-703](#)

*RIDs*, [720-721](#)

*weight*, [673-679](#)

eBGP/iBGP multipath, [725](#)

eBGP prefix

*listing of, [441](#)*

*local, [441](#)*

*remote, [440](#)*

EIGRP, propagation, [146](#)

LSPs, [331](#)

route policies, modification, [498](#)

## **authentication**

BGP, [462-463](#)

EIGRP, [174](#)

*enabling, [174](#)*

*key chain configuration, [174-177](#)*

IS-IS, [367](#)

*configuration, [369-370](#)*

*hello, [368](#)*

*LSP, [368](#)*

*types, [367](#)*

OSPF, [236](#)

*IOS, [236](#)*

*IOS XR, [237-239](#)*

*types supported, [236](#)*

OSPFv2 versus OSPFv3, [955](#)

OSPFv3, [970-973](#)

*areas, [972](#)*

*interfaces, [972](#)*

*IPsec verification, [973](#)*

**authentication command, [975](#)**

**authentication key-chain key-chain-name command, [186](#)**

**authentication keychain key-chain-name command, [954](#)**

**authentication mode command, [186](#)**

**authentication options field (OSPF Hello packet), [194](#)**

**Authority and Format Identifier (AFI), [320](#)**

**Autonomous System Boundary Routers (ASBRs), [247](#), [265-268](#)**

**autonomous systems, [67](#)**

*EIGRP, [126](#)*

*installed routes, displaying, [140-141](#)*

*interface verification, [136-139](#)*

*IOS, [132-133](#)*

*IOS XR, [134](#)*

*neighbor adjacencies verification, [139](#)*

*passive interfaces, [134](#)*

*sample topology, [134-136](#)*

*maximum BGP, [652-654](#)*

*numbers (ASNs)*

*allowing, [658-660](#)*

*local, [660-664](#)*

*private, removing, [656-658](#)*

**Auto-RPs, [773](#)**

*candidate RPs, [773](#)*

Cisco-RP-announce message filtering, [867](#)

configuring, [785-786](#)

multicast boundaries, [865-866](#)

RP MAs, [774](#)

TTL scoping, [862](#)

**auto-summary command, [188](#)**

## **B**

**backbone, [242, 380-382](#)**

**backdoor networks, [649-652](#)**

**backup connectivity, [101](#)**

**bandwidth (EIGRP), [177-179](#)**

**bandwidth-percent percent command, [186](#)**

**BDRs (backup designated routers), [213](#)**

LSA distribution, [213](#)

OSPF

*elections, [214-216](#)*

*placement, [216-219](#)*

**Bellman-Ford algorithms, [69](#)**

**BGP (Border Gateway Protocol), [68](#)**

AD, modifying, [532-534](#)

address families, [410](#)

Adj-RIB-Out table neighbor-specific view, [429-430](#)

advertisements

*conditional, [645-647](#)*

*default, [643-644](#)*

*default per neighbor, [644-645](#)*

aggregation

*aggregate addresses, [629-631](#)*

*AS\_Set, [639-641](#)*

*atomic, [637-639](#)*

*selective AS\_Set, [641-643](#)*

allow autonomous feature, [658-660](#)

ASNs, [408-409](#)

backdoor networks, [649-652](#)

best path algorithm, [672](#)

*AIGP. See [AIGP](#)*

*AS\_Path length, [694-698](#)*

*attribute classifications, [673](#)*

*attributes list, [672](#)*

*eBGP versus iBGP, [714-717](#)*

*locally originated via network or aggregate advertisement, [683-686](#)*

*local preference, [679-682](#)*

*lowest IGP metric, [718-720](#)*

*lowest neighbor address, [722](#)*

*MED. See [MED](#), BGP best path selection*

*minimum cluster list length, [721](#)*

*oldest EBGP, [720](#)*

*origin types, [700-703](#)*

*RIDs, [720-721](#)*

*weight, [673-679](#)*

communities, [609](#)

*conditionally matching*, [620-627](#)

*formats*, [610](#)

*Internet*, [611](#)

*No\_Advertise*, [614-617](#)

*No\_Export*, [611-614](#)

*No\_Export\_SubConfed*, [617-620](#)

*support*, [611](#)

confederations, [453-458](#)

*AS100 BGP table*, [457](#)

*configuring*, [455-457](#)

*iBGP versus eBGP sessions*, [454](#)

*NLRI*, [458](#)

*topology*, [453](#)

configuring, [418-419](#)

*IOS*, [419-420](#)

*IOS XR*, [420-421](#)

defined, [407](#)

eBGP

*BGP table*, [440](#)

*configuration*, [439](#)

*iBGP combinations*, [442-444](#)

*iBGP sessions, compared*, [438](#)

*local prefix attributes*, [441](#)

*next-hop addresses*, [444-445](#)

*prefix attributes, [441](#)*

*remote prefix attributes, [440](#)*

*topology, [438](#)*

*ECMP, [723](#)*

*AS\_Path relax feature, [731-733](#)*

*eBGP/iBGP multipath, [723-726](#)*

*eiBGP multipath. See [eiBGP multipath](#)*

*failure detection, [459](#)*

*fast convergence, [125](#), [189](#), [221](#), [315](#), [366](#), [459](#)*

*filtering, [546-548](#)*

*iBGP. See [iBGP](#)*

*inter-router communication, [410-411](#)*

*keepalive messages, [413](#)*

*message types, [411](#)*

*notification messages, [414](#)*

*open messages, [412-413](#)*

*update messages, [413-414](#)*

*IPv4 neighbor output, [423-424](#)*

*local autonomous feature, [660-664](#)*

*loop prevention, [409-410](#)*

*maximum autonomous system, [652-654](#)*

*maximum prefix, [654-656](#)*

*Multiprotocol. See [MBGP](#)*

*neighbors*

*AIGP metrics support, verifying, [690](#)*



*default advertisements, [644-645](#)*

*lowest address (path selection), [722](#)*

neighbor states, [415](#)

*Active, [416](#)*

*Connect, [415-416](#)*

*Established, [417](#)*

*idle, [415](#)*

*OpenConfirm, [417](#)*

*OpenSent, [416-417](#)*

OSPF preferred over BGP for RPF calculation, [878](#)

outbound filtering, [647-649](#)

PAs, [409](#)

peer configuration

*IOS groups, [664-665](#)*

*IOS templates, [665-666](#)*

*IOS XR templates, [667-668](#)*

prefix advertisements, [425-427](#)

redistribution

*destination-specific behaviors, [580-582](#)*

*source-specific behaviors, [562-563](#)*

regex reference topology, [476](#)

remove private autonomous system feature, [656-658](#)

routes

*displaying, [428-431](#)*

*receiving, [427](#)*

*summary with prefixes, [430](#)*

RPL verification

*BGP table, [517-518](#)*

*redistribution, [516-517](#)*

security, [459](#)

*authentication, [462-463](#)*

*eBGP multihop, [459-461](#)*

*TTL, [461-462](#)*

sessions

*clearing, [549](#)*

*IPv6 over IPv4, [998-1001](#)*

*overview, [415](#)*

*verification, [421-424](#)*

suboptimal routing with RRs, [733-734](#)

*Add-Path feature, [739-742](#)*

*RRs, adding, [734-735](#)*

*shadow RRs, [735-737](#)*

*shadow session RRs, [738-739](#)*

summarization, [628](#), [632-637](#)

summary fields, [422](#)

table fields, [429](#)

**bgp router-id router-id command, [1001](#)**

**bidirectional connectivity verification, [97-98](#)**

**Bidirectional Forwarding Detection. *See* [BFD](#)**

**Bidir-PIM (bidirectional PIM)**

designated forwarders, [804-808](#)

overview, [802-808](#)

## **binary notation**

base 2 calculations, [31-32](#)

bit values, [31-32](#), [47](#)

to decimal conversion, [33](#)

decimal to binary conversions, [32](#)

defined, [31](#)

IPv6 hexadecimal conversion, [896-898](#)

subnet mask conversion, [36](#)

## **bits**

overload, [394-396](#)

values, [31-32](#), [47](#)

## **bitwise AND operator, [41-42](#)**

## **boolean operators, [504](#)**

conjunction, [504](#)

disjunction, [505](#)

negation, [504](#)

order of processing, [505-506](#)

## **Bootstrap PIM control message, [764](#)**

**bootstrap router. *See* [BSR](#)**

**Border Gateway Protocol. *See* [BGP](#)**

## **boundaries**

IPv6, [894](#), [1032-1033](#)

multicast, [863-866](#)

*administratively scoped, [863-864](#)*

*Auto-RP, [865-866](#)*

*BSR, [866](#)*

**brackets ([ ]) query modifier, [479](#)**

## **broadcast addresses**

calculating, [43](#)

defined, [37](#)

## **broadcast interfaces**

as P2P interfaces, configuring, [353-355](#)

static routing, [98-99](#)

## **broadcast networks**

IS-IS, [326, 333-338](#)

*adjacency process, [333](#)*

*IIH hello with neighbors detected, [337](#)*

*IIH hello with no neighbors detected, [335](#)*

*parameter verification, [336](#)*

*simple topology, [334](#)*

OSPF, [221](#)

video feed, [747](#)

## **BSR (bootstrap router), [775-776](#)**

configuring, [786-787](#)

IPv6 PIM-SM, [1018-1021](#)

*BSR election, [1020](#)*

*candidate RP, [1021](#)*

*embedded RP, [1021-1024](#)*

*IOS XR configuration, [1020](#)*

*RP cache, [1020](#)*

multicast boundaries, [866](#)

redundant RPs, [840-846](#)

*group filtering, [843-845](#)*

*hash algorithm, [845-846](#)*

## **C**

### **calculating**

broadcast addresses, [43](#)

CIDR best paths, [54-55](#)

EIGRP paths, [145](#)

*attribute propagation, [146](#)*

*custom K values, [148-149](#)*

*default K values, [145](#)*

*with definitions, [145](#)*

*formula, [145](#)*

*interface delay settings, [149-151](#)*

*interface metrics, [146](#)*

*load balancing, [151-153](#)*

*lowest link speed/cumulative delay, [146](#)*

*specific prefix, [147-148](#)*

*variance multiplier, [151-152](#)*

*wide metrics, [153-154](#)*

host addresses

*first usable, [43](#)*

*last usable, 43*

OSPF interface costs, 235-236

SPF

*IS-IS, 361-362*

*Type 1 LSAs, 256*

subnets

*bitwise AND operations, 41-42*

*magic number method, 42-45*

usable IP addresses, 37-38

wildcard subnet masks, 63

*/24 networks, 64*

*single IP hosts, 63*

*summary routes, 64*

**candidate RP advertisement (C-RP-Adv), 764, 775**

**candidate RPs (C-RPs), 773, 1021-1024**

**caret in brackets [^] query modifier, 480**

**carets (^) query modifier, 478**

**CEF (Cisco Express Forwarding), 86**

centralized versus distributed forwarding architectures, 86

defined, 86

distributed, 88

hardware, 88-89

IPv6, enabling, 899

software, 87

tables

*nonrecursive multihop static routes, 107*

*static route recursion problem, 114*

## **centralized forwarding architecture, 86**

### **CE router connectivity**

default route configuration, 100

ISP to, 100

## **CIDR (classless interdomain routing), 35**

best path calculation, 54-55

IP addresses interface assignments, 60

route summarization, 56-58

*no summarization example, 56*

*with summarization example, 57*

*summary prefixes, 57-58*

updates, 54

VLSMs

*classless routing update, 55*

*defined, 55*

*successful route convergence, 55-56*

## **Cisco**

Express Forwarding. *See* CEF

feature navigator tool website, 3

## **Cisco-RP-announce message filtering, 867**

## **clarity (RPL), 496**

## **classes (IP addresses)**

A, 34

B, [35](#)

C, [35](#)

D, [35](#)

history, [34](#)

listing of, [34](#)

### **classful routing**

best path confusion, [53](#)

discontiguous networks, [53](#)

local decision process, [50-51](#)

subnet masks

*uniform*, [50-51](#)

*variable-length*, [52](#)

updates, [50](#)

### **classic mode (EIGRPv6), [945](#)**

**classless interdomain routing.** *See* [CIDR](#)

**clear bgp ipv6 unicast command, [1001](#)**

### **CLI (command-line interface)**

IOS modes, [9-11](#)

*global configuration*, [10-11](#)

*privileged*, [10](#)

*user*, [9-10](#)

IOS XR modes

*admin*, [14](#)

*EXEC*, [14](#)

*global configuration*, [14](#)



## **cluster lists (BGP), [721](#)**

### **commands**

address-family, [77](#)

address-family ipv6, [954](#)

address-family ipv6 unicast, [988](#), [1001](#)

address-family ipv6 unicast autonomous-system as-number, [954](#)

af-interface, [185](#)

af-interface interface-type interface-number, [954](#)

aggregate-addressipv6-prefix/prefix-length, [1001](#)

area area-id authentication ipsec spi spi authentication-algorithm, [975](#)

area area-id encryption ipsec spi spi esp encryption-algorithm, [975](#)

area area-id nssa, [975](#)

area area-id range ipv6-prefix/prefix-length, [975](#)

area area-id stub, [975](#)

area area-id stub no summary, [975](#)

authentication, [975](#)

authentication key-chain key-chain-name, [186](#)

authentication keychain key-chain-name, [954](#)

authentication mode, [186](#)

auto-summary, [188](#)

bandwidth-percent percent, [186](#)

bgp router-id router-id, [1001](#)

clear bgp ipv6 unicast, [1001](#)

commit, [17](#)

commit label, [18](#)

default-information originate, [975](#), [988](#)

default-originate, [1001](#)

distribute-list prefix-list list-name interface-number, [954](#)

distribute-list prefix-list list-name interface-type interface-number, [954](#)

eigrp router-id router-id, [185](#)

eigrp stub, [185](#)

encryption, [975](#)

hello-interval seconds, [186](#)

hold-time seconds, [186](#)

if as-path, [497](#)

if as-path in, [497](#)

if as-path is-local, [497](#)

if destination in, [497](#)

if local-preference, [497](#)

if med, [497](#)

if next-hop in, [497](#)

if origin is, [497](#)

if tag, [497](#), [498](#)

instance instance-id, [975](#)

interface interface-type interface-number, [19.225](#), [19.579](#), [19.783](#)

ip address ip-address subnet-mask, [60](#)

ip ospf process-id area area-id, [204](#)

ipv4 address ip-address subnet mask, [60](#)

ipv4 address ipv4-address prefix-length, [60](#)

ipv6 authentication key-chain eigrp as-number key-chain-name, [954](#)

ipv6 authentication mode eigrp as-number md5, [954](#)

ipv6 router eigrp as-number, [954](#)

ipv6 router isis, [988](#)

ipv6 summary-address eigrp as-number ipv6-prefix/prefix-length, [954](#)

ip vrf forwarding vrf-name, [77](#)

ip vrf vrf-name, [77](#)

isis ipv6 metric, [988](#)

match as-path acl-number, [490](#)

match ip address, [490](#)

match ip address prefix-list prefix-list-name, [490](#)

match local-preference, [490](#)

match metric, [490](#)

match tag tag-value, [490](#)

maximum-paths, [188](#)

metric, [988](#)

metric-style, [988](#)

metric weights TOS K1 K2 K3 K4 K5, [186](#)

multitopology, [988](#)

neighbor, [1001](#)

neighbor remote-as as-number, [1001](#)

neighbor ip-address local-as-alternate-as-number,

neighbor activate, [1001](#)

net network-entity-title, [988](#)

networkipv6-prefix/prefix-length, [1001](#)

network network, [185](#)

next-hop-self, [186](#)

nssa, [975](#)

nssa [default-information-originate] no-summary, [975](#)

ospfv3 authentication, [975](#)

ospfv3 encryption, [975](#)

ospfv3 process-id area area-id, [975](#)

ospfv3 process-id area area-ID, [975](#)

passive-interface, [186](#)

prepend as-path, [498](#)

pwd, [26](#)

range ipv6-prefix/prefix-length, [975](#)

remote-as as-number, [1001](#)

replace as-path, [498](#)

root, [26](#)

route-policy route-policy-name, [954](#), [1001](#)

router bgp as-number, [1001](#)

router eigrp as-number, [132](#), [954](#)

router eigrp process-name, [954](#)

router isis, [988](#)

router ospfv3 process-id, [975](#)

set local-preference, [498](#)

set med, [498](#)

set next-hop, [498](#)

set origin, [498](#)

set weight, [498](#)

show bgp ipv6 unicast, [993](#)

show bgp ipv6 unicast neighbors, [993](#)

show bgp ipv6 unicast summary, [993](#)

show clns interface, [985](#)

show clns neighbor, [985](#)

show configuration, [15](#)

show configuration commit changes, [18](#)

show configuration commit changes incremental, [18](#)

show configuration commit list, [17](#)

show configuration failed, [20](#)

show configuration merge, [16](#)

show interface interface-type interface-number, [149](#)

show ip eigrp interface, [136](#)

show ip interface, [61](#)

show ip route vrf vrf-name, [81](#)

show ipv4 route vrf-name, [81](#)

show ipv6 mld groups, [1024](#)

show ipv6 mld interface, [1024](#)

show ipv6 mld traffic, [1024](#)

show ipv6 mrib route, [1024](#)

show ipv6 mroute, [1024](#)

show ipv6 mroute active, [1024](#)

show ipv6 pim bsr candidate-rp, [1024](#)

show ipv6 pim bsr election, [1024](#)

show ipv6 pim bsr rp-cache, [1024](#)

show ipv6 pim group-map, [1024](#)

show ipv6 pim interface, [1024](#)

show ipv6 pim neighbor, [1024](#)

show ipv6 pim range-list, [1024](#)

show ipv6 pim topology, [1024](#)

show ipv6 pim traffic, [1024](#)

show ipv6 pim tunnel, [1024](#)

show ipv6 protocols, [985](#)

show ipv6 route bgp, [993](#)

show ipv6 route ospf, [962](#)

show ipv6 rpf ipv6-address, [1024](#)

show isis database, [985](#)

show isis interface, [985](#)

show isis neighbor, [985](#)

show isis neighbors, [985](#)

show mfib route, [798](#)

show mfib route rate, [1024](#)

show mfib route src-ip-address/group-address, [886](#)

show mld groups, [1024](#)

show mld interface, [1024](#)

show mld traffic, [1024](#)

show mrrib ipv6 route, [1024](#)

show mrrib route, [798](#)

show ospfv3 database, [962](#)

show ospfv3 interface, [962](#)

show ospfv3 neighbor, [962](#)

show ospfv3 virtual-links, [962](#)

show pim ipv6 bsr candidate rp, [1024](#)

show pim ipv6 bsr election, [1024](#)

show pim ipv6 bsr rp-cache, [1024](#)

show pim ipv6 group-map, [1024](#)

show pim ipv6 interface, [1024](#)

show pim ipv6 neighbor, [1024](#)

show pim ipv6 range-list, [1024](#)

show pim ipv6 rpf ipv6-address, [1024](#)

show pim ipv6 topology, [1024](#)

show pim ipv6 traffic, [1024](#)

show pim ipv6 tunnel info, [1024](#)

show pim topology, [799](#)

show protocols ipv6, [985](#)

show route eigrp, [163-165](#)

show route ipv6 bgp, [993](#)

show route ipv6 ospf, [962](#)

show running-config, [15](#)

show the commit, [23](#)

single-topology, [988](#)

split-horizon, [186](#)

stub, [975](#)

stub no-summary, [975](#)

summary-address ipv6-prefix/prefix-length, [954](#), [975](#)

summary-address network, [186](#)

summary-address prefix/prefix-length, [988](#)

summary-metric network bandwidth delay reliability load MTU [AD], [188](#)

summary-prefix prefix/prefix-length, [988](#)

timers active-time, [188](#)

topology base, [185](#), [954](#)

variance variance-multiplier, [188](#)

vrf definition vrf-name, [77](#)

vrf upgrade-cli multi-af-mode, [77](#)

**commit command, [17](#)**

**commit label command, [18](#)**

**commits (configurations), [16-17](#)**

changes, displaying, [18](#)

commit lists, displaying, [17](#)

confirming, [22-23](#)

failures, [20-21](#)

labels, [18-19](#)

lists, displaying, [17](#)

multiple commit options, [23-24](#)

replaces, [19-20](#)

**communities (BGP), [609](#)**

conditionally matching, [620-627](#)

*community sets*, [621](#)

*inline*, [622-625](#)

*private BGP communities*, [625-627](#)



formats, [610](#)

support, [611](#)

well-known

*Internet*, [611](#)

*No\_Advertise*, [614-617](#)

*No\_Export*, [611-614](#)

*No\_Export\_SubConfed*, [617-620](#)

**complete sequence number packets (CSNPs), [350](#)**

**complex matching route maps, [491-492](#)**

**conditional matching, [490](#)**

BGP communities, [620-627](#)

*community sets*, [621](#)

*inline*, [622-625](#)

*private BGP communities*, [625-627](#)

complex matching, [491-492](#)

multiple match, [491](#)

**conditional routing**

ABF

*configuring*, [523-525](#)

*overview*, [521](#)

BGP, [645-647](#)

disadvantages, [522](#)

PBR

*configuring*, [522](#)

*local*, [525-526](#)

*overview, [521](#)*

## **confederations (BGP), [453-458](#)**

AS100 BGP table, [457](#)

configuring, [455-457](#)

iBGP versions eBGP sessions, [454](#)

NLRI, [458](#)

topology, [453](#)

## **configurations**

archiving, [11-13](#)

ABF, [523-525](#)

BGP, [418-419](#)

*authentication, [462](#)*

*confederations, [455-457](#)*

*IOS, [419-420](#)*

*IOS peer groups, [664-665](#)*

*IOS peer templates, [665-666](#)*

*IOS XR, [420-421](#)*

*IOS XR templates, [667-668](#)*

*maximum prefix, [655](#)*

*ORF, [648](#)*

BSR, [786-787](#)

changes

*displaying in SysDB, [17-18](#)*

*rolling back, [21-22](#)*

commits, [16-17](#)

*changes, displaying, [18](#)*

*confirming, [22-23](#)*

*failures, [20-21](#)*

*labels, [18-19](#)*

*list, displaying, [17](#)*

*multiple commit options, [23-24](#)*

*replaces, [19-20](#)*

default routes

*CE router, [100](#)*

*XR1/R3 example, [101-102](#)*

DNS SSM mapping, [859](#)

eBGP, [438](#), [439](#)

*multihop sessions, [460](#)*

*multipath, [724](#)*

EIGRP

*bandwidth percent, [178](#)*

*installed routes, displaying, [140-141](#)*

*interface delays, [150](#)*

*interface verification, [136-139](#)*

*IOS, [132-133](#)*

*IOS XR, [134](#)*

*key chains, [174-177](#)*

*named configurations. See [EIGRP, named configurations](#)*

*neighbor adjacencies verification, [139](#)*

*passive interfaces, [134](#)*

*sample topology, [134-136](#)*

*stubs, [162-163](#)*

EIGRPv6

*classic mode, [945](#)*

*IOS XR, [946-947](#)*

*named mode, [946](#)*

*external route summarization, [282](#)*

*files, loading, [24](#)*

*hierarchical, [25-26](#)*

*iBGP, [724](#)*

*interarea route summarization, [279](#)*

*IOS relay agents, [927-928](#)*

IOS XR

*relay proxy agents, [928-929](#)*

*two-stage commit, [14-17](#)*

IPv6

*multicast, [1025](#)*

*static, [942-943](#)*

*unicast, [899](#)*

IS-IS, [340](#)

*authentication, [369-370](#)*

*broadcast interfaces as P2P, [353-355](#)*

*default routes, [400](#)*

*DISs, [352](#)*

*interfaces, verifying, [343-346](#)*

*IOS, 340*

*IOS XR, 340-341*

*metrics, 391*

*neighbor adjacencies, verifying, 346-347*

*overload bits, 395*

*prefix suppression, 403*

*route-leaking, 377*

*route verification, 347-348*

*summarization, 399*

*topology example, 341-343*

*IS-IS for IPv6, 979*

*IOS base, 979-980*

*IOS topology mode, 981*

*IOS XR base, 980*

*IOS XR topology mode, 981-984*

*MBGP, 815*

*IOS, 991-992*

*IOS XR, 992-993*

*merging, 16*

*multicast, 780-782, 786-787*

*multiple recursive lookups, 110*

*nested, exiting, 26*

*nonrecursive multihop static routes, 107*

*OSPF*

*IOS interface specific, 204-205*

*IOS network statement, [202-204](#)*

*IOS XR, [205](#)*

*multi-area adjacencies, [305-306](#)*

*NSSAs, [293-294](#)*

*point-to-multipoint network, [226](#)*

*prefix suppression, [310](#)*

*stub areas, [287-288](#)*

*topology sample, [206-208](#)*

*totally NSSAs, [297](#)*

*totally stubby areas, [290-291](#)*

*virtual links, [300](#)*

OSPFv3

*IOS, [960-961](#)*

*IOS XR, [961](#)*

PBR, [522](#)

prompts

*IOS, [26](#)*

*IOS XR, [26](#)*

replacing, [13](#)

route reflectors, [449-450](#)

RPs, [783-787](#)

*Auto-RPs, [785-786](#)*

*static, [784-785](#)*

running, displaying, [15](#)

SSM, [852-853](#), [860](#)

stateful DHCPv6

*IOS*, [930](#)

*IOS XR*, [931-932](#)

stateless DHCPv6

*IOS*, [924-925](#)

*IOS XR*, [925-926](#)

target, displaying, [15](#)

VRFs

*example*, [81](#)

*IOS*, [78](#)

*IOS XR*, [78-79](#)

**confirmed parameter**, [23](#)

**confirming commits**, [22-23](#)

**conjunction operator**, [504](#)

**connected networks**

defined, [91](#)

local routes, [93](#)

redistribution, [561](#)

routing table sample, [92-93](#)

sample topology, [92](#)

secondary, [94](#)

specific network routing detail output sample, [93](#)

**connections**

BGP, clearing, [549](#)

ES-IS/IS-IS, [316](#)

OSPF prefix suppression, verifying, [312](#)

**Connect state (BGP neighbors), [415-416](#)**

**continue keyword, [493-494](#)**

EIGRP, [158](#)

**C-RP-Adv (candidate RP advertisement), [775](#)**

**C-RPs (candidate RPs), [773](#)**

**CSNPs (complete sequence number packets), [350](#)**

## **D**

**DAD (duplicate address detection), [937-938](#)**

## **databases**

LSDB, [70](#), [192](#)

*OSPF Type 1 LSA fields, [252](#)*

*OSPFv3, [965-969](#)*

LSPDB (IS-IS)

*building topology, [359-360](#)*

*displaying, [356-357](#)*

*displaying topology, [360-361](#)*

*non-pseudonode LSPs, [357-358](#)*

*pseudonode LSPs, [358-359](#)*

*SPF calculations, [361-362](#)*

*without transit networks, [404-405](#)*

**data plane, [89](#)**

## **DBD packets**

OSPF, [194](#)

OSPFv3, [957](#)



**dCEF (distributed CEF), [88](#)**

**dead interval field (OSPF Hello packet), [194](#)**

**dead interval timer, [219](#)**

**debugging IOS XR traces, [8](#)**

**default-information originate command, [975](#), [988](#)**

**default-originate command, [1001](#)**

**default routes, [99-102](#)**

backup connectivity, [101](#)

BGP advertisements

*overview, [643-644](#)*

*per neighbor, [644-645](#)*

CE router

*configuration, [100](#)*

*routing tables, [100-101](#)*

EIGRPv6, [952-953](#)

IS-IS, [400-401](#)

OSPF summarization, [283-285](#)

XR1/R3

*configuration example, [101-102](#)*

*routing tables, [102](#)*

**Dense Mode (PIM), [765-767](#)**

**design**

EIGRP stubs, [164-166](#)

subnetting, [46](#)

*available host values, listing, [47](#)*

*available subnets, listing, [47](#)*

*binary bit values, listing, [47](#)*

*final address allocation scheme, [49](#)*

*LAN 1/LAN 2 subnet assignments, [47-48](#)*

*LAN 3 assignments, [48](#)*

**designated forwarders (DFs), [764](#), [804-808](#)**

**designated intermediate systems. *See* [DISs](#)**

**designated router & backup designated router field (OSPF Hello packet), [194](#)**

**designated routers. *See* [DRs](#)**

**destination addresses (MLD), [1012](#)**

**destination-specific redistribution behaviors**

*BGP, [580-582](#)*

*EIGRP, [563-566](#)*

*EIGRP-to-EIGRP, [566-568](#)*

*IS-IS, [576-578](#)*

*IS-IS to IS-IS, [578-580](#)*

*OSPF, [569-571](#)*

*forwarding addresses, [573-576](#)*

*OSPF-to-OSPF, [571-572](#)*

**DFs (designated forwarders), [764](#), [804-808](#)**

**DHCPv6**

*stateful, [926-927](#)*

*IOS relay agent configuration, [927-928](#)*

*IOS relay agent verification, [928](#)*

*IOS server configuration, [930](#)*

*IOS XR relay proxy agent configuration, [928-929](#)*

*IOS XR relay proxy agent verification, [929](#)*

*IOS XR server configuration, [931-932](#)*

*server verification, [932-934](#)*

*topology, [926](#)*

*stateless, [924-926](#)*

*IOS configuration, [924-925](#)*

*IOS XR configuration, [925-926](#)*

*verification, [926](#)*

**Diffused Update Algorithm (DUAL), [70](#)**

**Dijkstra shortest path first algorithm, [70](#)**

**discard routes, [399](#)**

**discontiguous networks, [301-303](#)**

**disjunction operator, [505](#)**

**displaying**

BGP

*routes, [428-431](#)*

configurations

*changes in SysDB, [17-18](#)*

*commit changes, [18](#)*

*commit lists, [17](#)*

*running, [15](#)*

*target, [15](#)*

EIGRP routes, [140-141](#)

interfaces/IP addresses status verification, [82-83](#)

IOS configuration archive, [12](#)

IPv4 interfaces

*specific*, [61](#)

*summaries*, [62](#)

IS-IS

*LSPDB*, [356-359](#)

*LSP topology*, [360-361](#)

NDP parameters, [916-917](#)

OSPFv3 LSDB summary view, [969](#)

route policy states, [515-516](#)

VRFs, [81-82](#)

**DISs (designated intermediate systems), [348](#)**

configuration, [352](#)

CSNPs, [350](#)

elections, [351](#)

logical drawing, [348](#)

LSP advertisement, [349](#)

placement, [352-353](#)

pseudonode LSPs, [349-350](#)

verification, [353](#)

**distance vector algorithms, [69](#)**

**distributed CEF (dCEF), [88](#)**

**distributed forwarding architecture, [86](#)**

**distribute-list prefix-list list-name interface-number command, [954](#)**

**distribute-list prefix-list list-name interface-type interface-number command, [954](#)**

## **distribution trees (multicast), [759](#)**

shared trees, [761](#)

source trees, [759-760](#)

## **DNS**

SLAAC RA options, [923-924](#)

SSM mapping, [857-860](#)

*configuration, [859](#)*

*topology, [858](#)*

*verification, [859](#)*

stateless DHCPv6, [924-926](#)

*IOS configuration, [924-925](#)*

*IOS XR configuration, [925-926](#)*

*verification, [926](#)*

**DNSSEC (DNS Search List), [923](#)**

**dollar sign (\$) query modifier, [478-479](#)**

**Domain Specific Part (DSP), [320-321](#)**

**done messages (MLD), [1010](#)**

## **dot-decimal notation**

binary conversions to, [33](#)

converting to binary, [32](#)

defined, [30](#)

octet subtraction, [63](#)

*/24 networks, [64](#)*

*single IP hosts, [63](#)*

*summary routes, [64](#)*

wildcard subnet masks, [62](#)

**Down state (OSPF neighbors), [196](#)**

**downstream interface, [763](#)**

**DRs (designated routers), [213](#)**

election, verifying, [795](#)

OSPF, [213](#)

*elections, [214-216](#)*

*LSA distribution, [213](#)*

*placement, [216-219](#)*

PIM-SM, [772](#)

Type 2 LSAs, [257-259](#)

**DSP (Domain Specific Part), [320-321](#)**

**DUAL (Diffused Update Algorithm), [70](#)**

**dual stacks, [893](#)**

**duplicate address detection (DAD), [937-938](#)**

**dynamic routing protocols, [68](#)**

administrative distance, [73-76](#)

distance vector, [69](#)

enhanced distance vector, [70](#)

link-state, [70-71](#)

listing of, [68](#)

path vector, [71-72](#)

routing table metrics, [75-76](#)

**E**

**eBGP**

BGP table, [440](#)

configuration, [439](#)

eiBGP multipath, [726-733](#)

*core deciding path*, [728](#)

*edge routers*, [727](#)

*topology*, [728-731](#)

iBGP

*combinations*, [442-444](#)

*sessions, compared*, [438](#)

multihop sessions, [459-461](#)

multipath, [723-726](#)

*attributes*, [725](#)

*configuration*, [724](#)

next-hop addresses, [444-445](#)

oldest path, [720](#)

over iBGP, [714-717](#)

*configuration*, [715](#)

*processing logic*, [717](#)

*routing table*, [717](#)

prefix attributes

*listing of*, [441](#)

*local*, [441](#)

*remote*, [440](#)

topology, [438](#)

**ECMP (equal-cost multipath), [75](#)**

BGP, [723](#)

*AS\_Path relax feature*, [731-733](#)

*eBGP/iBGP multipath*, [723-726](#)

*eiBGP multipath*, [726-733](#)

IS-IS, [387](#)

OSPF path selection, [274](#)

## **EGPs (exterior gateway protocols)**

defined, [68](#)

path selection algorithms

*distance vector*, [69](#)

*enhanced distance vector*, [70](#)

*link-state*, [70-71](#)

*path vector*, [71-72](#)

### **eiBGP multipath, [726-733](#)**

core deciding path, [728](#)

edge routers, [727](#)

topology, [728-731](#)

## **EIGRP (Enhanced Interior Gateway Routing Protocol), [68](#), [125](#)**

AD, modifying, [528-529](#)

authentication, [174](#)

*enabling*, [174](#)

*key chain configuration*, [174-177](#)

autonomous system configuration, [126](#)

*installed routes, displaying*, [140-141](#)

*interface verification*, [136-139](#)



*IOS, 132-133*

*IOS XR, 134*

*neighbor adjacencies verification, 139*

*passive interfaces, 134*

*sample topology, 134-136*

convergence, 158

feasibility condition, 142

feasible distance (FD), 142

feasible successor, 142

filtering

*hop counts, 538*

*offset lists, 538-541*

*prefix, 534-537*

inter-router communication, 127

link failures, 156-157

metrics, 145

*attribute propagation, 146*

*custom K values, 148-149*

*default K values, 145*

*with definitions, 145*

*formula, 145*

*interface delay settings, 149-151*

*interface metrics, 146*

*load balancing, 151-153*

*lowest link speed/cumulative delay, 146*

*specific prefix, [147-148](#)*

*variance multiplier, [151-152](#)*

*wide, [153-154](#)*

named configurations, [184](#)

*address family instance configuration, [185-186](#)*

*address family interface configuration, [186-188](#)*

*address family topology configuration, [188](#)*

*benefits, [184](#)*

neighbors

*advertised networks update packet, [131](#)*

*forming, [128-131](#)*

*hello packet capture, [129](#)*

*INIT flag and acknowledgment update packet, [130](#)*

*INIT flag set update packet, [129](#)*

*overview, [126](#)*

*requirements, [128](#)*

*route prefixes update packet, [130](#)*

*sample topology, [129](#)*

*verification, [139](#)*

packet types, [127](#)

PDMs, [126](#)

queries

*boundaries, establishing, [157](#)*

*packets, handling, [157](#)*

redistribution, [563-566](#)

reference topology, [142](#)

reported distance (RD), [142](#)

RIDs, [141](#)

RTP, [127](#)

stubs, [160](#)

*default configuration, [162-163](#)*

*designs, [164-166](#)*

*inter-region traffic, [161-162](#)*

*regional/remote topology, [161](#)*

*route tables after link failures, [163](#)*

successors, [142](#)

summarization

*automatic, [172-174](#)*

*hierarchical, [166](#)*

*interface-specific, [166-171](#)*

*metrics, [171-172](#)*

timers

*hello/hold, [155-156](#)*

*SIA, [159-160](#)*

topology table, [143-144](#)

WANs, [177](#)

*bandwidth percent, [177-179](#)*

*next-hop self behavior, [182-183](#)*

*split horizons, [179-182](#)*

**egrip router-id router-id command, [185](#)**

**eigrp stub command, [185](#)**

**EIGRP-to-EIGRP redistribution, [566-568](#)**

**EIGRPv6, [944](#)**

configuration

*classic mode, [945](#)*

*IOS XR, [946-947](#)*

*named mode, [946](#)*

configuration command reference chart, [954](#)

default routes, [952-953](#)

inter-router communication, [944](#)

route filtering, [953-954](#)

summarization, [950-952](#)

verification, [947-950](#)

*base configuration, [948-949](#)*

*neighbor adjacency, [949-950](#)*

*routing table, [950](#)*

*show commands, [947](#)*

*topology, [948](#)*

**elections**

DISs, [351](#)

DRs/DBRs, [214-216](#)

**embedded RP, [1021-1024](#)**

**encryption command, [975](#)**

**end systems (ES), [316](#)**

**enhanced distance vector algorithms, [70](#)**

**Enhanced Interior Gateway Routing Protocol.** *See* [EIGRP](#)

**Equal Cost Multi-Path.** *See* [ECMP](#)

**ES (end systems),** [316](#)

**Established state (BGP neighbors),** [417](#)

**Ethernet link failures**

floating static routes, [104](#)

static route recursion problems, [113-114](#)

**EUI-IDs (extended unique identifiers),** [920-921](#)

**Exchange state (OSPF neighbors),** [196](#)

**EXEC mode (IOS XR),** [14](#)

**exiting nested configurations,** [26](#)

**explicit IP addresses,** [133](#)

**explicit subnets,** [133](#)

**explicit tracking (MLD),** [1014](#)

**ExStart state (OSPF neighbors),** [196](#)

**extended ACLs**

BGP network selection, [470](#)

defining, [469](#)

IGP network selection, [469-470](#)

**extended BGP communities,** [610](#)

**extended unique identifiers (EUI-IDs),** [920-921](#)

**exterior gateway protocols.** *See* [EGPs](#)

**external routes,** [247-248](#)

IS-IS, [386](#)

OSPF path selection, [272](#)

OSPF summarization, [280-283](#)

*concept*, [280](#)

*configuration*, [282](#)

*loop-prevention routers*, [283](#)

*routing table before summarization*, [281](#)

*topology*, [281](#)

Type 1, [273](#)

Type 2, [273-274](#)

## **F**

### **failure detection**

BGP, [459](#)

IS-IS, [366](#)

*hello multiplier*, [367](#)

*hello timers*, [366-367](#)

*holding timer*, [367](#)

OSPF, [219](#)

*dead interval timer*, [219](#)

*fast-packet Hellos*, [220-221](#)

*Hello timer*, [219](#)

*timer verification*, [219](#)

### **failures**

area route advertisement, [242](#)

commit, [20-21](#)

EIGRP

*route tables after link failures with stubs*, [163](#)

*link*, [156-157](#)

fast reroutes. *See* FRR

full-mesh iBGP link, [433](#)

OSPF interarea summarization, [277](#)

**fast-packet Hellos**, [220-221](#)

**fast reroute**. *See* FRR

**FD (feasible distance)**, [142](#)

**feasibility condition**, [142](#)

**feasible successors**, [142](#)

**feature navigator tool**, [3](#)

**feature sets, identifying**, [2-3](#)

**FHR (first-hop router)**, [763](#)

**FIB (Forwarding Information Base)**, [87](#)

**fields**

BGP

*summary*, [422](#)

*table*, [429](#)

IGMPv2 messages, [757](#)

IIH packets, [328](#)

IS-IS neighbor states, [346](#)

LSP lifetime, [329](#)

NSAP

*DSP*, [321](#)

*IDP*, [320](#)

OSPF

*Hello packet, [194](#)*

*LSDB, [252](#)*

OSPFv3 Options bit field, [966](#)

Type 3 LSAs, [262](#)

Type 4 LSAs, [267](#)

Type 5 LSAs, [265](#)

Type 7 LSAs, [269](#)

ULA, [904](#)

**filter-autorp keyword, [864](#)**

**filtering routes, [534](#)**

Auto-RP Cisco-RP-announce messages, [867](#)

Auto-RP group, [836-840](#)

BGP, [546-548](#), [647-649](#)

BSR multiple RPs, [843-845](#)

EIGRP

*hop counts, [538](#)*

*offset lists, [538-541](#)*

*prefix, [534-537](#)*

EIGRPv6, [953-954](#)

IS-IS, [546](#)

OSPF

*areas, [543-546](#)*

*local, [541-543](#)*

PIM-SM source registration, [867-868](#)

**Finite State Machine (FSM), [415](#)**



**first-hop router (FHR), [763](#)**

**first usable host addresses, calculating, [43](#)**

**flexible route suppression**

leaking suppressed routes, [634-637](#)

selective prefix, [632-634](#)

**floating static routes, [103-105](#)**

AD verification, [104-105](#)

backups, configuring, [112](#)

defined, [103](#)

Ethernet link failures, [104](#)

XR1/R3

*configuration, [103](#)*

*routing table, [103-104](#)*

**flooding**

OSPF LSAs, [251](#)

*Type 1, [252](#)*

*Type 5, [263](#)*

OSPFv3 scopes, [959-960](#)

**Ford-Fulkerson algorithms, [69](#)**

**forwarding**

access-list-based (ABF), [521](#)

*configuring, [523-525](#)*

*overview, [521](#)*

bidirectional detection (BFD)

Cisco Express (CEF)

*centralized versus distributed forwarding architectures, [86](#)*

*defined, [86](#)*

*distributed, [88](#)*

*hardware, [88-89](#)*

*IPv6, enabling, [899](#)*

*software, [87](#)*

*tables, [107-114](#)*

continuous. *See* continuous forwarding

Information Base (FIB), [87](#)

nonstop. *See* NSF

multicast forwarding information base (MFIB), [764](#)

reverse path. *See* [RPF](#)

virtual routing and forwarding. *See* [VRF](#)

**fragment IDs (LSP IDs), [330](#)**

**frame relay interfaces, [222-223](#)**

**FSM (Finite State Machine), [415](#)**

**full-mesh topology (iBGP), [432-433](#)**

**Full state (OSPF neighbors), [196](#)**

## **G**

**gateways, [30](#)**

**general topology networks, [327](#)**

**global configuration mode**

IOS, [10-11](#)

IOS XR, [14](#)

**global routing tables**

IP address configuration, [80](#)

output, [80-81](#)

**global unicast addresses. See [GUAs](#)**

**GLOP Block, [751](#)**

**group membership LSAs, [959](#)**

**groups**

IPv6 multicast, [910-911](#)

MLD, [1013](#)

**GUAs (global unicast addresses), [900](#)**

address allocation hierarchy, [900-901](#)

prefixes

*bit boundaries, [903](#)*

*list website, [900](#)*

route aggregation, [902-903](#)

subnetting chart, [903](#)

**H**

**hardware CEF, [88-89](#)**

**hello authentication (IS-IS), [368](#)**

**hello interval field (OSPF Hello packet), [194](#)**

**hello-interval seconds command, [186](#)**

**hello multiplier (IS-IS failure detection), [367](#)**

**Hello packets**

area IDs, [245](#)

fast-packet, [220-221](#)

IIH, [327-328](#)

*fields, [328](#)*

*padding removing, [365](#)*

*TLVs, [328](#)*

*types, [327](#)*

IS-IS, [324](#)

OSPF, [194-195](#), [245](#)

OSPFv3, [957](#)

**Hello PIM control message, [764](#)**

**hello timers**

EIGRP, [155-156](#)

IS-IS failure detection, [366-367](#)

OSPF failure detection, [219](#)

**hexadecimal addresses**

IPv6 binary conversion, [896-898](#)

**hexets, [894](#)**

**hierarchy**

configurations, [25-26](#)

EIGRP summarization, [166](#)

GUA address allocation, [900-901](#)

IS-IS, [317](#)

OSPF, [192-193](#)

**high availability**

**HO-DSP (High Order DSP), [321](#)**

**hold timers, [367](#)**

BGP, [412](#)

EIGRP, [155-156](#)

**hold-time seconds command, [186](#)**

**hop counts (EIGRP filtering), [538](#)**

## **hosts**

addresses

*all-os, [37](#)*

*available, listing, [47](#)*

*first usable, calculating, [43](#)*

*last usable, calculating, [43](#)*

identifiers, [30](#)

subnet mask reference chart, [40](#)

**hyphen (-) query modifier, [479-480](#)**

## **I**

### **IANA (Internet Assigned Numbers Authority)**

AFI listing website, [812](#)

IPv6

*address blocks, [911](#)*

*multicast addresses, [910](#)*

multicast addresses, [749](#)

private address ranges, [58](#)

public ASNs, [408](#)

special purpose reserved addresses, [59](#)

### **iBGP**

custom routing, [432](#)

eBGP

*combinations, [442-444](#)*

*sessions, compared, [438](#)*

*eiBGP multipath, [726-733](#)*

*core deciding path, [728](#)*

*edge routers, [727](#)*

*topology, [728-731](#)*

*full-mesh requirement, [432-433](#)*

*multipath, [723-726](#)*

*attributes, [725](#)*

*configuration, [724](#)*

*over eBGP, [714-717](#)*

*configuration, [715](#)*

*processing logic, [717](#)*

*routing table, [717](#)*

*path attributes, [432](#)*

*peering via loopback addresses, [433-437](#)*

*BGP table, [437](#)*

*configuration source, [435-436](#)*

*IPv4 session summary, [437](#)*

*prefix advertisement behavior, [431](#)*

*scalability, [432, 446](#)*

*confederations, [453-458](#)*

*loop prevention in route reflectors, [451-452](#)*

*out-of-band route reflectors, [453](#)*

*route reflectors, [446-451](#)*

**ICMPv6, [919-920](#)**

**identifying**

active licenses, [3-4](#)

feature sets, [2-3](#)

sender network IDs, [36-37](#)

software versions, [2-3](#)

universal image versions, [3-4](#)

**IDI (Initial Domain Identifier), [320](#)**

**Idle state (BGP neighbors), [415](#)**

**IDP (Inter-Domain Part), [320](#)**

**IDs**

extended unique (EUI-IDs), [920-921](#)

LSPs, [330](#)

OSPF areas, [245-246](#)

router (RIDs)

*BGP, [413, 720-721](#)*

*EIGRP, [141](#)*

*EIGRPv6, [945](#)*

*OSPF, [196](#)*

*OSPFv2 versus OSPFv3, [955](#)*

sender network, [36-37](#)

**IETF (Internet Engineering Task Force), [59](#)**

**if as-path command, [497](#)**

**if as-path in command, [497](#)**

**if as-path is-local command, [497](#)**

**if destination in command, [497](#)**

**if local-preference command, [497](#)**

**if med command, [497](#)**

**if next-hop in command, [497](#)**

**if origin is command, [497](#)**

**if tag command, [497](#), [498](#)**

**IGMP (Internet Group Management Protocol), [745](#), [753](#)**

snooping, [753-756](#)

static joins, [882-886](#)

*IGMPv2 configuration, [884-885](#)*

*IGMPv3 configuration, [884-885](#)*

*mroute table, [885-886](#)*

v2, [757-758](#), [780-782](#)

v3, [753](#), [759](#)

**IGPs (interior gateway protocols)**

defined, [68](#)

lowest IGP metric (BGP path selection), [718-720](#)

path selection algorithms

*distance vector, [69](#)*

*enhanced distance vector, [70](#)*

*link-state, [70-71](#)*

*path vector, [71-72](#)*

**IIH (IS-IS hello) packets, [327-328](#)**

fields, [328](#)

IS-IS interface frequency, [393](#)



padding, removing, [365](#)

TLVs, [328](#)

types, [327](#)

**include-connected key word, [1004](#)**

**incoming interfaces, [763](#)**

**incremental SPF. *See* [iSPF](#)**

**INIT flag sets (EIGRP update packet), [129](#)**

**Initial Domain Identifier (IDI), [320](#)**

**Init state (OSPF neighbors), [196](#)**

**in keyword (ip multicast boundary access-list command), [864](#)**

**inline BGP community conditional matching, [622-625](#)**

**inline policy set expansion, [500](#)**

**inline prefix filtering route policy, [499](#)**

**instance instance-id command, [975](#)**

**instances**

EIGRP address family configuration, [185-186](#)

multiple

*OSPFv2 versus OSPFv3, [957](#)*

*OSPFv3, [973-975](#)*

**Integrated IS-IS, [323](#)**

**interarea routes, [246-247](#)**

IS-IS, [386](#)

LSAs, [959](#)

OSPF path selection, [272](#)

OSPF summarization, [276-280](#)

*concept, [276](#)*

*configuring, [279](#)*

*example, [278](#)*

*failure, [277](#)*

*loop-prevention routes, [280](#)*

*metrics, [277](#)*

*routing table after summarization, [279](#)*

*routing table before summarization, [278](#)*

*prefix LSAs, [959](#)*

## **interdomain multicast routing**

**MBGP, [812](#)**

*AFIs/SAFIs, [812](#)*

*BGP topology, [813-814](#)*

*configuration, [815](#)*

*multicast BGP session summary verification, [817](#)*

*unicast BGP session summary verification, [816](#)*

*update message packet capture, [813](#)*

**MSDP, [817-818](#)**

*keepalive messages, [819-821](#)*

*peers, [822-828](#)*

*SA messages, [818-819](#)*

*stub networks, [831-833](#)*

*verification, [828-831](#)*

**Inter-Domain Part (IDP), [320](#)**

**interface address mask field (OSPF Hello packet), [194](#)**

**interface interface-type interface-number command, [954](#), [975](#), [988](#)**

**interface priority field (OSPF Hello packet), [194](#)**

## **interfaces**

broadcast

*as P2P, configuring, [353-355](#)*

*static, [98-99](#)*

EIGRP

*address family configuration, [186-188](#)*

*authentication, enabling, [174](#)*

*delay settings, [149-151](#)*

*path calculation metrics, [146](#)*

*summarization, [166-171](#)*

*verifying, [136-139](#)*

IP addresses, assigning

*CIDR notation, [60](#)*

*full subnet masks, [60](#)*

*secondary addresses, [60-61](#)*

*SLAAC, [895-924](#)*

IPv4

*specific, displaying, [61](#)*

*summaries, displaying, [62](#)*

IPv6 status, [899-900](#)

IS-IS

*IIH frequency, [393](#)*

*metric, [387-394](#)*

*passive, [361-362](#)*

*specific levels, [385-386](#)*

*verifying, [343-346](#)*

*loopback, [433-437](#)*

null

*defined, [116](#)*

*packet traces demonstrating routing loops, [118-119](#)*

*preventing routing loops, [119-120](#)*

*routing loops, [117-118](#)*

OSPF

*areas. OSPF, areas*

*costs, [235-236](#)*

*frame relay, [222-223](#)*

*non-broadcast, [223, 301-303](#)*

*P2P, [225](#)*

*point-to-multipoint network type, [226-227](#)*

*verifying, [208-209](#)*

OSPFv3

*authentication/encryption, [972](#)*

*verification, [965](#)*

*overlapping IP addresses, [76-77](#)*

P2P

*broadcast interfaces as, configuring, [353-355](#)*

*static routing, [96-98](#)*

passive

*EIGRP, 134*

*OSPF, 205-206*

PIM

*downstream, 763*

*incoming, 763*

*RPF, 762*

*upstream, 763*

*verification, 791*

status verification, displaying, *82-83*

VRFs, assigning

*IOS, 78*

*IOS XR, 78-79*

**interior gateway protocols. *See IGPs***

**intermediate systems (IS), *316***

**Intermediate System-to-Intermediate System. *See IS-IS***

**Internet**

Assigned Numbers Authority. *See IANA*

BGP community, *611*

Engineering Task Force (IETF), *59*

Group Management Protocol. *See IGMP*

**Internetwork Control Block, *750***

**Internetworking Operating System. *See IOS***

**inter-region network connectivity, *161-162***

**inter-router communication, *957***

BGP, *410-411*

*keepalive messages, [413](#)*

*message types, [411](#)*

*notification messages, [414](#)*

*open messages, [412-413](#)*

*sessions, [415](#)*

*update messages, [413-414](#)*

EIGRP, [127](#)

EIGRPv6, [944](#)

IPv6, [978](#)

IS-IS, [323-325](#)

*IIH packets, [327-328](#)*

*IS PDU addressing, [326-327](#)*

*IS protocol headers, [325](#)*

*LSPs, [329-332](#)*

*neighbors. See [IS-IS, neighbors](#)*

*TLVs, [326](#)*

MBGP for IPv6, [989-990](#)

OSPF, [193-194](#)

*Hello packets, [194-195](#)*

*multicast addresses, [193](#)*

*packet types, [194](#)*

**intra-area prefix, [959](#)**

**intra-area routes, [246-247](#)**

IS-IS, [386](#)

OSPF path selection, [271-272](#)

## **IOS (Internetworking Operating System), 1**

administratively scoped multicast boundaries, 864

AS\_Path ACL, 484

BGP

*best path weight*, 678

*configuration scalability*, 664-666

*configuring*, 419-420

*identifier*, 413

CLI modes, 9-11

*global configuration*, 10-11

*privileged*, 10

*user*, 9-10

configurations

*archiving*, 11-13

*prompts*, 26

*replacing*, 13

DNS SSM mapping, enabling, 858

eBGP configuration, 438

EIGRP

*autonomous system configuration*, 132-133

*RIDs*, 141

EIGRPv6

*configuration*, 945-946

*route filtering*, 953-954

IS-IS

*configuring, [340](#)*

*route leaking, [377](#)*

IS-IS for IPv6

*base configuration, [979-980](#)*

*topology configuration, [981](#)*

kernel, [2](#)

memory management, [2](#)

MBGP for IPv6 configuration, [991-992](#)

MSDP

*MSDP Compliance, [823](#)*

*peer configuration, [822](#)*

OSPF

*authentication, [236](#)*

*interface specific configuration, [204-205](#)*

*network statement, [202-204](#)*

OSPFv3 configuration, [960-961](#)

PBR, [521](#)

*configuring, [522](#)*

*local, [525-526](#)*

*overview, [521](#)*

PIM-SM accept RP, [868-869](#)

prefix lists, [473](#)

private BGP communities, setting, [625-627](#)

process switching, [84](#)

run to completion scheduler, [2](#)



show ip mroute command flags, [794](#)

SLAAC configuration, [922](#)

software packaging, [2-4](#)

SSM, enabling, [851](#)

stateful DHCPv6

*relay agent configuration*, [927-928](#)

*relay agent verification*, [928](#)

*server configuration*, [930](#)

stateless DHCPv6 configuration, [924-925](#)

static route, configuring, [96](#)

summary, [7](#)

VRFs, creating, [78](#)

## **IOSd, [4](#)**

### **IOS XE, [4](#)**

kernel, [4](#)

memory management, [4](#)

scheduling, [4](#)

summary, [7](#)

### **IOS XR, [5](#)**

ABF, [521](#)

*configuring*, [523-525](#)

*overview*, [521](#)

AS paths, [484](#)

*is-local*, [485](#)

*length*, [485-486](#)

*neighbor-is, [487](#)*

*originates-from, [487](#)*

*passes-through, [486-487](#)*

*sets, [488](#)*

*unique-length, [486](#)*

BGP

*best path weight, [678](#)*

*community conditional matching, [621-625](#)*

*configuring, [420-421](#)*

*identifier, [413](#)*

*peer configuration, [667-668](#)*

commits

*confirming, [22-23](#)*

*failures, [20-21](#)*

*labels, [18-19](#)*

*multiple commit options, [23-24](#)*

*replaces, [19-20](#)*

configurations

*change rollbacks, [21-22](#)*

*displaying changes in SysDB, [17-18](#)*

*files, loading, [24](#)*

*hierarchical, [25-26](#)*

*nested, exiting, [26](#)*

*processing changes, [14-17](#)*

*prompts, [26](#)*

eBGP configuration, [438](#)

EIGRP

*autonomous system configuration, [134](#)*

*RIDs, [141](#)*

EIGRPv6

*configuration, [946-947](#)*

*route filtering, [954](#)*

IPv6 BSR PIM-SM configuration, [1020](#)

IS-IS

*configuring, [340-341](#)*

*route leaking, [377](#)*

IS-IS for IPv6

*base configuration, [980](#)*

*topology mode configuration, [981-984](#)*

kernel, [5](#)

MBGP for IPv6 configuration, [992-993](#)

memory management, [6](#)

modes

*admin, [14](#)*

*EXEC, [14](#)*

*global configuration, [14](#)*

MSDP peer configuration, [822-823](#)

multicast boundary scope configuration, [1032](#)

OSPF

*authentication, [237-239](#)*

*configuring, [205](#)*

OSPFv3 configuration, [961](#)

prefix

*lists, [473](#)*

*sets, [474-475](#)*

process switching, [84](#)

scheduling, [5](#)

software packaging, [6-7](#)

SSM, enabling, [851](#)

stateful DHCPv6

*relay proxy agent configuration, [928-929](#)*

*relay proxy agent verification, [929](#)*

*server configuration, [931-932](#)*

stateless DHCPv6 configuration, [925-926](#)

static routes, configuring, [96](#)

static VRF routes, [121-122](#)

summary, [7](#)

traces, [8](#)

VRFs, creating, [78-79](#)

## **IP addresses**

assigning to interfaces

*CIDR notation, [60](#)*

*full subnet masks, [60](#)*

*secondary addresses, [60-61](#)*

*SLAAC, [895-924](#)*

binary notation

*base 2 calculations, [31-32](#)*

*bit values, [31-32](#)*

*to decimal conversion, [33](#)*

*decimal to binary conversions, [32](#)*

*defined, [31](#)*

classes

*A, [34](#)*

*B, [35](#)*

*C, [35](#)*

*D, [35](#)*

*history, [34](#)*

*listing of, [34](#)*

defined, [30](#)

dot-decimal notation, [30](#)

explicit, [133](#)

gateway, [30](#)

global routing table configuration, [80](#)

history, [34](#)

interfaces, [60](#)

interesting octets, [43](#)

NAT, [58](#)

network/host identifiers, [30](#)

next-hop self, [182-183](#)

overlapping, [76-77](#)

private

*defined, 58*

*IPv4 address ranges, 58*

*NAT, 58*

secondary, assigning to interfaces, [60-61](#)

special-purpose reserved, 59

status verification, displaying, [82-83](#)

subnet fields, 39

subnet masks. *See* [subnet masks](#)

usable, calculating, [37-38](#)

**ip address ip-address subnet-mask command, [60](#)**

**ip ospf process-id area area-id command, [204](#)**

**IP packets, delivering, [30](#)**

**IPv4**

BGP neighbor output, [421-424](#)

interfaces

*specific, displaying, [61](#)*

*summaries, displaying, [62](#)*

IPv6

*address resolution, compared, [934](#)*

*over IPv4 BGP sessions, [998-1001](#)*

multicast routing. *See* [multicast](#)

multicast versus IPv6 multicast, [1008](#)

session summary of loopback interfaces, [437](#)

**ipv4 address ip-address subnet mask command, [60](#)**

**ipv4 address ipv4-address prefix-length command, [60](#)**

## **IPv6**

address allocation hierarchy, [900-901](#)

address structure

*abbreviations, [895-896](#)*

*bit length values, [894](#)*

*case, [894](#)*

*components, [894](#)*

*hexadecimal to binary conversion, [896-898](#)*

*hextets, [894](#)*

*network boundaries, [894](#)*

*unicast, [894](#)*

aggregation, [902-903](#)

anycast, [906-908](#)

*duplicate address detection, disabling, [908](#)*

*topology, [906](#)*

*updated topology, [907](#)*

CEF, enabling, [899](#)

EIGRPv6, [944](#)

*classic mode configuration, [945](#)*

*configuration command reference chart, [954](#)*

*default routes, [952-953](#)*

*inter-router communication, [944](#)*

*IOS XR configuration, [946-947](#)*

*named mode configuration, [946](#)*

*route filtering, [953-954](#)*

*summarization, [950-952](#)*

*topology, [948](#)*

*verification, [947-950](#)*

*IS-IS, [977](#)*

*configuration, [979](#)*

*configuration commands reference chart, [988](#)*

*inter-router communication, [978](#)*

*IOS base configuration, [979-980](#)*

*IOS topology mode configuration, [981](#)*

*IOS XR base configuration, [980](#)*

*IOS XR topology mode configuration, [981-984](#)*

*TLVs, [944-978](#)*

*topology modes, [978-979](#)*

*verification, [985-987](#)*

*MBGP, [989](#)*

*BGP configuration commands reference chart, [1001](#)*

*inter-router communication, [989-990](#)*

*IOS configuration, [991-992](#)*

*IOS XR configuration, [992-993](#)*

*IPv6 over IPv4 BGP sessions, [998-1001](#)*

*verification. See [verification](#), [MBGP for IPv6](#)*

*multicast, [908-911](#)*

*active traffic flows, [1030](#)*

*address format, [909](#)*



*common addresses, [910](#)*

*configuration, [1025](#)*

*enabling, [1010](#)*

*enhancements, [1008](#)*

*groups, [910-911](#)*

*IPv4 multicast comparison, [1008](#)*

*MAC address mapping, [1009](#)*

*MLD, [1010-1015](#)*

*PIM, [1015](#)*

*PIM-SM. See [PIM-SM](#)*

*PIM-SSM, [1033-1034](#)*

*RPF, [1030-1032](#)*

*scope boundary, [1032-1033](#)*

*scope types, [909](#)*

*stream statistics, enabling, [1029](#)*

*topology, [908](#)*

*verification commands, [1024](#)*

*NDP, [912](#)*

*address resolution, [934-936](#)*

*default router preferences, modifying, [916](#)*

*disabling RA messages, [918](#)*

*duplicate address detection, [937-938](#)*

*ICMPv6 message types, [913](#)*

*NUD (neighbor unreachability detection), [936](#)*

*parameters, displaying, [916-917](#)*

*prefix valid lifetime/preferred lifetime values, [915-916](#)*

*RA lifetime values, [917-918](#)*

*RA messages, [913-914](#)*

*reachability, [936-937](#)*

*redirects, [919-920](#)*

*stateful DHCPv6. See [DHCPv6, stateful](#)*

*stateless address autoconfiguration, [895-924](#)*

*stateless DHCPv6, [924-926](#)*

*OSPFv3, [955](#)*

*authentication, [970-973](#)*

*configuration commands reference chart, [975](#)*

*flooding scopes, [959-960](#)*

*inter-router communication, [957](#)*

*IOS configuration, [960-961](#)*

*IOS XR configuration, [961](#)*

*LSAs, [958](#)*

*multiple instances, [973-975](#)*

*OSPFv2, compared, [955-957](#)*

*verification, [962-970](#)*

*overview, [893](#)*

*redistribution, [1002-1005](#)*

*IOS, [1002](#)*

*route tables, [1003-1005](#)*

*special purpose addresses, [911](#)*

*static routing, [941-942](#)*

*configuration commands reference chart, 943*

*configuring, 942-943*

*show commands reference chart, 943*

*subnetting chart, 903*

*unicast, 898-900*

*configuring, 899*

*global, 900-903*

*interface status, 899-900*

*link-local, 905-906*

*scopes, 898-899*

*unique local, 904-905*

**ipv6 authentication key-chain eigrp as-number key-chain-name command, 954**

**ipv6 authentication mode eigrp as-number md5 command, 954**

**ipv6 router eigrp as-number command, 954**

**ipv6 router isis command, 988**

**ipv6 summary-address eigrp as-number ipv6-prefix/prefix-length command, 954**

**ip vrf forwarding vrf-name command, 77**

**ip vrf vrf-name command, 77**

**IS (intermediate systems), 316**

**IS-IS (Intermediate System-to-Intermediate System), 68**

*AD, modifying, 531-532*

*areas, 318-319*

*authentication, 367*

*configuration, 369-370*

*hello, 368*

*LSP, 368*

*types, 367*

backbone continuity, [380-382](#)

broadcast interfaces as P2P, configuring, [353-355](#)

configuring

*interfaces, verifying, 343-346*

*IOS, 340*

*IOS XR, 340-341*

*neighbor adjacencies, verifying, 346-347*

*route verification, 347-348*

*topology example, 341-343*

default routes, [400-401](#)

DISs, [348-353](#)

*configuration, 352*

*CSNPs, 350*

*elections, 351*

*logical drawing, 348*

*LSP advertisement, 349*

*placement, 352-353*

*pseudonode LSPs, 349-350*

*verification, 353*

failure detection, [366](#)

*hello multiplier, 367*

*hello timers, 366-367*

*holding timer, 367*

filtering, [546](#)

hierarchy, [317](#)

IIH padding, removing, [365](#)

Integrated, [323](#)

interarea topology, [374](#)

interfaces

*IIH frequency, [393](#)*

*metric, [387-394](#)*

*passive, [362-364](#)*

*specific levels, [385-386](#)*

*verifying, [343-346](#)*

inter-router communication, [323-325](#)

*IIH packets, [327-328](#)*

*IS PDU addressing, [326-327](#)*

*IS protocol headers, [325](#)*

*LSPs, [329-332](#)*

*TLVs, [326](#)*

IPv6, [977](#)

*configuration, [979](#)*

*configuration commands reference chart, [988](#)*

*inter-router communication, [978](#)*

*IOS base configuration, [979-980](#)*

*IOS topology mode configuration, [981](#)*

*IOS XR base configuration, [980](#)*

*IOS XR topology mode configuration, [981-984](#)*

*TLVs, 944-978*

*topology modes, 978-979*

*verification, 985-987*

*loop prevention, 382-384*

*LSPDB, 317, 355*

*building topology, 359-360*

*displaying, 356-357*

*displaying topology, 360-361*

*non-pseudonode LSPs, 357-358*

*pseudonode LSPs, 358-359*

*SPF calculations, 361-362*

*LSPs, 329-332*

*attribute fields, 331*

*IDs, 330*

*lifetime, 329*

*sequence number, 331*

*TLVs, 332*

*types, 329*

*metrics*

*configuration, 391*

*mismatch, 389*

*narrow and wide, 388*

*topology, 391*

*verification, 392*

*neighbors, 333-339*

*adjacency capability chart, [386](#)*

*broadcast, [333-338](#)*

*P2P, [338-339](#)*

*verifying, [346-347](#)*

NET addresses, [322-323](#)

*common private, [322](#)*

*expanded structure, [322](#)*

*guidelines, [323](#)*

*minimal format, [322](#)*

*multiple, [323](#)*

NSAP, [320](#)

*DSP, [321](#)*

*IDP, [320](#)*

*structure, [320](#)*

OSPF commonalities, [316](#)

overload bits, [394-396](#)

*configuration, [395](#)*

*routing table, [395-396](#)*

*topology, [394](#)*

packets, [324](#)

path selection, [386](#)

*ECMP, [387](#)*

*interface metrics, [387-394](#)*

*processing order, [387](#)*

prefix

*suppression, [401-405](#)*

*redistribution, [576-578](#)*

*route-leaking, [377-380](#)*

*configuring, [377](#)*

*interarea TLVs, [379-380](#)*

*routing table, [378-379](#)*

*router specific levels, [384-385](#)*

*segmenting domains into multiple levels, [373-376](#)*

*SPT, [317](#)*

*suboptimal routing, [377](#)*

*summarization, [396](#)*

*configuration, [399](#)*

*discard routes, [399](#)*

*metric, [396-397](#)*

*ranges, [397](#)*

*routing table, [398](#)*

*topology, [398](#)*

***isis ipv6 metric command, [988](#)***

***IS-IS to IS-IS redistribution, [578-580](#)***

***is-local option (AS\_Path), [485](#)***

***iSPF (incremental SPF)***

***ISP to CE router connectivity, [100](#)***

***J - K***

***Join/prune PIM control message, [764](#)***

***keepalive messages, [413, 819-821](#)***



## **kernels**

IOS, [2](#)

IOS XE, [4](#)

IOS XR, [5](#)

**key chains, [174-177](#)**

## **keywords**

and, [504](#)

as-set, [639-641](#)

continue, [493-494](#)

filter-autorp, [864](#)

in, [864](#)

include-connected, [1004](#)

ip multicast boundary access-list command, [864](#)

not, [504](#)

or, [505](#)

out, [864](#)

rp-list, [867](#)

## **K values (EIGRP metrics)**

custom, [148-149](#)

default, [145](#)

## **L**

**label parameter, [23](#)**

**LAG (link aggregation) bundles, [722](#)**

**last-hop router (LHR), [763](#)**

**last usable host addresses, calculating, [43](#)**

**Layer 2 addressing, [83-84](#)**

**Layer 2 multicast addresses, [752-753](#)**

## **leaking**

configuring, [377](#)

interarea TLVs, [379-380](#)

routing table, [378-379](#)

suppressed routes, [634-637](#)

**length option (AS\_Path), [485-486](#)**

**LFA FRR (loop-free alternate FRR)**

**LHR (last-hop router), [763](#)**

**licenses, [3-4](#)**

**lifetime field (LSPs), [329](#)**

## **links**

aggregation (LAG) bundles, [722](#)

failures (EIGRP), [156-157](#), [163](#)

link-state

*acknowledgment packets, [194](#), [957](#)*

*advertisements. See [LSAs](#)*

*algorithms, [70-71](#)*

*database. See [LSDB](#)*

*packet database. See [LSPDB](#)*

*packets. See [LSPs](#)*

*request (LSR) packets, [194](#), [200](#)*

*update (LSU) packets, [194](#), [200](#)*

local addresses. See [LLAs](#)

OSPF virtual, [298-301](#)

*configuring*, [300](#)

*verification*, [300-301](#)

## **Linux differentials, [13](#)**

## **LLAs (link-local addresses), [905-906](#)**

assignment, [906](#)

format, [905](#)

manually assigning, [906](#)

## **load balancing**

EIGRP, [151-153](#)

RP groups, [836-840](#)

*configuration*, [837](#)

*group-to-RP mapping*, [837](#), [839-840](#)

*redundancy Auto-RP group-to-RP mappings*, [838](#)

*redundancy configuration*, [838](#)

## **loading configuration files, [24](#)**

## **Loading state (OSPF neighbors), [196](#)**

## **local**

ASNs, [660-664](#)

Network Control Block, [750](#)

OSPF filtering, [541-543](#)

PBR, [525-526](#)

preferences (BGP), [679-682](#)

*configuration*, [680](#)

*processing logic*, [682](#)

*routing table, [682](#)*

*topology, [679](#)*

routes, [93](#)

**LocPrf field (BGP tables), [429](#)**

**looking glass servers, [483](#)**

**loopback addresses (iBGP peering), [433-437](#)**

BGP table, [437](#)

configuration source, [435-436](#)

IPv4 session summary, [437](#)

OSPF, [229-230](#)

## **loops**

packet traces demonstrating, [118-119](#)

prevention, [119-120](#)

*BGP, [409-410](#)*

*IS-IS, [382-384](#)*

*route reflectors, [451-452](#)*

redistribution

*AD, [601-603](#)*

*overview, [590-592](#)*

*prefix filtering, [593-595](#)*

*seed metrics, increasing, [598-600](#)*

*summarization, [603-605](#)*

*tagging, [595-598](#)*

topology, [117-118](#)

**LSAs (link-state advertisements), [70, 191](#)**

DR/BDR distribution, [213](#)

exponential LSA sessions per routers on same segment, [212](#)

OSPF, [249-251](#)

*age*, [251](#)

*flooding*, [251](#)

*reduction through area segmentation*, [274](#)

*summary*, [270](#)

*Type 1*, [252-257](#)

*Type 2*, [257-259](#)

*Type 3*, [259-262](#)

*Type 4*, [265-268](#)

*Type 5*, [263-265](#)

*Type 7*, [268-269](#)

OSPFv2 versus OSPFv3, [955](#)

OSPFv3, [958](#)

*flooding scopes*, [959-960](#)

*types*, [959](#)

**LSDB (link-state database)**, [70](#), [192](#)

OSPF Type 1 LSA fields, [252](#)

OSPFv3, [965-969](#)

**LSPDB (link-state packet database) (IS-IS)**, [317](#), [355](#)

displaying, [356-357](#)

non-pseudonode LSPs, [357-358](#)

pseudonode LSPs, [358-359](#)

SPF calculations, [361-362](#)

topology

*building, 359-360*

*displaying, 360-361*

without transit networks, 404-405

## **LSPs (link-state packets), 70**

DISs

*advertisement, 349*

*pseudonode, 349-350*

IS-IS, 324, 329-332

*attribute fields, 331*

*authentication, 368*

*building topology, 359-360*

*displaying topology, 360-361*

*IDs, 330*

*lifetime, 329*

*LSPDB, displaying, 356-357*

*non-pseudonode, 357-358*

*pseudonode, 358-359*

*sequence number, 331*

*TLVs, 332*

*types, 329*

## **LSRs (link-state request) packets, 194, 200**

## **LSU (link-state update) packets, 194, 200**

**M**

**MAC addresses**

defined, [752](#)

mapping, [1009](#)

multicast, [752-753](#)

### **magic number method, [42-45](#)**

address range calculation examples

*10.55.200.33/12, [45](#)*

*172.16.2.3/23, [45](#)*

*192.168.100.5/26, [45](#)*

broadcast addresses, calculating, [43](#)

first usable addresses, [43](#)

interesting IP address octets, [43](#)

interesting subnet octets, [42-43](#)

last usable host address, [43](#)

process overview, [44](#)

### **management plane, [89](#)**

#### **mapping (SSM), [857](#)**

DNS, [857-860](#)

*configuration, [859](#)*

*topology, [858](#)*

*verification, [859](#)*

static, [860-862](#)

*configuring, [860](#)*

*verification, [861](#)*

#### **MAAs (RP mapping agents), [773](#)**

#### **match as-path acl-number command, [490](#)**

**matching length parameters (prefix matching), [471](#)**

**match ip address command, [490](#)**

**match ip address prefix-list prefix-list-name command, [490](#)**

**match local-preference command, [490](#)**

**match metric command, [490](#)**

**match statements, [497](#)**

**match tag tag-value command, [490](#)**

**maximum-paths command, [188](#)**

**MBGP (Multiprotocol BGP), [812](#)**

*AFIs/SAFIs, [812](#)*

*BGP*

*configuration commands reference chart, [1001](#)*

*topology, [813-814](#)*

*configuration, [815](#)*

*IOS, [991-992](#)*

*IOS XR, [992-993](#)*

*inter-router communication, [989-990](#)*

*IPv6 over IPv4 BGP sessions, [998-1001](#)*

*multicast BGP session summary verification, [817](#)*

*multicast traffic engineering, [875-882](#)*

*as chosen RPF source of information, [880](#)*

*configuring, [876](#)*

*incongruent unicast/multicast data paths, [878-880](#)*

*MBGP table, [877](#)*

*multicast BGP as chosen RPF source of information, [880](#)*



*OSPF preferred over BGP for RPF calculation, [878](#)*

*RPF information, [877](#)*

*traceroute unicast traffic, [881](#)*

*unicast BGP as RPF source after multicast link failures, [882](#)*

*unicast BGP session summary verification, [816](#)*

*update message packet capture, [813](#)*

*verification*

*commands, [993](#)*

*IPv6 route table, [997](#)*

*IPv6 unicast BGP table, [996](#)*

*neighbor status, [996](#)*

*next-hop address selection, [997](#)*

*router configuration, [994-995](#)*

**MDT, [851](#)**

**MED (Multi-Exit Discriminator), [693](#)**

*AIGP metric conversion, [693-694](#)*

*BGP best path selection, [704-714](#)*

*always compare MED feature, [711-713](#)*

*configuration, [706](#)*

*deterministic, [713-714](#)*

*missing MED behaviors, [710-711](#)*

*outbound traffic, influencing, [705](#)*

*processing logic, [708](#)*

*routing table after modification, [707](#)*

**memory, managing**

IOS, [2](#)

IOS XE, [4](#)

IOS XR, [6](#)

**merging configurations, [16](#)**

### **messages**

Auto-RP Cisco-RP-announce filtering, [867](#)

BGP, [411](#)

*keepalive*, [413](#)

*notification*, [414](#)

*open*, [412-413](#)

*update*, [413-414](#)

IGMPv2 messages, [757](#)

keepalive, [819-821](#)

MLD, [1010](#)

PIM control, [764](#)

PIM-SM IPv6 multicast, [1015](#)

RA, [913-914](#)

SA, [818-819](#), [828](#)

**metric command, [988](#)**

**Metric field (BGP tables), [429](#)**

### **metrics**

AIGP

*configuration*, [689](#)

*guidelines*, [688](#)

*MED conversion*, [693-694](#)

*modifications, [688](#)*

*neighbor sessions metrics verification, [690](#)*

*processing logic, [692-719](#)*

## EIGRP

*attribute propagation, [146](#)*

*custom K values, [148-149](#)*

*default K values, [145](#)*

*with definitions, [145](#)*

*formula, [145](#)*

*interface delay settings, [149-151](#)*

*interface metrics, [146](#)*

*load balancing, [151-153](#)*

*lowest link speed/cumulative delay, [146](#)*

*specific prefix, [147-148](#)*

*summarization, [171-172](#)*

*variance multiplier, [151-152](#)*

*wide, [153-154](#)*

external routes

*Type 1, [273](#)*

*Type 2, [273-274](#)*

interarea summarization, [277](#)

## IS-IS

*configuration, [391](#)*

*interface, [387-394](#)*

*mismatch, [389](#)*

*narrow and wide, 388*

*SPF, 361-362*

*summarization, 396-397*

*topology, 391*

*verification, 392*

lowest IGP (BGP path selection), 718-720

OSPF interface costs, 235-236

redistribution, 558

routing tables, 75-76

seed redistribution, 598-600, 606

Type 3 LSAs, 261

Type 4 LSAs, 265

**metric-style command, 988**

**metric weights TOS K1 K2 K3 K4 K5 command, 186**

**MFIB (multicast forwarding information base), 764**

**MLD (Multicast Listener Discovery), 1010-1015**

destination addresses, 1012

disabling, 1014

exclude filter mode, 1011

explicit tracking, 1014

groups, 1013

include filter mode, 1011

message types, 1010

querier election, 1012

query messages, 1012

static joins, [1015](#)

versions, [1010](#)

## **modes**

IOS

*global configuration, [10-11](#)*

*privileged, [10](#)*

*user, [9-10](#)*

IOS XR

*admin, [14](#)*

*EXEC, [14](#)*

*global configuration, [14](#)*

**MPLS (Multiprotocol Label Switching), [76](#)**

**MRIB (multicast routing information base), [763](#)**

**mroutes, [872-875](#)**

**MSDP (Multicast Source Discovery Protocol), [817-818](#)**

keepalive messages, [819-821](#)

peers, [822-828](#)

*IOS, [822](#)*

*IOS XR, [822-823](#)*

*MSDP and MBGP sessions between PIM domains, [823-827](#)*

*MSDP Compliance, [823](#)*

*SA message packet captures, [828](#)*

*SA RPF checks, [823](#)*

*status, [829](#)*

SA messages, [818-819](#)

stub networks, [831-833](#)

verification, [828-831](#)

*Mroute table*, [829](#)

*peer status*, [829](#)

*RPF check for source subnet*, [831](#)

*RPF check for source subnet without MBGP*, [831](#)

*SA cache*, [830](#)

## **mtrace, multicast troubleshooting, [887-889](#)**

with active source, [888](#)

no active source, [888](#)

PIM disabled, [889](#)

reference topology, [887](#)

## **multi-area adjacencies, [304-308](#)**

configuration, [305-306](#)

inefficient topologies, [304](#)

nonbroadcast interface verification, [304](#)

OSPF neighborship, [306-307](#)

verification, [306](#), [307-308](#)

## **multicast, [745](#)**

addressing

*Administratively Scoped Block*, [751](#)

*GLOP Block*, [751](#)

*IANA assigned*, [749](#)

*Internet Control Block*, [750](#)

*Layer 2*, [752-753](#)

*Local Network Control Block, 750*

*OSPF, 193*

*reserved addresses, 750*

*Source-Specific Multicast Block, 751*

*any source (ASM), 850*

*architecture, 745*

*configuration, 780-782, 786-787*

*distribution trees, 759*

*shared trees, 761*

*source trees, 759-760*

*forwarding information base (MFIB), 764*

*IGMP, 753*

*snooping, 753-756*

*v2, 757-758*

*v3, 759*

*versions, 753*

*IPv6, 908-911*

*active traffic flows, 1030*

*address format, 909*

*common addresses, 910*

*configuration, 1025*

*enabling, 1010*

*enhancements, 1008*

*groups, 910-911*

*IPv4 multicast comparison, 1008*

*MAC address mapping, [1009](#)*

*MLD, [1010-1015](#)*

*PIM, [1015](#)*

*PIM-SM. See [PIM-SM](#)*

*PIM-SSM, [1033-1034](#)*

*RPF, [1030-1032](#)*

*scope boundary, [1032-1033](#)*

*scope types, [909](#)*

*show commands, [1024](#)*

*stream statistics, enabling, [1029](#)*

*topology, [908](#)*

*MBGP, [812](#)*

*AFIs/SAFIs, [812](#)*

*BGP topology, [813-814](#)*

*configuration, [815](#)*

*multicast BGP session summary verification, [817](#)*

*unicast BPG session summary verification, [816](#)*

*update message packet capture, [813](#)*

*MSDP, [817-818](#)*

*keepalive messages, [819-821](#)*

*peers, [822-828](#)*

*SA messages, [818-819](#)*

*stub networks, [831-833](#)*

*verification, [828-831](#)*

*PIM, [762](#)*



*bidirectional, [802-808](#)*

*control messages, [764](#)*

*Dense Mode (PIM-DM), [765-767](#)*

*downstream, [763](#)*

*first-hop router (FHR), [763](#)*

*forwarder, [778-779](#)*

*incoming interface, [763](#)*

*last-hop router (LHR), [763](#)*

*multicast forwarding information base (MFIB), [764](#)*

*multicast routing information base (MRIB), [763](#)*

*multicast state, [764](#)*

*operating modes, [764](#)*

*outgoing interface (OIF), [763](#)*

*outgoing interface list (OIL), [763](#)*

*reference topology, [762](#)*

*RPF. See [RPF](#)*

*Sparse Mode, [768-772](#)*

*upstream, [763](#)*

*redundant RPs, [833-834](#)*

*anycast, [847-849](#)*

*Auto-RP load balancing, [836-840](#)*

*Auto-RP with multiple RPs, [835-840](#)*

*BSR, [840-846](#)*

*static RP, [846](#)*

*RPF, [776-777](#)*

*RPs, [772](#)*

*Auto-RP, [773-774](#)*

*configuring. See [RPs, configuring](#)*

*PIM bootstrap router, [775-776](#)*

*static, [773](#)*

*security, [862](#)*

*Auto-RP Cisco-RP-announce message filtering, [867](#)*

*Auto-RP TTL scoping, [862](#)*

*boundaries, [863-866](#)*

*PIM neighbor control, [869-870](#)*

*PIM register rate limit, [870](#)*

*PIM-SM, [867-869](#)*

*SSM, [850](#)*

*configuration, [852-853](#)*

*group-to-PIM mode mapping, [853](#)*

*IOS, enabling, [851](#)*

*IOS XR, enabling, [851](#)*

*mapping, [857-862](#)*

*MDT creation, [851](#)*

*mroute table, [855-857](#)*

*RPF neighbors, identifying, [854](#)*

*topology, [854](#)*

*stream statistics, [801](#)*

*traffic engineering*

*MBGP, [875-882](#)*

*RPF rules, [871-872](#)*

*static IGMP joins, [882-886](#)*

*static mroutes, [872-875](#)*

*unicast incongruent, [871](#)*

*troubleshooting, [-889](#)*

*mtrace, [887-889](#)*

*show commands, [886](#)*

verification

*active traffic flows, [801](#)*

*configuration, [788-789](#)*

*DR election, [792, 795](#)*

*IGMP, [790-791](#)*

*IOS show ip mroute flags, [794](#)*

*PIM group mapping, [792](#)*

*PIM interfaces, [791](#)*

*PIM topology, [793](#)*

*show mrib route/show mfib route flags, [798](#)*

*show pim topology flags, [799](#)*

*source registration between RP and FHR tree creation, [796-798](#)*

*SPT between RP and FHR tree creation, [796-798](#)*

*SPT path, [800](#)*

*SPT switchover, [799-800](#)*

*stream statistics, [801](#)*

*video feed, [748](#)*

**Multicast Listener Discovery. See [MLD](#)**

**multicast routing information base (MRIB), [763](#)**

**Multicast Source Discovery Protocol. *See* [MSDP](#)**

**Multi-Exit Discriminator. *See* [MED](#)**

## **multihop**

static routes

*multiple recursive lookups, [109-111](#)*

*overview, [108](#)*

*single recursive lookups, [108-109](#)*

**multiple instances (OSPFv3), [973-975](#)**

**multiple match route maps, [491-492](#)**

**multiple recursive static routes, [109-111](#)**

beneficial, [111](#)

configuring, [110](#)

recursive static routes routing table, [110-111](#)

routing table, [110](#)

## **multiprotocol**

BGP. *See* [MBGP](#)

Label Switching (MPLS), [76](#)

VRFs

*IOS, [78](#)*

*IOS XR, [78-79](#)*

*defined, [77](#)*

*IPv4-only migrations, [77](#)*

**multitopology command, [988](#)**

## **Part VII: High Availability**

## Chapter 21. High Availability

This chapter covers the following topics:

- Continuous forwarding
- Failure avoidance
- Event-driven failure detection
- Fast routing convergence
- Fast reroute

In networking, *availability* refers to the operational uptime of the network. The aim of *high availability* is to achieve continuous network uptime by designing a network to avoid single points of failure, incorporate deterministic network patterns, and utilize event-driven failure detection to provide fast network convergence. This chapter outlines various techniques for improving availability by reducing packet loss and accelerating network convergence around failures.

### Note

The features discussed in this chapter are not universally available for all Cisco router platforms. Use the software feature navigator tool on Cisco's website <http://www.cisco.com> to verify your minimum hardware and software requirements.

For demonstration purposes, the configuration examples in the text use the IPv4 address family, yet in most cases the features also apply to IPv6.

## NETWORK CONVERGENCE OVERVIEW

Network convergence is the time required to redirect traffic around a failure that causes loss of connectivity (LoC). The latency requirements for convergence vary by application. For example, an Open Shortest Path First (OSPF) network using default settings may take 5 or more seconds to converge around a link failure. This length of time may be acceptable for users reading Internet website articles, but it is completely unacceptable for IP telephony users.

A number of factors influence network convergence speed. In general, the higher the number of network prefixes and routers in the network, the slower the convergence will be. The primary factors that influence network convergence are as follows:

- **T1:** Time to **detect** the failure event
- **T2:** Time to **propagate** the event to neighbors
- **T3:** Time to **process** the event and calculate new best path
- **T4:** Time to **update** the routing table and program forwarding tables

Figure 21-1 illustrates the various time intervals that influence network convergence.

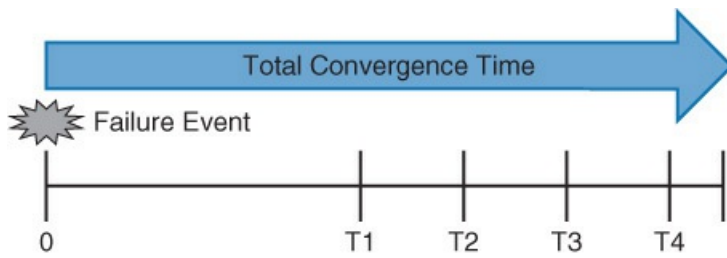


Figure 21-1 Convergence Time

## CONTINUOUS FORWARDING

Routers specifically designed for high availability include hardware redundancy, such as dual power supplies and route processors (RPs). An RP, which is also called a *supervisor* on some platforms, is responsible for learning the network topology and building the route table (Routing Information Base [RIB]). An RP failure can trigger routing protocol adjacencies to reset, resulting in packet loss and network instability. During an RP failure, it may be more desirable to hide the failure and allow the router to continue forwarding packets using the previously programmed Cisco Express Forwarding (CEF) table entries versus temporarily dropping packets while waiting for the secondary RP to reestablish the routing protocol adjacencies and rebuild the forwarding table.

The following two high availability features allow the network to route through a failure during an RP switchover:

- Stateful switchover (SSO) with nonstop forwarding (NSF)
- Stateful switchover (SSO) with nonstop routing (NSR)

### Stateful Switchover

Stateful switchover (SSO) is a redundancy feature that allows a Cisco router with two RPs to synchronize router configuration and control plane state information. The process of mirroring information between RPs is referred to as *checkpointing*. SSO-enabled routers always checkpoint line card operation and Layer 2 protocol states. During a switchover, the standby RP immediately takes control and will prevent problems such as interface link flaps and router reloads; however, Layer 3 packet forwarding is disrupted without additional configuration. The standby RP does not have any Layer 3 checkpoint information about the routing peer, so a switchover will trigger a routing protocol adjacency flap that clears the route table. After the route table is cleared, the CEF entries are purged, and traffic is no longer routed until the network topology is relearned and the forwarding table is reprogrammed. Enabling NSF or NSR high availability capabilities informs the routers to maintain the CEF entries for a short duration and continue forwarding packets through an RP failure until the control plane recovers.

SSO requires that both RPs have the same software version. [Example 21-1](#) demonstrates enabling stateful switchover using the redundancy mode configuration command **mode sso**. The feature is automatically enabled by default on all IOS XR routers.

### Example 21-1 Supervisor SSO Redundancy

```
IOS
redundancy
mode sso
```

The IOS and IOS XR command **show redundancy** provides details on the current SSO state operation.

[Example 21-2](#) displays the current redundancy status for an IOS router and an IOS XR router. The primary RP is highlighted in the example. In IOS terminology, the STANDBY HOT router is the backup RP, while in IOS XR the Standby is the backup RP.

### Example 21-2 SSO Redundancy Status

[Click here to view code image](#)

```
R1#show redundancy
! Output omitted for brevity
Current Processor Information :
-----
          Active Location = slot 5
          Current Software state = ACTIVE

Peer Processor Information :
-----
          Standby Location = slot 6
          Current Software state = STANDBY HOT
```

```
RP/0/RP1/CPU0:XR3#show redundancy summary
Active/Primary   Standby/Backup
-----
0/RP1/CPU0 (A)   0/RP0/CPU0 (S) (Node Ready, NSR: Ready)
0/RP1/CPU0 (P)   0/RP0/CPU0 (B) (Proc Group Ready, NSR: Ready)
```

Manually triggering a switchover between route processors is performed with the command **redundancy force-switchover** on IOS routers and with the command **redundancy switchover** on IOS XR nodes.

#### Nonstop Forwarding and Graceful Restart

Nonstop forwarding (NSF) is a feature deployed along with SSO to protect the Layer 3 forwarding plane during an RP switchover. With NSF enabled, the router continues to forward packets using the stored entries in the Forwarding Information Base (FIB) table.

There are three categories of NSF routers:

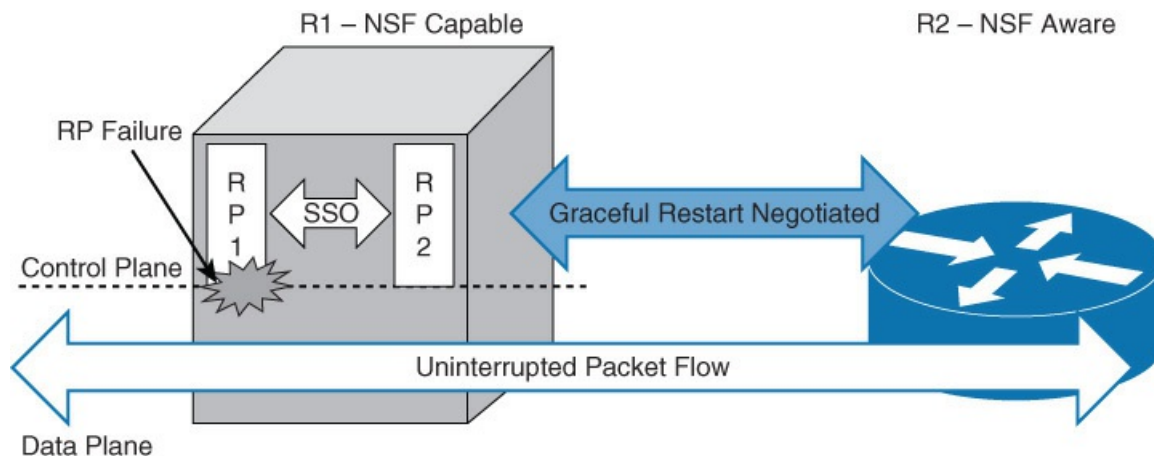
- **NSF-capable router:** A router that has dual RPs and is manually configured to use NSF to preserve the forwarding table through a switchover. The router restarts the routing process upon completion of the RP switchover.
- **NSF-aware router:** A neighbor router, which assists the NSF-capable router during the restart by preserving the routes and adjacency state during the RP switchover. An NSF-aware router does not require dual RPs.
- **NSF-unaware router:** A router that is not aware or capable of assisting a neighboring router during an RP switchover.

SSO with NSF is a high availability feature that is part of the internal router operation. Graceful restart (GR) is a subcomponent of NSF and is the mechanism the routing protocols use to signal



NSF capabilities and awareness.

In [Figure 21-2](#), R1 has NSF with SSO enabled. The primary RP has failed, but the router continues to forward packets using the existing CEF tables (FIB). During this time, the backup RP transparently takes over and reestablishes communication with R2 to restore the control plane and repopulate the routing tables (RIB). Throughout this *grace period*, R2 does not notify the rest of the network that a failure has occurred on R1, which maintains stability in the network and prevents a networkwide topology change event.



**Figure 21-2** SSO with NSF

[Table 21-1](#) outlines the expected routing protocol interaction between R1 and R2 when an RP switchover occurs on R1.

<b>SSO with NSF (R1 --&gt; R2 Graceful Restart Signaling)</b>				
	<b>Neighbor Adjacency Reset</b>		<b>RIB Recalculated Route Age Refreshed</b>	
	<b>R1</b>	<b>R2</b>	<b>R1</b>	<b>R2</b>
OSPF	Yes	No	Yes	No
IS-IS	Yes (IETF)	No	Yes	No
EIGRP	Yes	No	Yes	No
BGP	Yes	Yes	Yes	No

**Table 21-1** NSF Routing Protocol Interaction

The GR signaling mechanism differs slightly for each protocol, but the general concept is the same for all. Multiple RFC standards exist describing in detail how the GR should be communicated by each routing protocol. You can find a list of the relevant RFCs in the reference section at the end of the chapter.

[Figure 21-3](#) illustrates the NSF capability exchange:

- R1 signals to R2 that it is NSF/SSO-capable while forming the initial routing adjacency. The two agree that if R1 should signal a control plane reset, R2 will not drop the peering and will continue

sending and receiving traffic to R1 as long as the routing protocol hold timers do not expire.

- R1 sends a GR message to R2 indicating that the control plane is temporarily going offline immediately preceding the RP failover.
- R1 maintains the CEF table programming to forward traffic to R2 while the RP switchover takes place.
- Upon completion of the switchover, the new primary RP on R1 reestablishes communication with R2 and requests updates for repopulating the route table.

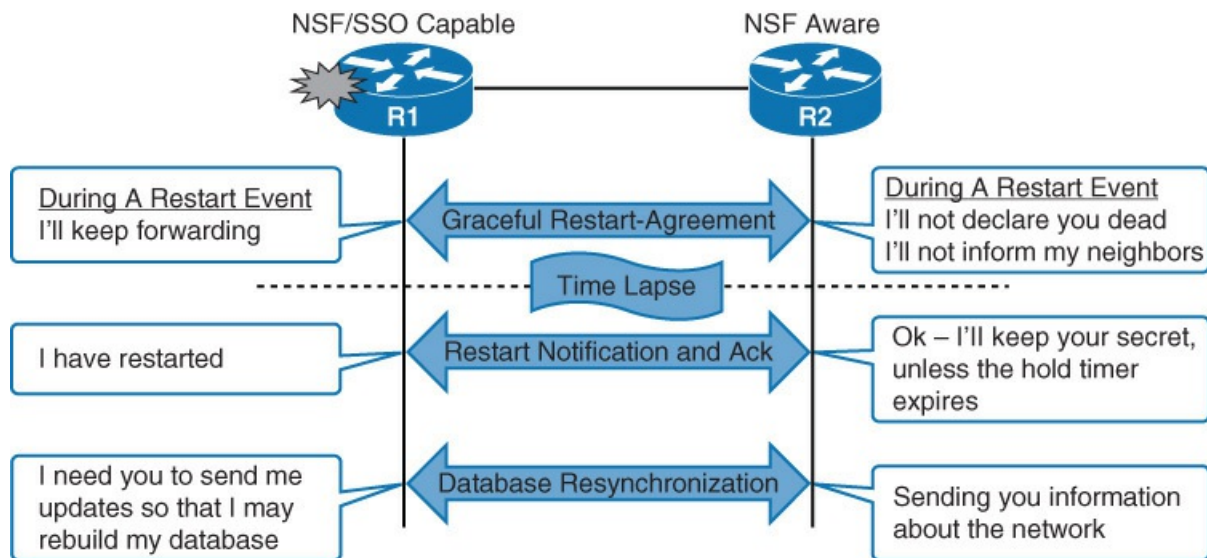


Figure 21-3 NSF Graceful Restart Relationship Building Process

R2 provides the route information to R1, while at the same time suppressing a notification to the rest of the network that an adjacency flap has occurred to R1. Stability is improved by preventing an unnecessary networkwide best path recalculation for the route flap.

Note

NSF freezes the CEF table and allows the router to forward packets to the last known good next-hop from prior to the RP switchover. If the network topology changes while the router is recovering, the packets may be suboptimally routed or possibly sent to the wrong destination and dropped.

NSF should not be deployed in parallel with routing protocol keepalive and holddown timers of less than 4 seconds. The NSF-capable router requires time to reestablish control plane communication during an RP switchover, and aggressive holddown timers can expire before this activity completes, leading to a neighbor adjacency flap. Bidirectional forwarding detection (BFD) is a better solution in most scenarios than aggressive keepalive timers. BFD is discussed in detail later in this chapter.

The interior routing protocols Open Shortest Path First (OSPF) Protocol, Intermediate System-to-Intermediate System (IS-IS) Protocol, and Enhanced Interior Gateway Routing Protocol (EIGRP) are automatically NSF-aware for both IOS and IOS XR. Routers with dual RPs need to be configured as NSF-capable within the routing protocol.

The routers in Figure 21-4 are referenced throughout the next series of continuous forwarding examples. In each of the exercises, R1 is the NSF-capable router that is performing an RP stateful switchover. Where possible, verification captures from the NSF-aware routers R2 and XR3 demonstrate the GR signaling.

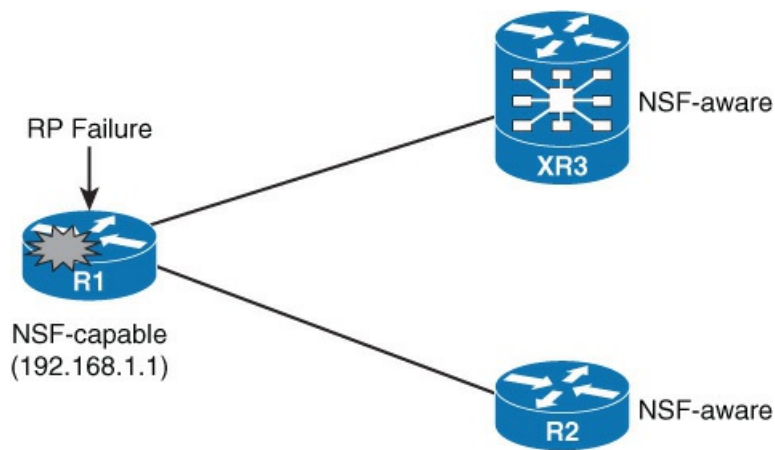


Figure 21-4 NSF Routers

## OSPF

Two GR configuration modes are available for OSPF:

- Cisco:** The Cisco mode for performing GR adds Link Local Signaling (LLS) bits to the hello and DBD packets. The LSDB Resynchronization (LR) bit is included in the database description (DBD) packets to indicate out-of-band resynchronization (OOB) capability. A hello packet with the LLS Restarting R bit set indicates that the router is about to perform a restart. This method was developed by Cisco and was later standardized by the IETF in RFC 4811, 4812, and 5613.
- IETF:** The IETF RFC 3623 method for performing GR uses link-local opaque link-state advertisements (LSAs). A router sends a Grace LSA to indicate it is about to restart the OSPF process. The router resynchronizes the LSDB using Grace LSAs once the restart completes.

The routing protocol command to enable GR and NSF capabilities is `nsf [cisco | ietf]`. [Example 21-3](#) demonstrates how to enable a router to be NSF-capable and to signal GR capabilities to its peers.

### Example 21-3 OSPF NSF Configuration

```
IOS
router ospf 100
nsf cisco
```

```
IOS XR
router ospf 100
nsf cisco
```

You can view the NSF status with the IOS command `show ip ospf neighbor detail` or with the IOS XR command `show ospf neighbor detail`.

[Example 21-4](#) demonstrates that the neighbors are using LLS, which is required for NSF awareness and successful GR negotiations. Notice that the neighbor adjacency peering includes the LLS LSDB OOB Resynchronization (LR) capability bit set.

## Example 21-4 NSF Graceful Restart Agreement

[Click here to view code image](#)

```
R2#show ip ospf neighbor detail
! Output omitted for brevity
Neighbor 192.168.1.1, interface address 10.0.12.1
  In the area 0 via interface GigabitEthernet0/0
  Neighbor priority is 1, State is FULL, 6 state changes
  Options is 0x12 in Hello (E-bit, L-bit)
  Options is 0x52 in DBD (E-bit, L-bit, O-bit)
  LLS Options is 0x1 (LR)
  Dead timer due in 00:00:38
  Neighbor is up for 00:04:58
  Index 1/1, retransmission queue length 0, number of retransmission 10
```

```
RP/0/RSP0/CPU0:XR3#show ospf 100 neighbor detail
! Output omitted for brevity
Neighbors for OSPF 100

Neighbor 192.168.1.1, interface address 10.0.13.1
  In the area 0 via interface GigabitEthernet0/0/0/13
  Neighbor priority is 1, State is FULL, 6 state changes
  Options is 0x52
  LLS Options is 0x1 (LR)
  Dead timer due in 00:00:38
  Neighbor is up for 00:04:58
  Number of DBD retrans during last exchange 0
  Index 1/1, retransmission queue length 0, number of retransmission 16
  First 0(0)/0(0) Next 0(0)/0(0)
  LS Ack list: NSR-sync pending 0, high water mark 0
```

**Example 21-5** demonstrates that a GR has just taken place on the neighboring router R1. Notice that R2 and XR3 do not terminate the adjacency session because of the previously negotiated GR agreement. During a GR event, IOS routers display the OOB resynchronization countdown timer for the recovery. If R1 does not respond by the end of the timer, R2 and XR3 will consider the connection down. IOS XR routers also use the OOB-Resync timer to track the state of the neighbor adjacency but the timer status is not included in the **show** command output.

## Example 21-5 OSPF Graceful Restart Initiated

[Click here to view code image](#)

```
R2#show ip ospf neighbor detail
```

```
! Output omitted for brevity
```

```
Neighbor 192.168.1.1, interface address 10.0.12.1
  In the area 0 via interface GigabitEthernet0/0
  Neighbor priority is 1, State is FULL, 12 state changes
  Options is 0x52 in DBD (E-bit, L-bit, O-bit)
  LLS Options is 0x1 (LR)
  oob-resync timeout in 00:00:39
  Dead timer due in 00:00:39
  Neighbor is up for 00:08:00
  Index 1/1, retransmission queue length 0, number of retransmission 98
  First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
```

```
RP/0/RSP0/CPU0:XR3#show ospf neighbor detail
```

```
! Output omitted for brevity
```

```
Neighbors for OSPF 100
```

```
Neighbor 192.168.1.1, interface address 10.0.13.1
  In the area 0 via interface GigabitEthernet0/0/0/13
  Neighbor priority is 1, State is FULL, 6 state changes
  LLS Options is 0x1 (LR)
  Dead timer due in 00:00:39
  Neighbor is up for 00:08:00
  Number of DBD retrans during last exchange 0
  Index 1/1, retransmission queue length 0, number of retransmission 0
  First 0(0)/0(0) Next 0(0)/0(0)
  LS Ack list: NSR-sync pending 0, high water mark 0
```

**Example 21-6** demonstrates that a GR completed successfully 11 seconds ago. R1 recovered, and the neighbor adjacency was not reset on the R2 and XR3 side of the connection per the GR agreement.

### **Example 21-6** *OSPF Graceful Restart Completed*

**[Click here to view code image](#)**

```
R2#show ip ospf neighbor detail
```

```
! Output omitted for brevity
```

```
Neighbor 192.168.1.1, interface address 10.0.12.1
  In the area 0 via interface GigabitEthernet0/0
  Neighbor priority is 1, State is FULL, 16 state changes
  Options is 0x52 in DBD (E-bit, L-bit, O-bit)
  LLS Options is 0x1 (LR), last OOB-Resync 00:00:11 ago
  Dead timer due in 00:00:34
  Neighbor is up for 00:08:37
  Index 2/2, retransmission queue length 0, number of retransmission 98\
```

```
RP/0/RSP0/CPU0:XR3#show ospf neighbor detail
```

```
! Output omitted for brevity
```

```
Neighbors for OSPF 100
```

```
Neighbor 192.168.1.1, interface address 10.0.13.1
  In the area 0 via interface GigabitEthernet0/0/0/13
  Neighbor priority is 1, State is FULL, 10 state changes
  LLS Options is 0x1 (LR), last OOB-Resync 00:00:11 ago
  Dead timer due in 00:00:34
  Neighbor is up for 00:08:37
  Number of DBD retrans during last exchange 0
  Index 2/2, retransmission queue length 0, number of retransmission 0
  LS Ack list: NSR-sync pending 0, high water mark 0
```

#### Note

During the GR, the NSF-aware router does not clear the route table entries or the neighbor adjacency, and therefore the age of the routes predates the GR event as if nothing happened. The NSF-capable router performing the SSO restarts the routing process, so the neighbor adjacency and route table entries age is reset to zero on the restarting router.

## IS-IS

There are two GR configuration modes available for IS-IS.

- **Cisco:** The Cisco IS-IS NSF mode does not perform any GR signaling, instead control plane checkpoint state information is propagated between the active and standby RPs. During a switchover, the restarting router reestablishes the IS-IS adjacency with the neighbor using the control plane state information from prior to the RP switchover. The neighboring router does not need to be NSF-aware as the entire process completes successfully as long as the IS-IS keepalive timers do not expire.

- **IETF:** RFC 3847 describes the IETF GR method. IS-IS signals a GR using a new Restart Option TLV 211 extension in the hello packet. When the NSF-capable router restarts, it signals a hello packet with a Restart Request (RR) bit set to 1. The NSF-aware router responds back with a hello packet with the Restart Acknowledgment (RA) bit set to 1. The Restart Option TLV is removed from the hello packets once the LSDB is synchronized between the two peers.

Example 21-7 demonstrates how to configure the IETF GR method for IS-IS.

### Example 21-7 IS-IS NSF Configuration

#### IOS

```
router isis LAB
nsf ietf
```

#### IOS XR

```
router isis LAB
nsf ietf
```

The IOS command **show clns neighbor detail** and the IOS XR command **show isis neighbor detail** display whether the neighbor is NSF capable. [Example 21-8](#) demonstrates that the neighboring router is capable of supporting an IS-IS GR.

### Example 21-8 Verifying IS-IS NSF Capabilities

[Click here to view code image](#)

```
R2#show clns neighbors detail
System Id      Interface  SNPA                State Holdtime  Type Protocol
R1             Gi0/0     000c.860a.9000     Up    24         L2   IS-IS
  Area Address(es): 49.0001
  IP Address(es):  10.0.12.1*
  Uptime: 00:25:23
  NSF capable
  Interface name: GigabitEthernet0/0
```

```
RP/0/RSP0/CPU0:XR3#show isis neighbors detail

IS-IS LAB neighbors:
System Id      Interface  SNPA                State Holdtime  Type IETF-NSF
R1             Gi0/0/0/13 000c.860a.9000     Up    24         L2   Capable
  Area Address(es): 49.0001
  IPv4 Address(es): 10.0.13.1*
  Topologies: 'IPv4 Unicast'
  Uptime: 00:25:23
```

## EIGRP

Enhanced Interior Gateway Routing Protocol (EIGRP) includes a Restart (RS) bit that allows for the signaling of a GR. When the RS bit is set to 1, the neighboring NSF-aware router knows that an RP switchover is about to take place on the NSF-capable router. Once the SSO event completes, the two routers synchronize route tables with the RS bit still enabled. The NSF-aware router sends an End-of-Table (EOT) signal to indicate that it has provided all the updates, at which point the two routers clear the RS bit from the EIGRP packets.

[Example 21-9](#) demonstrates that the IOS command to enable GR on the NSF-capable router is **nsf**. On IOS XR routers, the NSF capability is enabled by default. The EIGRP mode command to disable NSF is **nsf disable**.

### Example 21-9 IS-IS NSF Configuration

[Click here to view code image](#)

```
IOS (Classic AS Configuration)
router eigrp 100
nsf
```

```
IOS (Named Mode Configuration)
router eigrp LAB
  address-family ipv4 unicast autonomous-system 100
  nsf
```

Example 21-10 displays the EIGRP neighbor status on an NSF-aware router. The highlighted output indicates the neighbor has been up for 50 minutes but that an RP switchover occurred on R1 52 seconds ago.

### Example 21-10 Viewing EIGRP Neighbor Status

[Click here to view code image](#)

```
R2#show ip eigrp neighbors detail
EIGRP-IPv4 Neighbors for AS(100)
H   Address                Interface      Hold Uptime   SRTT  RTO  Q  Seq
                               (sec)         (ms)  (ms)  Cnt  Num
0   10.0.12.1               Gi0/0         11 00:50:25   1    200  0  11
    Time since Restart 00:00:52
    Version 5.0/3.0, Retrans: 1, Retries: 0, Prefixes: 5
    Topology-ids from peer - 0
```

```
RP/0/RSP0/CPU0:XR3#show eigrp neighbors detail
IPv4-EIGRP neighbors for AS(100) vrf default
H   Address                Interface      Hold Uptime   SRTT  RTO  Q  Seq
                               (sec)         (ms)  (ms)  Cnt  Num
0   10.0.13.1               Gi0/0/0/13    10 00:50:25   4    200  0  10
    Restart time 00:00:52
    Version 5.0/3.0, Retrans: 0, Retries: 0, Prefixes: 5
```

## BGP

RFC 4724 describes BGP GR signaling. Enabling the features modifies the initial BGP open negotiation message to include GR capability code 64. The GR capability informs the neighboring router that it should not reset the BGP session and immediately purge the routes when it performs an SSO.

During an RP SSO event, the TCP connection used to form the BGP session is reset, but the routes in the RIB are not immediately purged. The BGP NSF-aware router detects that the BGP TCP socket has cleared, marks the old routes stale, and begins a countdown timer, while at the same time continuing to forward traffic using the route table information from prior to the reset. Once the NSF-capable router recovers, it forms a new TCP session and sends a new GR message notifying the NSF-capable router that it has restarted. The two routers exchange updates until the NSF-capable router signals the end-of-RIB (EOR) message. The NSF-aware router clears the stale countdown timer and any stale entries that are no longer present are removed.

#### Note

An RP failure will cause the BGP session to temporarily reset on both sides of the connection. The session uptime will correlate to the RP failure. On the restarting router, the route table entries will be purged, but on the NSF-aware router, the learned routes will remain unchanged with an age that precedes the GR.

Unlike the other routing protocols, BGP routers are not NSF-aware or NSF-capable by default. The GR capability requires manual configuration on both sides of the session.

The steps for configuring BGP NSF are as follows:



### Step 1. Enable GR and NSF awareness.

GR support is enabled for all peers with the IOS command **bgp graceful-restart** or the IOS XR command **graceful-restart** in BGP configuration mode. The IOS command to selectively enable GR per peer is **neighbor ip-address ha-mode graceful-restart**. In IOS XR, GR can be enabled or disabled per peer with the command **graceful-restart [disable]**.

### Step 2. Set the restart time (optional).

The GR restart time determines how long the router will wait for the restarting router to send an open message before declaring the neighbor down and resetting the session. The value may be changed with the IOS command **bgp graceful-restart restart-time seconds** or with the IOS XR command **graceful-restart restart-time seconds**.

The default value is 120 seconds.

### Step 3. Set stale route timeouts (optional).

The stale routes timeout determines how long the router will wait for the end-of-record (EOR) message from the restarting neighbor before purging routes. The value may be changed with the IOS command **bgp graceful-restart stalepath-time seconds** or with the IOS XR command **graceful-restart stalepath-time seconds**. The default value is 360 seconds.

#### Note

Enabling GR capabilities on an IOS router after the session has already been established will trigger the session to renegotiate and will result in an immediate session flap. In IOS XR, enabling GR is nondisruptive. The session will continue working with the previous setting until the session is manually reset and the capability is negotiated.

Example 21-11 demonstrates how to configure GR. R1 is using BGP per neighbor GR, and XR3 has the feature enabled globally for all sessions.

### Example 21-11 BGP Graceful Restart

[Click here to view code image](#)

```
R1
router bgp 65001
  bgp graceful-restart restart-time 120
  bgp graceful-restart stalepath-time 360
  neighbor 10.0.13.3 ha-mode graceful-restart
```

```
XR3
router bgp 65003
  bgp graceful-restart restart-time 120
  bgp graceful-restart stalepath-time 360
  bgp graceful-restart
```

The command **show bgp ipv4 unicast neighbor** can be used to determine whether GR

capabilities have been negotiated. [Example 21-12](#) verifies that routers R1 and XR3 have successfully negotiated NSF capabilities with each other.

### Example 21-12 Verifying Negotiated GR and NSF Capabilities

[Click here to view code image](#)

```
R1#show bgp ipv4 unicast neighbors 10.0.13.3 | i Graceful
Graceful Restart Capability: advertised and received
Graceful-Restart is-enabled, restart-time 120 seconds, stalepath-time 360 seconds
RP/0/RSP0/CPU0:XR3#show bgp ipv4 unicast neighbors 10.0.13.1 | i Graceful
Graceful restart is-enabled
Graceful Restart (GR Awareness): received
Graceful Restart capability advertised and receive
```

#### Note

It is common to have BGP and IGP routing protocols on the same routers. To avoid suboptimal routing during an RP failover, the protocols should have matching NSF capabilities configured.

### Nonstop Routing

Nonstop routing (NSR) is an internal Cisco router feature that does not use a GR mechanism to signal to neighboring routers that an RP switchover has taken place. Instead, the primary RP is responsible for constantly transferring all relevant routing control plane information to the backup RP, including routing adjacency and TCP sockets. During a failure, the new RP uses the “checkpoint” state information to maintain the routing adjacencies and recalculate the route table without alerting the neighboring router that a switchover has occurred.

[Figure 21-5](#) demonstrates an RP switchover on R1, which has SSO with NSR enabled. The routing protocol peering between R1 and R2 is unaffected by the RP failure. R2 is unaware that a failure has occurred on R1. Throughout the entire RP failover process, the traffic continues to flow between the two routers unimpeded using the Cisco Express Forwarding (CEF) forwarding table.

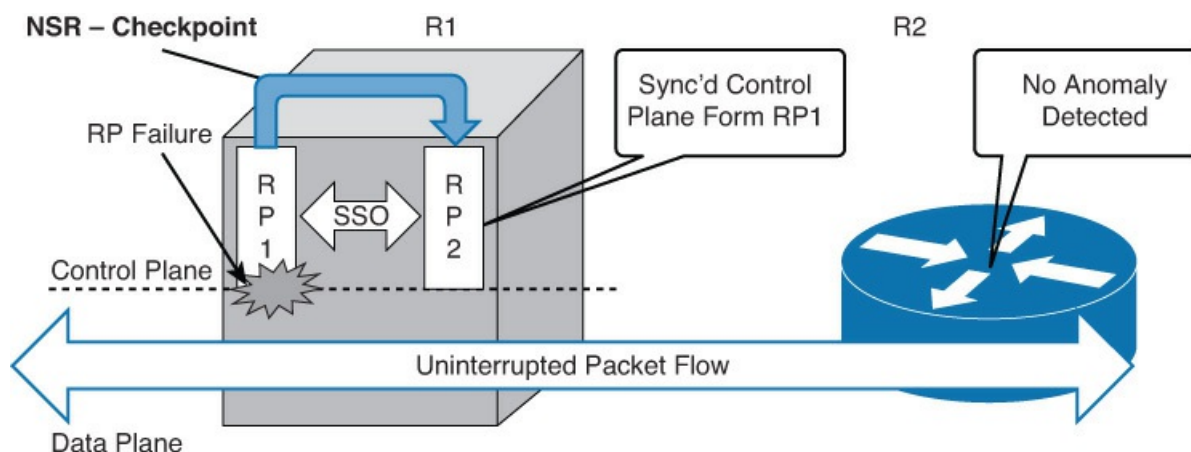


Figure 21-5 SSO with NSR

NSR's primary benefit over NSF is that it is a completely self-contained high availability solution. There is no disruption to the routing protocol interaction so the neighboring router does not need to be NSR- or NSF-aware. [Table 21-2](#) outlines the expected routing protocol interaction

between R1 and R2 when an RP switchover occurs on R1.

<b>SSO with NSR (R1 Internal RP1 --&gt; RP2 Control Plane Checkpoint)</b>				
	<b>Neighbor Adjacency Reset</b>		<b>RIB Recalculated Route Age Refreshed</b>	
	<b>R1</b>	<b>R2</b>	<b>R1</b>	<b>R2</b>
OSPF	No	No	Yes	No
IS-IS	No	No	Yes	No
EIGRP	NA			
BGP	No	No	Yes	No

**Table 21-2** NSR Routing Protocol Interaction

### NSR for OSPF

NSR for OSPF is enabled with the routing configuration mode command **nsr**. [Example 21-13](#) shows how to enable NSR for OSPF.

### Example 21-13 OSPF NSR Configuration

```
IOS
router ospf 100
nsr
```

```
IOS XR
router ospf 100
nsr
```

The IOS command to verify whether NSR is operational is **show ip ospf nsr**, and the IOS XR command is **show redundancy**. [Example 21-14](#) demonstrates that the router is configured for NSR and that the backup RP is available for a switchover if required.

### Example 21-14 OSPF NSR Verification

[Click here to view code image](#)

```

R1#show ip ospf nsr
Active RP
Operating in duplex mode
Redundancy state: ACTIVE
Peer redundancy state: STANDBY HOT
Checkpoint peer ready
Checkpoint messages-enabled
ISSU negotiation complete
ISSU versions compatible

Routing Process "ospf 100" with ID 192.168.1.1
NSR configured
Checkpoint message sequence number: 2917
Standby synchronization state: synchronized
! Output omitted for brevity

```

```

RP/0/RP0/CPU0:XR3#show redundancy summary
Active/Primary Standby/Backup
-----
0/RP0/CPU0 (A) 0/RP1/CPU0 (S) (Node Ready, NSR: Ready)
0/RP0/CPU0 (P) 0/RP1/CPU0 (B) (Proc Group Ready, NSR: Ready)

```

## NSR for IS-IS

The IS-IS NSF operating mode **nsf cisco** is essentially NSR because the router uses SSO to checkpoint IS-IS control plane data between the RPs. The router does not signal a GR to its neighbor in this mode. [Example 21-15](#) demonstrates how to enable NSF mode Cisco for IS-IS.

### Example 21-15 IS-IS NSF Mode Cisco Configuration

```

IOS
router isis LAB
nsf cisco

```

```

IOS XR
router isis LAB
nsf cisco

```

## NSR for BGP

IOS enables NSR for BGP on a per peer basis with the command **neighbor ip-address ha-mode sso**. The IOS XR BGP command **nsr** enables NSR for all neighbor sessions.

### Note

NSR and NSF/GR can be configured at the same time. Typically, NSR will take precedence over NSF GR. However, when deployed together with BGP on IOS, the router will attempt to use the NSF GR method over NSR. The IOS command **neighbor ip-address ha-mode graceful-restart disable** ensures that NSR is the active high availability feature used for the peering.

IOS XR routers give NSR preference over NSF GR when the two features are deployed in unison.

[Example 21-16](#) demonstrates how to configure NSR for BGP. GR has been globally enabled

within the BGP process. The GR capability has been disabled for the specific peer to ensure that SSO with NSR is employed.

### Example 21-16 BGP NSR

[Click here to view code image](#)

```
R1
router bgp 65001
  bgp graceful-restart restart-time 120
  bgp graceful-restart stalepath-time 360
  bgp graceful-restart
  neighbor 10.0.13.1 ha-mode sso
  neighbor 10.0.13.1 ha-mode graceful-restart disable
```

```
XR3
router bgp 65003
  nsr
```

The IOS command **show bgp ipv4 unicast sso summary** or the IOS XR command **show bgp ipv4 unicast sso summary** may be used to verify BGP NSR operational status, as demonstrated in [Example 21-17](#).

### Example 21-17 BGP NSR Verification

[Click here to view code image](#)

```
R1#show bgp ipv4 unicast sso summary
Load for five secs: 1%/0%; one minute: 1%; five minutes: 2%

Total sessions with stateful switchover support-enabled: 1
Total sessions configured with NSR mode and in established state: 1
! Output omitted for brevity
```

```
RP/0/RSP0/CPU0:XR3#show bgp ipv4 unicast sso summary
BGP router identifier 192.168.3.3, local AS number 65003
BGP generic scan interval 60 secs
Non-stop routing is-enabled
! Output omitted for brevity
Neighbor      Spk   AS   TblVer  SyncVer  AckVer  NBRState   NSRState
10.0.13.1     0 65001 321      0        321 Established None
Address Family IPv4 Unicast:
  Distance: external 20 internal 200 local 200
Routing Information Sources:
  Neighbor      State/Last update received  NSR-State  GR-Enabled
  10.0.13.1     00:00:05                    NSR Ready  Yes
```

## Nonstop Forwarding and Nonstop Routing Together

Routing protocols may use NSF and NSR together at the same time. NSR takes precedence for the IGP routing protocols, and NSF will be used as a fallback option where NSR recovery is not possible. For example, NSR does not support process restarts, and therefore there are benefits to deploying the two high availability features in tandem.

The IOS XR global command **nsr process-failures switchover** may be used when NSF is not enabled to force an RP failover when a routing process restarts.

## FAILURE AVOIDANCE

Routing protocols react to a link failure by recalculating a new best path to route around the problem. An unstable link can have a destabilizing effect on network convergence as the routers are constantly updating the routing table. To stabilize the overall topology, it may be desirable to temporarily remove the unstable prefix from the routing table as a method of failure avoidance until that section of the network stabilizes.

### Route Flap Dampening

IP event dampening and BGP dampening are two features that suppress the effects of an oscillating or “flapping” link or route from routing protocol use. The two features follow the same route flap dampening (RFD) model. IP event dampening applies a penalty for a link transition, up to down. BGP dampening applies the penalty for unstable BGP routes. Each flap event incurs a penalty of 1000 until the dampening suppress threshold is met, at which point the routing protocols suppress advertising routes associated with the dampening. To prevent the routes from permanently being suppressed, an exponential decay mechanism reduces the associated dampening penalty. A half-life period determines how quickly the dampening penalty is reduced. Once the dampening penalty is below the reuse threshold, the route is unsuppressed and advertised throughout the network again.

Table 21-3 outlines the parameters used by route flap dampening.

Parameter	Description
Half-life	The time it takes for a penalty to be decreased by half is the half-life period. The higher the half-life time interval, the longer dampening will penalize an unstable link. The exponential decay mechanism for reducing the penalty occurs every 5 seconds.
Suppress	A route is suppressed when its penalty exceeds the suppress value limit.
Reuse	The route is unsuppressed when the penalty for the flapping route falls below the reuse value threshold.
Restart penalty	The restart penalty is an optional IP event dampening parameter that assigns a penalty to an interface when it initializes for the first time after a router reload.
Max suppress time	This is the maximum amount of time a route prefix may be suppressed.
Max suppress penalty	The max suppress penalty is the largest penalty value a route may accrue. To ensure that a route is correctly suppressed, always use a suppress limit value that is less than the maximum penalty; otherwise, the route will not be susceptible to dampening. The value is not assigned in the command-line interface (CLI), but instead calculated by the router using the formula in Figure 21-6.

Table 21-3 RFD Parameters

- 1 Max-penalty = reuse-limit ×  $\frac{\text{max-suppress-time}}{\text{half-life}}$

---

- 2 Max-penalty = 1000 × 2  $\frac{(60/15)}$

---

- 3 Max-penalty = 16000

Figure 21-6 Maximum Dampening Formula

Figure 21-7 illustrates how IP event dampening applies a penalty to an interface that is unstable.

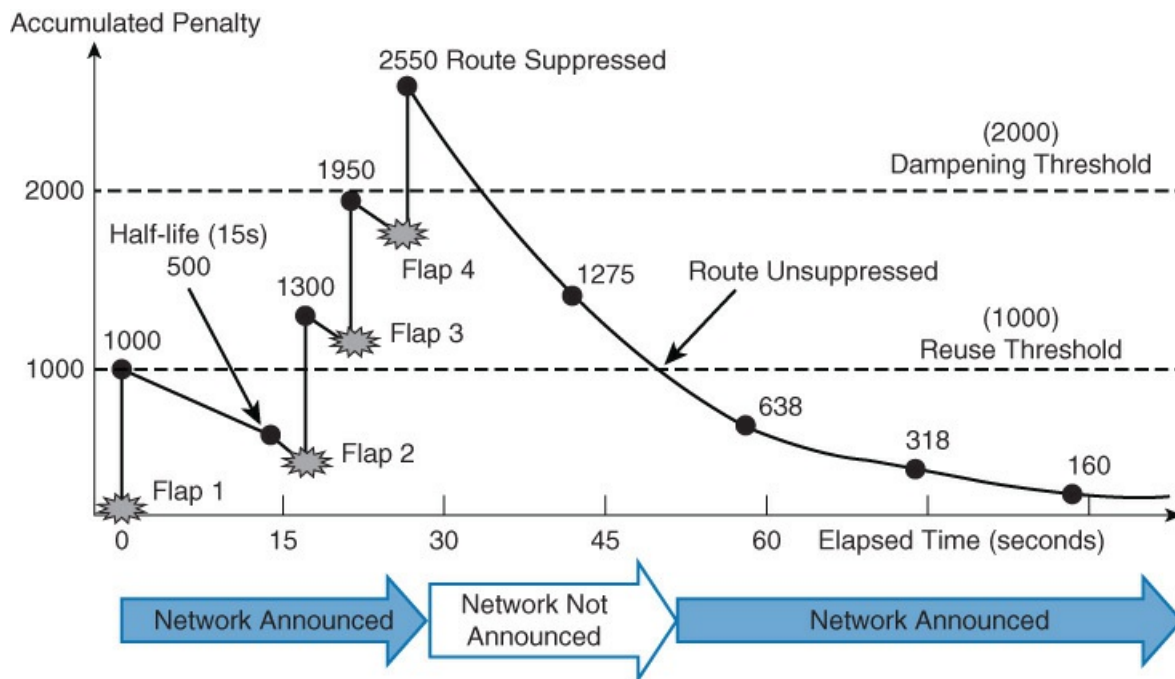


Figure 21-7 Dampening

The dampening in the illustration uses the following values:

- **Half-life:** 15 seconds
- **Reuse limit:** 1000
- **Suppress:** 2000
- **Max suppress time:** 60 seconds
- **Maximum penalty:** 16000

Initially, the interface has zero penalties accumulated. Link transitions increment a penalty of 1000. The dampening penalty is continuously decremented every 5 seconds using the exponential decay backoff algorithm. After 15 seconds (one half-life period), the value is reduced to 500. Over the next few seconds, the interface link-state transitions three more times. The fourth flap finally breaches the dampening suppression threshold of 2000, causing the routes associated with the interface to be suppressed. The routes are unsuppressed approximately 20 seconds later once the accumulated penalty value is reduced below the dampening reuse threshold.

## IP Event Dampening

The IP event dampening feature dampens all routing protocol activity on an unstable interface. The dampened interface suppresses all routing activity, including advertising routes for the connected interface and forming neighbor adjacencies. In many designs, especially networks with link redundancy, it may be desirable to temporarily remove the unstable connection from the routing path as a method of failure avoidance until the link stabilizes. The feature is enabled on a per interface basis with the IOS interface command **dampening** [*half-life-period reuse-threshold*] [*suppress-threshold max-suppress*] [*restart-penalty*] or with the IOS XR command **dampening** [*half-life-period*] [*reuse-threshold suppress-threshold max-suppress*] [*restart-penalty*].

Table 21-4 lists the default dampening values to be used when applying only the **dampening** command to the interface. Notice that the IP event dampening timer values are implemented in seconds for IOS and in minutes for IOS XR.

Parameter	IOS	IOS XR
Half-life	5 seconds	1 minute
Reuse	1000	750
Suppress	2000	2000
Max suppress time	Four times the half-life or 20 seconds	Four times the half-life
Restart penalty	0	0

Table 21-4 IP Event Dampening Default Timers

Example 21-18 demonstrates how to configure IP event dampening. The IOS router is using the values from Figure 21-7, and the IOS XR router is using the default values from Table 21-4.

### Example 21-18 IP Event Dampening

[Click here to view code image](#)

```
IOS
interface GigabitEthernet4/14
dampening 15
```

```
XR
interface GigabitEthernet0/0/0/13
dampening
```

The dampening status for an interface is viewed with the IOS command **show interface dampening** and the IOS XR command **show im dampening interface-type interface-number**.

Example 21-19 displays the status of the IOS and IOS XR interfaces after three successive interface flap events over a few seconds interval. Notice that the penalty exceeds the 2000 suppress threshold, so the interface manager process has notified the routing protocol to suppress the routes for the interface. The reuse timer indicates how long the interface needs to



remain stable before the penalty value drops below the reuse threshold.

### Example 21-19 IP Event Dampening

[Click here to view code image](#)

```
R1#show interfaces dampening
GigabitEthernet1/2/3
  Flaps Penalty   Supp ReuseTm   HalfL  ReuseV  SuppV  MaxSTm  MaxP Restart
    3    2678     TRUE    17      15    1000   2000   60    16000    0

RP/0/RSP0/CPU0:ASR9001-B#show ip dampening interface gigabitEthernet 0/0/0/13
GigabitEthernet0/0/0/13 (0x04001380)
Dampening-enabled: Penalty 2678, SUPPRESSED (106 secs remaining)
  underlying-state: Up
  half-life:       1      reuse:           750
  suppress:       2000   max-suppress-time: 4
  restart-penalty: 0
```

### BGP Dampening

BGP route dampening applies a penalty for unstable route prefixes received from an eBGP neighbor. BGP dampening should be deployed with caution, as the feature may inadvertently filter business-critical networks. For example, dampening should not be applied to DNS subnets, especially root DNS servers.

BGP dampening may be applied for all routes or for a subset of routes using a route policy. The command to apply BGP dampening in IOS is **bgp dampening** [*half-life-time reuse suppress maximum-suppress-time* | **route-map** *route-map-name*] and IOS XR nodes use the command **bgp dampening** [*half-life* [*reuse suppress max-suppress-time*] | **route-policy** *route-policy-name*].

An IOS prefix-list or IOS XR prefix-set may be deployed with a BGP dampening route policy to specify which routes are eligible for dampening. Dampening values must be included within the route policy. The RFD eligible routes are identified using a permit statement in IOS or a pass statement in IOS XR, while the ineligible routes are denied in IOS or dropped in IOS XR.

Table 21-5 displays the default BGP dampening values for IOS and IOS XR

Parameter	IOS	IOS XR
Half-life	15 minutes	15 minute
Reuse	750	750
Suppress	2000	2000
Max-suppress time	Four times the half-life or 60 minutes	Four times the half-life
Restart penalty	0	0

Table 21-5 BGP Dampening Default Timers

#### Note

The RIR Routing Working Group estimates that only 3 percent of all Internet prefixes are responsible for 36 percent of the BGP update messages on the Internet. To reduce routing churn, while at the same time penalizing unstable routes, the group recommends the minimum suppress limit threshold should be set to at least 6000 to 12,000 when using BGP dampening.

**Example 21-20** demonstrates how to configure BGP dampening. The route policy BGP-DAMPENING is included in the configuration to demonstrate how to prevent prefixes of any length that reside in the 192.168.0.0/16 network from dampening. In the IOS example, notice that the prefix lists denies routes that are not subject to dampening. In the IOS XR RPL example, the dropped networks should not be dampened. All other network prefixes outside the range are passed and can be dampened by BGP.

The routers in the example are using the following dampening values:

- **Half-life time:** 15 minutes
- **Reuse value:** 750
- **Suppress value:** 6000
- **Max suppress time:** 60 minutes
- **Resulting max penalty:** 12,000

#### **Example 21-20** *BGP Dampening Configuration*

[Click here to view code image](#)

```
R1
router bgp 65001
  neighbor 10.0.12.2 remote-as 65002
  !
  address-family ipv4
    bgp dampening route-map BGP-DAMPENING
    network 192.168.1.1 mask 255.255.255.255
    neighbor 10.0.12.2 activate
  !
  ip prefix-list DAMPENING seq 5 deny 192.168.0.0/16 le 32
  ip prefix-list DAMPENING seq 10 permit 0.0.0.0/0 le 32
  !
  route-map BGP-DAMPENING permit 10
  match ip address prefix-list DAMPENING
  set dampening 15 750 6000 60
```

**XR2**

```
route-policy BGP-DAMPENING
  set dampening half-life 15 suppress 6000 reuse 750 max-suppress 60
  if destination in (192.168.0.0/16 le 32) then
    drop
  else
    pass
  endif
end-policy
!
router bgp 65002
  address-family ipv4 unicast
    bgp dampening route-policy BGP-DAMPENING
    network 192.168.3.3/32
  !
  neighbor 10.0.12.1
    remote-as 65001
    address-family ipv4 unicast
      route-policy pass in
      route-policy pass out
```

The command to view BGP dampening flap statistics in IOS is **show bgp ipv4 unicast dampening flap-statistics** and in IOS XR is **show bgp ipv4 unicast flap-statistics**.

**Example 21-21** demonstrates how to view the BGP prefix flap statistics. In the example, the prefix 10.100.100.0/24 has flapped seven times over a span of less than 10 minutes, which has triggered BGP to dampen the prefix. The duration timer indicates when the first flap event occurred, and the reuse timer indicates when the prefix will be unsuppressed as long as the prefix remains stable.

### **Example 21-21** BGP Dampen Flap Statistics

[Click here to view code image](#)

```
R1#show bgp ipv4 unicast dampening flap-statistics
BGP table version is 99, local router ID is 192.168.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
```

	Network	From	Flaps	Duration	Reuse	Path
*d	10.100.100.0/24	192.168.100.100	7	00:08:16	00:43:30	65005

```

RP/0/0/CPU0:XR2#show bgp ipv4 unicast flap-statistics
BGP router identifier 192.168.2.2, local AS number 65002
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0xe0000000 RD version: 109
BGP main routing table version 109
Dampening-enabled
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
                i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete

```

Network	From	Flaps	Duration	Reuse	Path
*d 10.100.100.0/24	192.168.100.100	7	00:08:16	00:43:30	65005 i

The command **show bgp ipv4 unicast dampened-paths** may be used to view actively dampened routes.

BGP dampened paths can be manually unsuppressed in IOS with the command **clear bgp ipv4 unicast dampening network mask** or in IOS XR with the command **clear bgp ipv4 unicast dampening network/length**.

## EVENT-DRIVEN FAILURE DETECTION

The first step in routing around a network failure is detecting that a problem has occurred. For example, a link failure can result in dropped packets for several seconds or minutes if the router continues to forward packets while waiting for the routing protocol hold timers to expire. Alternatively, event-driven detection, such as loss of signal (LoS), from a fiber cut, detects a link failure in fractions of a second and can significantly reduce the overall time required for routing convergence.

### Carrier Delay

Carrier delay is the amount of time the router will wait before notifying upper layer protocols that the physical hardware interface has detected a LoS. The IOS interface command to tune the detection value is **carrier-delay seconds** and the IOS XR interface command is **carrier-delay up seconds down seconds**. The default timer value for IOS is 2 seconds, and for IOS XR it is 0 seconds.

Obviously, the lower the carrier detect value is set, the quicker the link failure will be detected and the sooner routing convergence may begin. Having the carrier delay value set to 0 turns off any delay, which might be detrimental to routing stability if the link is constantly flapping. Therefore, it is recommended that IP event dampening be used together with carrier delay to prevent constant routing recalculations. IOS XR and the latest versions of IOS XE also include a carrier delay up value. It is recommended that the up value is set to a higher value to ensure that the interface does not report active unless the link has been stable for a fixed amount of time. [Example 21-22](#) demonstrates how to configure carrier delay.

### Example 21-22 Carrier Detect Configuration

[Click here to view code image](#)

```

IOS
interface GigabitEthernet1/2/3
carrier-delay msec 0

```

## IOS XR

```
interface GigabitEthernet0/0/0/13
carrier-delay up 3 down 0
```

### Note

The IOS optimization command **ip routing protocol purge interface** accelerates Forwarding Information Base (FIB) entry deletion by allowing routing protocols that are capable of responding to link failure to purge the route, instead of waiting for the less-efficient RIB process to walk the table to determine which routes are associated with the down interface. This command is enabled by default in Cisco IOS Software 15.1(2)S and later.

## BFD

In some environments, no carrier detect signaling mechanism is available for quickly detecting whether the link between the two routers is down. Figure 21-8 illustrates three types of environments where a link failure may not occur on the directly connected interface. When the link failure is not directly connected, the router needs to rely on the routing protocol keepalive messages to determine remote-end neighbor reachability. This can take an unacceptably long amount of time by today's standards. For example, OSPF by default waits 40 seconds to declare a neighbor down.

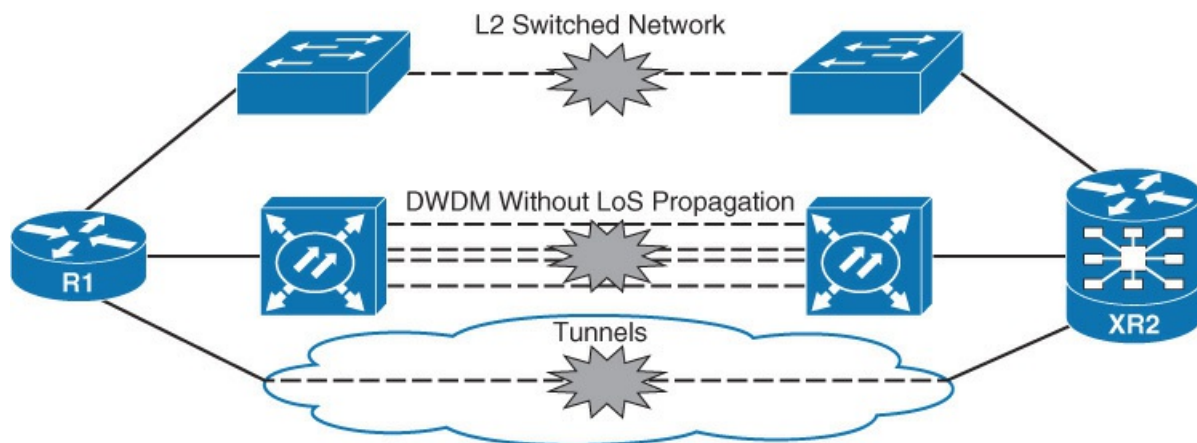


Figure 21-8 Loss of Signal Detection Challenges

One option for quickly identifying neighbor reachability loss is to set the hello and keepalive timers on the routing protocol to a very short interval. Introducing *fast hellos* does not always reduce the failure detection interval to a level where the network can route around the problem before time-sensitive applications notice the communication failure. In addition, fast hellos can tax the router's CPU and do not scale well as the number of neighbors sessions increase.

Bidirectional Forwarding Detection (BFD) is a simple lightweight hello protocol that routers can use on any media type for quickly detecting failures in the forwarding path. The BFD hello interval and hold timers for BFD are tunable, allowing for subsecond link failure detection.

### Note

BFD is commonly described as a lightweight hello protocol because of its small fixed packet header size. The packet's small size is less CPU-intensive to process than a more complex variable-length IGP routing protocol hello packet. On distributed routing platforms, the BFD packets are not punted to the RP CPU; instead, the line card processes the packets allowing for significant BFD session scalability.

BFD does not have an auto discovery mode. BFD treats routing protocols, such as OSPF, as clients for creating the BFD sessions. The routing protocol discovers the neighbor using its own detection mechanism and then uses this information to form the BFD session with the

neighboring router. If a link failure is detected by BFD, the client routing protocol is notified. This allows OSPF to tear down the routing neighbor adjacency immediately, instead of waiting multiple seconds for the hold timers to expire.

Note

Routing protocol hello and hold timers do not need to be modified to take advantage of BFD. In addition, if multiple routing protocols are enabled on the same interface with BFD, they will share the single BFD session.

RFCs 5880, 5881, 5882, and 5883 describe two modes of BFD operation:

■ **Asynchronous mode:** In asynchronous mode, routers periodically send control packets to activate and maintain BFD sessions.

Asynchronous mode is available in two submodes:

- Asynchronous mode without echo
- Asynchronous mode with echo

Asynchronous mode with echo is the default operating mode for Cisco routers.

■ **Demand mode:** In demand mode, the routers have an independent method of verifying connectivity to the other system. Once the BFD session is established, the routers are not required to share control packets with each other unless one side explicitly requests connectivity verification.

**Asynchronous Mode Without Echo**

BFD asynchronous mode without echo uses only control packets for negotiating session parameters and for detecting neighbor reachability. The BFD control packets have a UDP source port of 49152 and a destination port 3784.

Figure 21-9 illustrates an active BFD session between R1 and XR2. Notice that the BFD packet stream is unidirectional, meaning that there is not a request and response like with a client/server session. Instead, if the neighboring router does not receive the expected number of BFD packets in a row, the session is torn down.

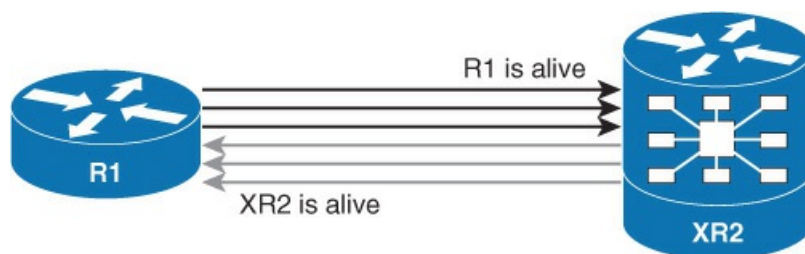


Figure 21-9 BFD Asynchronous Mode Without Echo

## Asynchronous Mode with Echo

Asynchronous mode with echo uses both control UDP 3784 and echo UDP 3785 packets for forming and maintaining the BFD session. The control packets are sent at a default rate of 2 seconds (BFD slow-timers rate), while the echo packets are sent at a high rate for fast neighbor detection. The neighboring router receiving the BFD echo packets does not actually process the packets contents; instead, the packets are simply looped back to the originating router, unchanged. The primary benefit of the echo function is that it more accurately tests the forwarding path between the two routers in comparison with the BFD without echo.

Figure 21-10 illustrates an active BFD session between R1 and XR2. Notice that XR2 loops the echo packets back to the originating router, R1.

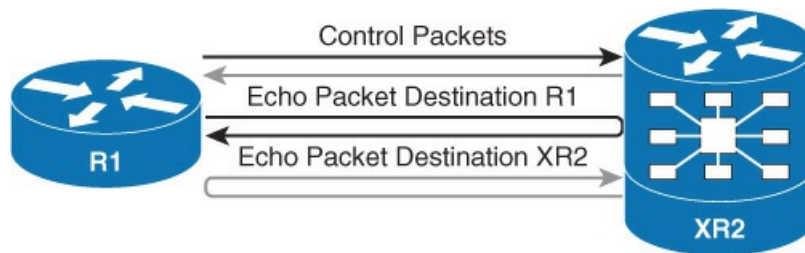


Figure 21-10 BFD Asynchronous Echo Mode

### Note

When unicast reverse path forwarding (uRPF) is enabled on an interface, asynchronous mode without echo is usually deployed to avoid inadvertently dropping looped echo packets. The IOS command to disable echo mode on an interface is `no bfd echo`. The IOS XR command is `bfd interface interface-type interface-number echo disable`.

Table 21-6 outlines the control and echo packets source and destination IP addresses and ports.

	Control Packets	Echo Packets
UDP destination	3784 (single) 4784 (multihop)	3785
UDP source	49152	3785
IP destination	Target IP address	IP address of outgoing interface
IP source	Local IP address identified by BFD client or IP address of outgoing interface	Selection order: 1. Interface-specific configured address 2. Global configured address 3. Router ID 4. IP address of outgoing interface

Table 21-6 BFD IP Addresses and UDP Ports

## BFD Configuration for OSPF, IS-IS, and EIGRP

In IOS, three steps are required for enabling BFD support for a routing protocol.

### Step 1. Disable sending Internet Control Message Protocol (ICMP) redirect messages.

BFD echo packets will trigger an IOS router to generate an ICMP redirect message back to the neighboring router because it believes a more optimal path exists out the local interface. The

ICMP message is unnecessary and can trigger high CPU utilization. The interface command **no ip redirects** disables the sending of ICMP redirect messages.

### Step 2. Enable BFD on an interface.

The BFD session parameters are enabled on the interface with the **bfd interval milliseconds min\_rx milliseconds multiplier multiplier-value** command.

The **interval milliseconds** setting advertises the desired transmit interval for the BFD packets, while the **min-rx milliseconds** setting advertises the desired receive interval for BFD packets. The **multiplier multiplier-value** indicates how many missed BFD packets will trigger a session failure.

The packet interval range for IOS routers is 50 to 999 milliseconds. The negotiated timer values may be changed at any time and are negotiated in each direction. The router requesting the slowest rate determines the value.

### Step 3. Configure the routing protocol client.

The routing protocol needs to be configured to use BFD. The routing protocol process configuration command **bfd all-interfaces** enables BFD on all interfaces associated with the routing protocol. BFD is enabled or disabled on a per-interface basis for EIGRP within the routing process:

- **Classic mode:** **bfd interface** *interface-type interface-number*

- **Named mode:** **af-interface** *interface-type interface-number* **bfd**

BFD can be enabled or disabled on a per-interface basis for OSPF or IS-IS with the following interface parameter commands:

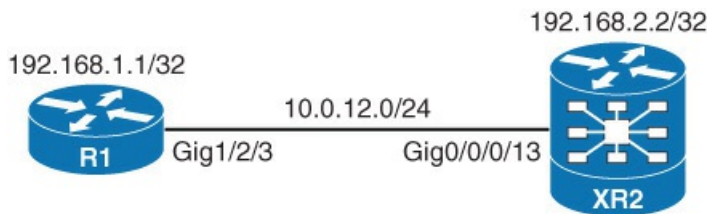
```
ip ospf bfd [disable]
```

```
isis bfd [disable]
```

In IOS XR, BFD activation and session parameters are both configurable under the dynamic routing protocol. The BFD session parameters are enabled within the routing protocol globally or on a per-interface basis. It is not necessary to disable **ipv4 redirects** because the feature is disabled by default on the interfaces. The command **bfd fast-detect** enables the routing protocol to react to BFD failure detection events. The commands **bfd minimum-interval milliseconds** and **bfd multiplier multiplier** manually set the desired transmit and receive packet interval. The range for IOS XR is 15 to 30,000 milliseconds.

Figure 21-11 illustrates the topology example for the BFD examples in this section.





**Figure 21-11** BFD Topology

**Example 21-23** demonstrates how to configure the routers to form an OSPF routing adjacency using BFD. BFD is set to 150 ms for all interfaces. BFD packets are sent every 50 ms with three consecutive missed BFD echo packets triggering a session failure (150 ms).

### Example 21-23 OSPF BFD Configuration

[Click here to view code image](#)

```
R1
interface GigabitEthernet1/2/3
 ip address 10.0.12.1 255.255.255.0
 no ip redirects
 bfd interval 50 min_rx 50 multiplier 3
!
router ospf 100
network 10.0.12.1 255.255.255.255 area 0
 bfd all-interfaces
```

```
XR2
router ospf 100
 bfd minimum-interval 50
 bfd fast-detect
 bfd multiplier 3
area 0
 interface GigabitEthernet0/0/0/13
```

#### Note

Less-sensitive BFD interval timers may be required for long-haul communication WAN links that have high latency. BFD timers of 500 ms \* 3 or higher may be desirable for older SSO-capable router platforms that terminate the BFD session on the RP, instead of offloading the session to line card hardware. The higher timers are necessary to avoid a session flap during the time it takes the system control plane to failover to the secondary RP.

**Example 21-24** demonstrates that the required OSPF interfaces have BFD enabled.

### Example 21-24 OSPF BFD Verification

[Click here to view code image](#)

```
R1#show ip ospf interface GigabitEthernet 1/2/3 | i BFD
Transmit Delay is 1 sec, State BDR, Priority 1, BFD-enabled
RP/0/RSP0/CPU0:XR2#show ospf interface gigabitEthernet 0/0/0/13 | i BFD
BFD-enabled, BFD interval 150 msec, BFD multiplier 3
```

The IOS command **show bfd neighbors [details]** and the IOS XR command **show bfd session [details]** verify that the BFD session is active between the routers. [Example 21-25](#) confirms that a BFD session successfully established between R1 and XR2.

### Example 21-25 BFD Session Status Verification

[Click here to view code image](#)

```
R1#show bfd neighbors
IPv4 Sessions
NeighAddr          LD/RD          RH/RS          State          Int
10.0.12.2          1/2148073480 Up              Up             Gi1/2/3
```

---

```
RP/0/RSP0/CPU0:XR2#show bfd session
Interface          Dest Addr          Local det time(int*mult)          State
                   Echo              Async
-----
Gi0/0/0/13         10.0.12.1         150ms (50ms*3)                   6s (2s*3)          UP
```

### BFD Configuration for BGP

The BGP fast peering session deactivation feature (fallover) is paired with BFD detection to ensure that a session is deactivated as soon as the peer's IP address is no longer reachable.

The IOS command to enable BFD support for a specific neighbor session is **neighbor ip-address fall-over [bfd]**.

#### Note

BGP fast peering session deactivation does not require BFD and may be deployed for neighbor sessions that do not support BFD. Without BFD enabled, the session is torn down only if the local interface detects LoS or the route to the peer changes.

In IOS XR, BFD fast session deactivation is configured with the command **bfd fast-detect** under the neighbor peering. The packet timers may be configured once globally for all BGP sessions or individually for each peering.

[Example 21-26](#) demonstrates how to configure BFD detection on the BGP peering between R1 and XR2.

### Example 21-26 BGP BFD Configuration

[Click here to view code image](#)

## IOS

```
interface GigabitEthernet1/2/3
 ip address 10.0.12.1 255.255.255.0
 no ip redirects
 bfd interval 50 min_rx 50 multiplier 3
!
router bgp 65001
 neighbor 10.0.12.2 remote-as 65002
 neighbor 10.0.12.2 fall-over bfd
!
 address-family ipv4
 neighbor 10.0.12.2 activate
```

## XR

```
router bgp 65003
 neighbor 10.0.12.1
 remote-as 65001
 bfd fast-detect
 bfd multiplier 3
 bfd minimum-interval 50
 address-family ipv4 unicast
 route-policy PASS in
 route-policy PASS out
```

**Example 21-27** demonstrates that the BGP routers are peering and that BFD fast detection is enabled.

### Example 21-27 OSPF BFD Configuration

[Click here to view code image](#)

```
R1#show bgp ipv4 unicast neighbors 10.0.12.2 | i BFD
 BFD is configured. BFD peer is Up. Using BFD to detect fast fallover (single-hop).
RP/0/RSP0/CPU0:XR2#show bgp ipv4 unicast neighbors 10.0.12.1 | i BFD
 BFD-enabled (session up): mininterval: 50 multiplier: 3
```

#### Note

Older router platforms do not support the usage of BFD and BGP NSF together. The latest IOS and IOS XE routers include support for RFC 5882 BFD control plane independent C bit, which allow for BFD and BGP NSF interoperability. The BGP command **neighbor ip-address fall-over bfd check-control plane-failure** allows the two features to work together at the same time. IOS XR routers support BGP NSF and BFD without the need for additional configuration requirements because BFD is offloaded to line card hardware making the sessions RP control plane independent.

### BFD Configuration for BGP Multihop

BGP can use BFD multihop to help improve convergence when the peer is not directly connected. The BFD echo function is disabled when using BFD multihop per the guidelines of RFC 5883.

An IOS multihop session is defined using a BFD template and BFD map:

1. The template command **bfd-template multi-hop template-name** defines the multihop session parameters.

2. The BFD map command **bfd map ipv4 destination-ip-prefix-length source-ip-prefix-length template-name** associates the source and destination address for the BFD session.

IOS XR requires identification of the specific line card that is to host the BFD multipath session with the command **bfd multipath include location node-id**.

Example 21-28 demonstrates how to configure an eBGP multihop session between R1's Loopback 0 interface (192.168.1.1/32) and XR2's Loopback 0 interface (192.168.2.2/32).

### Example 21-28 BGP Multihop BFD Configuration

[Click here to view code image](#)

```
IOS
bfd map ipv4 192.168.2.2/32 192.168.1.1/32 BGP-XR2
bfd-template multi-hop BGP-XR2
  interval min-tx 50 min-rx 50 multiplier 3
!
router bgp 65001
  neighbor 192.168.2.2 remote-as 65002
  neighbor 192.168.2.2 ebgp-multihop 2
  neighbor 192.168.2.2 update-source Loopback0
  neighbor 192.168.2.2 fall-over bfd
!
address-family ipv4
  neighbor 192.168.2.2 activate
```

```
XR
bfd
  multipath include location 0/0/CPU0
!
router bgp 65002
  neighbor 192.168.1.1
  remote-as 65001
  bfd fast-detect
  bfd multiplier 3
  bfd minimum-interval 50
  ebgp-multihop 2
  update-source Loopback0
  address-family ipv4 unicast
  route-policy pass in
  route-policy pass out
```

Example 21-29 demonstrates an active BGP multihop session using BFD detection.

### Example 21-29 BGP Multihop BFD Verification

[Click here to view code image](#)

```
R1#show bgp ipv4 unicast neighbors 192.168.2.2 | i BFD
  BFD is configured. BFD peer is Up. Using BFD to detect fast fallover (multi-hop).
```

## BFD Configuration for Static Routes

Static routes may use BFD sessions for monitoring the neighbor router's next-hop reachability. Unlike the routing protocols, static route BFD does not have a dynamic method of learning the peer address for forming the BFD session. An IOS router can designate a peer for the BFD response with the interface command **bfd neighbor ipv4 ip-address** or with a fully specified BFD-enabled static route.

The IOS command **ip route static bfd interface-type interface-number ip-address** identifies the neighbor router for the BFD session. Any additional static routes that include the BFD peer as a next-hop forwarding address will use BFD detection.

The IOS XR command to enable BFD for IPv4 static routes is **address-family ipv4 unicast ip-address next-hop bfd fast-detect [minimum interval interval] [multiplier multiplier]**.

[Example 21-30](#) demonstrates how to configure static routes to provide reachability between R1's and XR2's loopback interface. Notice that R1's configuration requires two static routes. The first static route identifies the target IP address 10.0.12.2 for forming the BFD session, and the second static route is for XR2's loopback address 192.168.2.2, which includes the next-hop forwarding address 10.0.12.2 that has BFD detection.

### Example 21-30 Static Route in Both Directions

[Click here to view code image](#)

```
R1
interface GigabitEthernet1/2/3
 ip address 10.0.12.1 255.255.255.0
 no ip redirects
 bfd interval 50 min_rx 50 multiplier 3
 !
ip route static bfd GigabitEthernet1/2/3 10.0.12.2
ip route 192.168.2.2 255.255.255.255 GigabitEthernet1/2/3 10.0.12.2
```

```
XR2
router static
 address-family ipv4 unicast
 192.168.1.1/32 GigabitEthernet0/0/0/13 10.0.12.1 bfd fast-detect minimum-interval
 50 multiplier 3
```

The same BFD session verification commands apply to static routes. The IOS command **show ip static route bfd** and the IOS XR command **show bfd session destination ip-address** may be used to ensure that the static route's next-hop address is being monitored correctly by BFD.

[Example 21-31](#) demonstrates how to verify the BFD session is active for the static route.

### Example 21-31 Static Route BFD Verification

[Click here to view code image](#)

```
R1#show ip static route bfd
Codes in []: R - Reachable, U - Unreachable, L - Loop, D - Not Tracked

GigabitEthernet1/2/3 10.0.12.2 [R]
```

```
RP/0/RSP0/CPU0:XR2#show bfd session destination 10.0.12.1
Tue Apr 15 01:05:04.364 UTC
Interface          Dest Addr          Local det time(int*mult)  State
                   Echo              Async
-----
Gi0/0/0/13         10.0.12.1         150ms (50ms*3)          6s (2s*3)          UP
```

Example 21-32 demonstrates how to configure a default static route on the IOS XR router to use BFD, while the IOS router includes BFD neighbor awareness to allow the session to form.

### Example 21-32 BFD Session with XR2 Side Static Route

[Click here to view code image](#)

```
R1
interface GigabitEthernet1/2/3
 no ip redirects
 bfd interval 50 min_rx 50 multiplier 3
 bfd neighbor ipv4 10.0.12.2
```

```
XR2
router static
address-family ipv4 unicast
 0.0.0.0/0 Gig0/0/0/13 10.0.12.1 bfd fast-detect minimum-interval 50 multiplier 3
```

#### Note

The IOS interface configuration command **bfd neighbor** is not currently available on Cisco IOS XR Software.

## FAST ROUTING CONVERGENCE

Once a router has detected a problem, it must still notify its neighbor of the failure, calculate a new best path, and then program the RIB/FIB forwarding tables before the network can converge. Historically, link-state routing protocol convergence timers focused on stability, instead of convergence speed. For example, before link-state advertisement (LSA) generation throttling was introduced, Cisco IOS Software waited a full 5 seconds after detecting a failure before generating an LSA message. For most modern applications, this is an unacceptably slow convergence time. The processing power today is significantly greater than when link-state routing protocols were first developed, and therefore the protocols can be safely modified to allow for faster convergence.

Figure 21-12 illustrates how the various routing protocol-tuning features contribute to event propagation and best path calculation after a link failure has been detected.

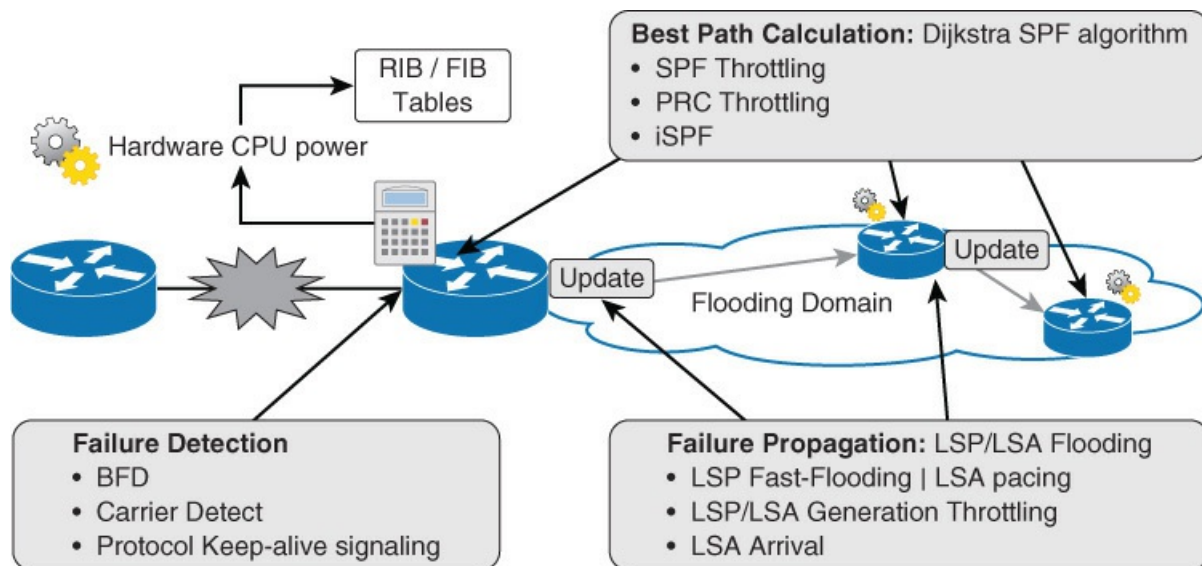


Figure 21-12 Link-State Routing Protocol Convergence

The OSPF and IS-IS link-state routing protocols propagate the failure information to neighboring routers in the same flooding domain to allow for the construction of a network connectivity map. Each router then independently performs a shortest path forwarding (SPF) calculation, called an *SPF run*, to determine the shortest path tree (SPT) for each destination network. The LSA/LSP flooding, along with the SPF run calculation, use a backoff throttling algorithm. The backoff timer allows for a quick reaction to a single event, but a delayed or more conservative stable reaction to a series of failures to reduce routing protocol churn. Throttling slows down convergence but is necessary for preventing uncontrolled routing fluctuations that can consume router resources and prevent routing convergence completion.

Table 21-7 describes the wait timers used by OSPF and IS-IS SPF backoff algorithm.

OSPF	ISIS	Description
SPF start	SPF initial wait	The initial time delay after a topology change before performing an SPF calculation.
SPF hold	SPF second wait	The holdtime delay is the interval between two consecutive SPF calculations. The holdtime delay interval doubles after each SPF calculation and will continue doubling if a route flap event is detected during the timer countdown.
SPF max wait	SPF max wait	The maximum interval allowed between consecutive SPF calculations.

Table 21-7 SPF Throttling Terminology

Figure 21-13 demonstrates how the exponential backoff algorithm works for SPF calculations. The illustration demonstrates SPF run throttling, but the concept is the same for link-state packet/advertisement (LSP/LSA) generation.

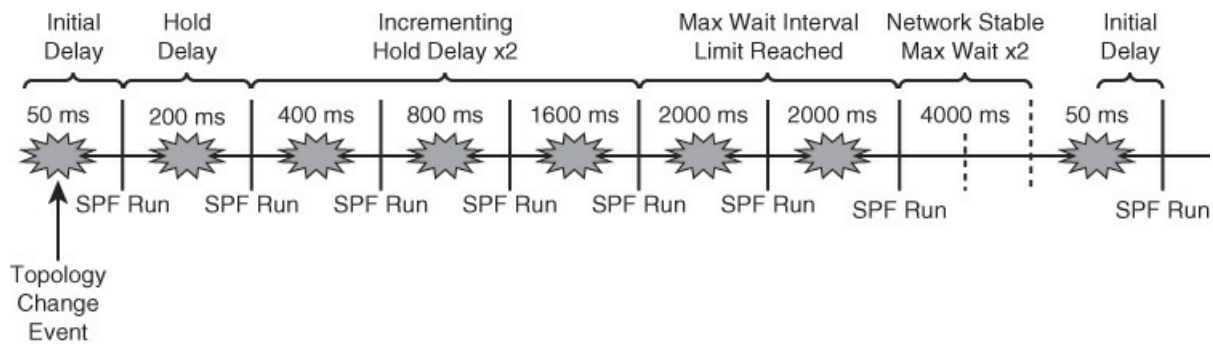


Figure 21-13 SPF Exponential Backoff Throttle Mechanism

The following IS-IS nondefault SPF throttling values are included for demonstrating the concept:

- **Initial holdtime start delay:** 50 milliseconds
- **Incrementing holdtime start delay:** 200 milliseconds
- **Maximum holdtime delay:** 2000 milliseconds

Notice how the router responds very quickly to the first route flap and performs an SPF calculation 50 milliseconds after learning of the event. The network is unstable, so each successive link flap delay doubles the incremental hold time delay of 200 milliseconds until finally the max holdtime delay limit of 2 seconds is reached. At this stage, the router waits a full 2 seconds before computing a best path for each successive link flap. The throttling hold timers do not reset until the network is stable for two full maximum wait holdtime intervals.

The next section of this chapter reviews the features that you can modify to reduce routing convergence times for the IS-IS and OSPF link-state protocols. It is important to note that IOS and IOS XR routers have different default LSP/LSA propagation and SPF processing values. Cisco IOS XR development engineers have tuned the values to allow convergence at a subsecond rate. Some of Cisco's largest service provider customers have safely used the values provided in the example configurations; but, of course, there is not a single "best practice" for all environments. The size of the network and the aggressiveness of the timer will determine the processing load that is placed on the router during periods of instability.

Modifying protocol timers can improve convergence times, but a high availability network requires a good network design. Where possible, the interfaces should be designated as network type point-to-point to avoid additional delays with the designated router (DR) and backup designated router (BDR) selection. A good design can improve convergence and stability by controlling LSA/LSP flooding through summarization and proper area placement.

### IS-IS Convergence Tuning

Table 21-8 lists the default IS-IS LSP propagation timer values for IOS and IOS XR.



Parameter	IOS	IOS XR
Maximum LSP lifetime	7500 seconds (2 hours, 5 minutes)	1200 seconds (20 minutes)
LSP refresh interval	900 seconds (15 minutes)	900 seconds (15 minutes)
Fast flooding	disabled	10 LSPs per interface
LSP generation throttling	LSP initial wait: 50 ms LSP second wait: 5 seconds LSP max wait: 5 seconds	Initial wait: 50 ms Secondary wait: 200 ms Maximum wait: 5000 ms

**Table 21-8** IS-IS Default LSP Flooding Values

The following configurable parameters influence IS-IS link-state PDU (LSP) flooding:

■ **Maximum LSP lifetime:** The maximum amount of time a LSP can be in a router's database without being refreshed.

IOS: **max-lsp-lifetime** *seconds*

IOS XR: **max-lsp-lifetime** *seconds* [**level** {**1** | **2**}]

■ **LSP refresh interval:** The interval at which the router will periodically refresh LSPs that it locally originates. Setting a high lifetime and refresh interval will reduce LSP generation, minimize routing protocol control traffic, and conserve router resources.

IOS: **lsp-refresh-interval** *seconds*

IOS XR: **max-lsp-lifetime** *seconds* [**level** {**1** | **2**}]

■ **LSP fast flooding:** The fast flood feature tells the router how many LSP packets to flood to neighboring routers before starting the local SPF calculation. Fast flooding is recommended when using aggressive SPF throttle timers to ensure that the LSPs that triggered the topology change are flooded to neighbor routers. Flooding the packets before the local SPF computation is run will ensure that neighbor routers process the change simultaneously, enabling faster convergence time throughout the entire network.

IOS: **fast-flood** [*lsp-number*]

IOS XR: Automatically fast floods ten LSP by default. Individual interface fast flooding packet counts can be modified with the command **lsp fast-flood threshold** [*lsp-number*].

■ **LSP generation throttling:** The amount of time the router waits before flooding new LSP updates to neighbors.

IOS: **lsp-gen-interval** [**level-1** | **level-2**] *lsp-max-wait* [*lsp-initial-wait* *lsp-second-wait*]

IOS XR: **lsp-gen-interval** [**initial-wait** *initial*] [**secondary-wait** *secondary*] [**maximum-wait** *maximum*] [**level** {**1** | **2**}]

Table 21-9 lists the default IS-IS SPF calculation interval values that affect best path calculation for IOS and IOS XR.

Parameter	IOS	IOS XR
SPF calculation throttling	SPF initial wait: 5500 ms	Initial wait: 50 ms
	SPF second wait: 5500 ms	Second wait: 200 ms
	SPF max wait: 10 seconds	Maximum wait: 5 seconds
PRC calculation throttling	PRC initial wait: 2000 ms	Initial wait: 50 ms
	PRC second wait: 5000 ms	Second wait: 200 ms
	PRC max wait: 5 seconds	Maximum wait: 5 seconds
Incremental SPF	Disabled	Disabled

Table 21-9 IS-IS Default SPF Calculation Intervals

The following parameters influence how long it takes IS-IS to process topology change notifications:

■ **SPF calculation throttling:** SPF calculation throttling is the amount of time the router waits before performing a new best path calculation for the entire database. IS-IS performs a full SPF run only when the physical topology changes (for example, if a transit link between two routers fails within a level).

IOS: **spf-interval** [**level-1** | **level-2**] *spf-max-wait* [*spf-initial-wait* *spf-second-wait*]

IOS XR: **spf-interval** [**initial-wait** *initial* | **secondary-wait** *secondary* | **maximum-wait** *maximum*] [**level** {**1** | **2**}]

■ **PRC calculation throttling:** A partial route calculation (PRC) calculates routes without needing to perform a full SPF computation. This may occur when the router learns new prefix information, yet the underlying physical network topology is the same. For example, a router changes the IP address of an interface.

IOS: **prc-interval** *prc-max-wait* [*prc-initial-wait* *prc-second-wait*]

IOS XR: PRC interval based on the settings of the **spf-interval**

Note

Take special notice that the LSP generation, SPF, and PRC calculation throttling parameters entry order is not the same in IOS as it is in IOS XR. In addition, the maximum-wait hold time is set in seconds for IOS and in milliseconds for IOS XR. All other parameters are set in milliseconds.

■ **Incremental SPF:** In large networks, the optional incremental SPF (iSPF) feature can accelerate routing table convergence. With iSPF enabled, the router does not perform a full SPT

computation for the entire Level 1 (L1) or Level 2 (L2) database when a link failure is detected; instead, only the branches of the SPT that are affected by the topology change are recalculated.

IOS: **ispf** {**level-1** | **level-2** | **level-1-2**}

IOS XR: **ispf** [**level** {**1** | **2**}]

Example 21-33 demonstrates how to configure the IS-IS tuning features discussed in this section. The example uses the following values to allow for subsecond routing convergence:

- **Maximum LSP lifetime:** 65535 seconds
- **LSP refresh interval:** 65000 seconds
- **Fast flooding** Enabled, 10 packets
- **LSP generation \ SPF \ PRC throttling:** 50 ms initial wait, 150 ms second wait, 5000 ms maximum wait
- **Incremental SPF:** Enabled for the Level 1 database.

Additional fast convergence features, such as BFD, carrier detect, and the IS-IS network type point-to-point are included in the sample configuration to demonstrate a full configuration. The IS-IS router is configured to form only L1 neighbor adjacencies in the example to avoid multiple adjacencies.

**Example 21-33** *IS-IS Protocol Tuning Configuration*

[Click here to view code image](#)

**IOS**

```
interface GigabitEthernet1/2/3
  dampening 15
  ip address 10.0.12.1 255.255.255.0
  no ip redirects
  ip router isis LAB
  carrier-delay msec 0
  bfd interval 50 min_rx 50 multiplier 3
  isis network point-to-point
!
router isis LAB
  net 49.0001.0000.0000.0001.00
  is-type level-1
  ispf level-1
  fast-flood 10
  set-overload-bit on-startup 180
  max-lsp-lifetime 65535
  lsp-refresh-interval 65000
  spf-interval 5 50 150
  prc-interval 5 50 150
  lsp-gen-interval 5 50 150
  no hello padding
  nsf cisco
  bfd all-interfaces
```

**IOS XR**

```
interface GigabitEthernet0/0/0/13
  ipv4 address 10.0.12.2 255.255.255.0
  carrier-delay up 3 down 0
  dampening
!
router isis LAB
  set-overload-bit on-startup 180
  is-type level-1
  net 49.0001.0000.0000.0002.00
  nsf cisco
  lsp-gen-interval maximum-wait 5000 initial-wait 50 secondary-wait 150
  lsp-refresh-interval 65000
  max-lsp-lifetime 65535
  address-family ipv4 unicast
  ispf level 1
  spf-interval maximum-wait 5000 initial-wait 50 secondary-wait 150
!
interface Loopback0
  passive
  address-family ipv4 unicast
!
!
interface GigabitEthernet0/0/0/13
  bfd minimum-interval 50
  bfd multiplier 3
  bfd fast-detect ipv4
  point-to-point
  hello-padding sometimes
  address-family ipv4 unicast
```

## OSPF Convergence Tuning

Table 21-10 lists the default OSPF LSA propagation timer values for IOS and IOS XR.

Parameter	IOS	IOS XR
Maximum LSA lifetime (RFC 2328)	3600 seconds (1 hour)	3600 seconds (1 hour)
LSA refresh interval (RFC 2328)	1800 seconds (30 minutes)	1800 seconds (30 minutes)
LSA packet pacing	33 ms	33 ms
LSA generation throttling	Start interval: 0 ms Hold interval: 5000 ms Max interval: 5000 ms	Start interval: 50 ms Hold interval: 200 ms Max interval: 5000 ms
LSA arrival	1000 ms	100 ms

Table 21-10 OSPF LSA Throttling

The following configurable parameters influence OSPF LSA flooding:

■ **Flood reduction:** The OSPF maximum LSA lifetime (1 hour) and LSA refresh interval (30 minutes) are standards defined by the IETF in RFC 2328. To reduce excessive LSA flooding in large networks, flood reduction can be enabled per interface. Flood reduction disables LSA aging over the specified interface, reducing network overhead. Once enabled, LSAs will be flooded again only if there is a topology change.

IOS: **ip ospf flood-reduction**

IOS XR: **flood-reduction enable**

### Note

OSPF flood reduction requires that the adjacent router supports the DC bit (RFC 3883).

■ **LSA packet pacing:** LSA packet pacing controls the interpacket timing gap between LSA updates. Pacing the packets ensures that an unstable network with a large amount of updates does not completely consume router resources.

IOS: **timers pacing flood milliseconds**

IOS XR: **timers pacing flood milliseconds**

■ **LSA generation throttling:** The amount of time the router waits before flooding new LSA updates to neighbors. Throttling reduces routing protocol churn by reducing network updates during periods of network instability.

IOS: **timers throttle lsa start-interval hold-interval max-interval**

IOS XR: **timers throttle lsa all** *start-interval hold-interval max-interval*

■ **LSA arrival:** The LSA minimal arrival timer determines the minimum amount of time that must pass before accepting the same LSA update.

IOS: **timers lsa arrival** *milliseconds*

IOS XR: **timers lsa min-arrival** *milliseconds*

Table 21-11 lists the default OSPF SPF calculation interval values that affect best path calculation for IOS and IOS XR.

Parameter	IOS	IOS XR
SPF calculation throttling	No throttling	Start interval: 50 ms Hold interval: 200 ms Max interval: 5000 ms
PRC calculation throttling	No throttling	No throttling
Incremental SPF	Disabled	Disabled

Table 21-11 OSPF Default SPF Calculation Intervals

The following parameters influence how long it takes OSPF to process topology change notifications:

■ **SPF calculation throttling:** SPF calculation throttling is the amount of time the router waits after receiving a topology change before performing a new best path calculation for the entire network. Throttling is a protection mechanism that ensures that an unstable link does not result in uncontrolled routing churn. When building the SPT in OSPF, all intra-area routers and networks (LSA Type 1 and 2) are considered nodes on the SPT and interarea network summary routes, Autonomous System Border Router (ASBR) summary, and autonomous system external redistributed prefixes (LSA Types 3, 4, 5) are considered the leaves, and therefore only LSA Type 1 or an LSA Type 2 will trigger a full SPF run.

IOS: **timers throttle spf** *spf-start spf-hold spf-max-wait*

IOS XR: **timers throttle spf** *start-interval hold-interval max-interval*

Note

The IOS OSPF throttle timers are entered in a different order than the IS-IS command. In OSPF, the initial SPF run is entered first, whereas in IS-IS the value is entered second after the maximum wait interval.

■ **PRC calculation:** OSPF supports partial route calculations for LSA Type 3, 4, and 5 topology changes. Cisco routers do not throttle PRC computations.

■ **Incremental SPF:** Incremental SPF can reduce router CPU utilization by minimizing the scope of SPF calculations. A link-state change does not usually affect the SPT for every node in

the network. With iSPF-enabled, a LSA Type 1 and 2 topology change does not trigger a full SPF run, only the branch of the tree that is affected is recalculated.

IOS: **ispf**

IOS XR: Not available

Note

The iSPF feature only affects the SPT computation on the local router. On most modern routers, there is minimal benefit to enabling this feature because there is enough computational power to perform a full SPF run in the same amount of time as an iSPF.

Example 21-34 demonstrates how to configure OSPF protocol tuning to allow for subsecond convergence.

- **LSA packet pacing:** 10 packets
- **LSA generation \ SPF throttling:**
  - 50 ms initial wait
  - 150 ms incremental hold interval
  - 5000 ms max wait
- **LSA minimal arrival:** 100 ms
- **Incremental SPF:** Enabled for IOS

**Example 21-34** *OSPF Protocol Tuning Configuration*

[Click here to view code image](#)

## IOS

```
interface GigabitEthernet1/2/3
  dampening 15
  ip address 10.0.12.1 255.255.255.0
  no ip redirects
  ip ospf flood-reduction
  ip ospf network point-to-point
  carrier-delay msec 0
  bfd interval 50 min_rx 50 multiplier 3
!
router ospf 100
  router-id 192.168.1.1
  ispf
  nsr
  nsf cisco
  timers throttle spf 50 150 5000
  timers throttle lsa 50 150 5000
  timers lsa arrival 100
  timers pacing flood 10
  passive-interface Loopback0
  network 10.0.12.1 0.0.0.0 area 0
  network 192.168.1.1 0.0.0.0 area 0
  bfd all-interfaces
```

## IOS XR

```
interface GigabitEthernet0/0/0/13
  ipv4 address 10.0.12.2 255.255.255.0
  carrier-delay up 3 down 0
  dampening
!
router ospf 100
  nsr
  router-id 192.168.2.2
  nsf cisco
  timers throttle lsa all 50 150 5000
  timers throttle spf 50 150 5000
  timers lsa min-arrival 100
  timers pacing flood 10
  area 0
  interface Loopback0
    passive enable
!
interface GigabitEthernet0/0/0/13
  bfd minimum-interval 50
  bfd fast-detect
  bfd multiplier 3
  network point-to-point
  flood-reduction enable
```

Cisco routers maintain a SPF run history, including the event that triggered the calculation. To view the OSPF SPF history, the IOS command is **show ip ospf statistics [detail]**, and the IOS XR command is **show ospf statistics spf [detail]**.



## SPF Prefix Prioritization

The speed a router can program the RIB and FIB forwarding tables directly correlates to the processing power of its hardware. The prefix prioritization feature improves routing convergence for critical networks by allowing installation into the RIB based on priority. Prefix prioritization provides the greatest benefit in large networks. Instead of processing thousands of prefixes in a linear manner after completing the SPF run, the most important network prefixes are loaded into the route table and CEF forwarding table first.

Table 21-12 lists the default prefix priority for IOS and IOS XR.

Protocol	IOS	IOS XR
IS-IS	<b>High:</b> None, may be matched by a tag	<b>Critical:</b> None, may be matched by an ACL
	<b>Medium:</b> /32 prefixes	<b>High:</b> None, may be matched by an ACL
	<b>Low:</b> All other prefixes	<b>Medium:</b> /32 prefixes
OSPF	<b>High:</b> /32 prefixes	<b>Low:</b> All other prefixes
	<b>Low:</b> All other prefixes	<b>Critical:</b> None, may be matched by an RPL
		<b>High:</b> None, may be matched by an RPL
		<b>Medium:</b> /32 prefixes
		<b>Low:</b> All other prefixes

Table 21-12 Default Prefix Priority

By default, IS-IS and OSPF give /32 prefixes special priority. The remaining prefixes default to the low-priority RIB processing queue. Customizing the priority queue disables the default priority for all prefixes. All prefixes that are not matched by the new policy, including /32 prefixes are moved to the low-priority queue.

Figure 21-14 illustrates a network using prefix prioritization to improve convergence on the IP TV subnet 10.0.1.0/24.



Figure 21-14 Delay and Packet-Loss Sensitive IP TV Network

In IOS, the IS-IS configuration mode command **ip route priority high tag tag-value** matches a prefix route tag value and assigns it to the high priority processing queue. Route tags may be assigned during redistribution or at the interface level with the command **isis tag tag-number**.

In IOS XR, prefixes are matched using an access list. The IS-IS address family command **spf prefix-priority [level {1 | 2}] {critical | high | medium} {acl-name | tag tag}** assigns the matched prefixes to the specified priority queue.

Example 21-35 demonstrates how to set the IP TV subnet to the high-priority queue in IOS and to the critical prefix priority queue in IOS XR. The /32 loopback prefixes are also manually set to the high-priority queue in both IOS and IOS XR. Notice that the IOS router performs matching using a route tag value, whereas in IOS XR, an ACL matches against the network address. To

ensure that the IOS router correctly prioritizes the IOS XR loopback prefix, a tag is added on the IOS XR router's loopback interface.

### Example 21-35 IS-IS Prefix Prioritization

[Click here to view code image](#)

```
IOS (R1)
interface GigabitEthernet1/2/3
 ip address 10.0.1.1 255.255.255.0
 ip router isis LAB
 isis tag 100
!
interface Loopback0
 ip address 192.168.1.1 255.255.255.255
 isis tag 100
!
router isis LAB
 ip route priority high tag 100
```

```
IOS XR
ipv4 access-list HIGH
 10 permit ipv4 host 192.168.1.1 any
 20 permit ipv4 host 192.168.2.2 any
!
ipv4 access-list CRITICAL
 10 permit ipv4 10.0.1.0/24 any
!
router isis LAB
 is-type level-1
 net 49.0001.0000.0000.0002.00
 address-family ipv4 unicast
 spf prefix-priority critical CRITICAL
 spf prefix-priority high HIGH
!
interface Loopback0
 passive
 address-family ipv4 unicast
 tag 100
```

In IOS, prefix prioritization is enabled with the OSPF command **prefix-priority high route-map route-map-name**, and in IOS XR it is enabled with the command **spf prefix-priority route-policy route-policy-name**.

#### Note

IOS sends high-priority OSPF prefixes to the RIB before low-priority prefixes of the same route type. For example, a high-priority external route will be processed before a low-priority external route, but after a low-priority interarea route.

IOS XR processes high-priority prefixes before low-priority prefixes regardless of route type.

[Example 21-36](#) demonstrates how to set the IP TV subnet to the high priority in IOS and the critical priority in IOS XR. The policy also matches all /32 prefixes and sets them to the high-

priority queue. Notice that OSPF is different from IS-IS because it uses a route map or RPL to match prefixes and set priority level.

### Example 21-36 OSPF Prefix Prioritization

[Click here to view code image](#)

```
IOS
ip prefix-list PRIORITIZATION seq 5 permit 10.0.1.0/24
ip prefix-list PRIORITIZATION seq 10 permit 0.0.0.0/0 ge 32
!
route-map RIB-HIGH permit 10
  match ip address prefix-list PRIORITIZATION
!
Router ospf 100
prefix-priority high route-map RIB-HIGH
```

```
IOS XR
route-policy RIB-PRIORITY
  if destination in (10.0.1.0/24) then
    set spf-priority critical
  elseif destination in (0.0.0.0/0 ge 32) then
    set spf-priority high
  endif
end-policy
!
router ospf 100
spf prefix-priority route-policy RIB-PRIORITY
```

The IOS command to view the prefix priority level is **show ip ospf rib** [*ip-address* | *ip-address-mask*]. In IOS XR, the prefix priority level is viewable with the command **show route** [*ip-address/prefix-length*] **detail**.

[Example 21-37](#) demonstrates how to view the priority level for the 10.0.1.0/24 prefix. In IOS, the priority has been set to high, and in IOS XR the priority has been set to critical for the prefix.

### Example 21-37 Verifying Prefix Priority

[Click here to view code image](#)

```
R1#show ip ospf rib 10.0.1.0 | i Flags
Flags: Connected, HiPrio
```

```
RP/0/0/CPU0:XR2#show ip route 10.0.1.0/24 detail | i RIB
Route Priority: RIB_PRIORITY_NON_RECURSIVE_CRITICAL (4) SVD Type RIB_SVD_TYPE_LOCAL
```

## BGP Convergence Tuning

This section reviews how to improve BGP convergence through next-hop tracking, accelerated advertisement updates, and tuning the underlying TCP protocol for faster update exchanges.

## Next-Hop Tracking

When there is a network failure, it can take BGP a significant amount of time to detect that the next-hop forwarding address for a prefix is no longer valid. The BGP scanner process is responsible for verifying that a prefix's next-hop forwarding address information is still valid in the global route table. The scanner process runs once every 60 seconds. The scanner interval is configurable with the BGP configuration command **bgp scan-time seconds**; the increase in CPU load is significant, however, so modifying the scan time to a lower value is usually not recommended.

### Note

The process of examining every route's path information and attributes (NLR) is commonly referred to as *walking the table*.

BGP next-hop tracking (NHT) is an event-driven failure detection mechanism where the RIB notifies BGP of the next-hop routing change, instead of having to wait for the BGP scanner process to periodically walk the table and discover the changes. BGP NHT is enabled by default in IOS and IOS XR. The BGP configuration command **bgp nexthop trigger enable** may be required in older versions of IOS software to turn on the feature.

BGP includes a trigger delay before reacting to a next-hop change to allow time for the IGP routing protocols to converge.

- **IOS:** 5 second delay

- **IOS XR:** 3 second delay (critical) / 10 second delay (non-critical)

IOS treats all routing changes for a BGP next-hop address the same, whereas IOS XR is more granular and considers a RIB route installation/withdrawal a critical event and a route metric change a noncritical event.

The NHT delay is configurable with the IOS BGP address family configuration command **bgp nexthop trigger delay seconds**. In IOS XR, the trigger delay is modified with the command **nexthop trigger-delay {critical milliseconds | non-critical milliseconds}**.

### Note

The next-hop trigger delay may be safely modified to a lower value of 0 or 1 if the IGP routing protocol has already been tuned for fast routing convergence. If the IGP has not been tuned or if there are frequent route flaps in the network, there could be a series of constant RIB changes. The reduced trigger delay can actually slow down convergence because the RIB implements NHT dampening. NHT dampening is not configurable and frequent next-hop route fluctuations can delay the BGP run by as much as 60 seconds.

The IOS global configuration command **ip routing protocol purge interface** is required if the NHT trigger delay is set to 0. This command ensures that if there is a local link failure that causes the next-hop address to be removed, the RIB process does not immediately notify BGP. Instead, the IGP protocols have a chance to converge and then notify BGP.

An optional route policy may be used for selective next-hop address tracking. The IOS command is **bgp nexthop route-map route-map-name** and the IOS XR address family command is **nexthop route-policy route-policy-name**. When creating an NHT route policy for IOS XR the policy must also include the source routing protocol of the next-hop prefix. The source routing protocol is conditionally matched in the route policy with the RPL configuration command **protocol in (protocol)**. The route policy can be setup to allow only certain routes to be next-hop addresses for the BGP prefixes. For example, it may be desirable to only allow point-to-point (/30

or /31) and loopback interfaces (/32) as valid next-hop addresses. This type of policy intentionally avoids recursion to determine whether the next-hop address is reachable for the BGP prefix. A default or summary route will no longer be accepted as a valid option, potentially allowing for accelerated network convergence.

A thorough understanding of the BGP peering arrangement is necessary before deploying selective NHT. It is important that valid routes are allowed by the policy because if the route fails the next-hop policy evaluation, the BGP entry is marked as inaccessible, and the route is not installed in the global route table. [Example 21-38](#) demonstrates a BGP path that is not usable because the next-hop address 10.0.23.2 fails the selective next-hop policy evaluation.

### **Example 21-38** *BGP Selective NHT Invalidating Next-Hop Address*

[Click here to view code image](#)

```
RP/0/0/CPU0:XR1#show bgp ipv4 unicast 192.168.1.1
! Output omitted for brevity
Paths: (1 available, no best path)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
Local
  10.0.23.2 (inaccessible) from 0.0.0.0 (192.168.3.3)
    Origin incomplete, metric 3, localpref 100, weight 32768, valid, redistributed
    Received Path ID 0, Local Path ID 0, version 0
```

[Example 21-39](#) demonstrates how to configure BGP NHT along with an optional selective next-hop tracking policy. The two routers are using a policy that limits candidate routes to prefixes with a /30 prefix length or greater and that are not already being learned from BGP. The trigger delay is set to 1 second. Notice that the IOS router sets the delay value using seconds and the IOS XR router uses milliseconds.

### **Example 21-39** *BGP with Selective Next-Hop Tracking*

[Click here to view code image](#)

```
IOS
ip prefix-list RECURSION-ROUTES seq 5 permit 0.0.0.0/0 le 29
!
route-map SELECTIVE-NHT deny 10
  match ip address prefix-list RECURSION-ROUTES
!
route-map SELECTIVE-NHT deny 20
  match source-protocol bgp 65000
!
route-map SELECTIVE-NHT permit 30
!
router bgp 65000
  address-family ipv4
    bgp nexthop route-map SELECTIVE-NHT
    bgp nexthop trigger delay 1
```

#### IOS XR

```
route-policy SELECTIVE-NHT
  if destination in (0.0.0.0/0 ge 30) and protocol in (connected, ospf 100, static)
  then
    pass
  endif
end-policy
!
router bgp 65000
 address-family ipv4 unicast
  nexthop route-policy SELECTIVE-NHT
  nexthop trigger-delay critical 1000
  nexthop trigger-delay non-critical 1000
```

### Minimum Route Advertisement Interval

A BGP router does not immediately propagate network updates to neighbors. Instead, a minimum route advertisement interval (MRAI) is enforced that delays the advertisement of a prefix withdraw or announcement. RFC 4271 recommends a MRAI interval value of 30 seconds for eBGP neighbors and 5 seconds for iBGP sessions. These high values help reduce routing churn on the Internet and allow for fewer update messages to be sent but also slow down convergence. In IOS and IOS XR, the default MRAI values are 0 seconds for iBGP sessions and 30 seconds for eBGP.

#### Note

In older versions of Cisco IOS Software, the MRAI values are 5 seconds for iBGP and 30 seconds for eBGP.

To accelerate BGP convergence the MRAI interval can be reduced or disabled all together when set to 0. The IOS command to change the interval is **neighbor ip-address advertisement-interval seconds**, and the IOS XR BGP neighbor command is **advertisement-interval seconds**.

### TCP Performance

BGP uses the TCP protocol to form sessions and send updates. The larger the TCP packet payload, the more information that can be shared at one time and the sooner the BGP network will converge. To maximize the TCP payload size, the router can use maximum transmission unit (MTU) discovery so that the largest possible frame size can be identified for sending the packets. MTU discovery is enabled by default in IOS. The IOS command for BGP MTU path discovery is **bgp transport path-mtu-discovery**. MTU path discovery is not enabled by default in IOS XR and needs to be turned on with the global command **tcp path-mtu-discovery**.

To minimize TCP chattiness and achieve a higher BGP control session throughput the TCP window size can be increased to a larger value, such as 65535 bytes, with the IOS command **ip tcp-window-size 65535** and the IOS XR command **tcp window-size 65535**.

#### Note

Multihop BGP sessions may encounter problems if a smaller MTU link is present in the path. The TCP MTU path discovery mechanism may not work when there are firewalls or asymmetric routing paths. The maximum segment size (MSS) can be set to a value within the tolerance of a transit router with the interface parameter command **ip tcp adjust-mss 500-1460** on IOS nodes. IOS XR routers set the TCP MSS size for the entire router with the command **tcp mss 68-10000**.

A TCP packet drop may slow down BGP convergence because the sending router has to retransmit all the packets from the previous window of data. Selective acknowledgments can be

activated, allowing only the missing data segment to be resent. Selective acknowledgments are enabled in IOS with the global command `ip tcp selective-ack` and in IOS XR with the command `tcp selective-ack`.

## FAST REROUTE

Fast reroute (FRR) is a mechanism for reducing the router's failure reaction time by quickly redirecting traffic around a failure. The router accomplishes this task by precomputing a RIB and FIB repair path entry before the failure event. When a failure is detected, the traffic is forwarded to the repair path within tens of milliseconds without having to wait for the network to converge.

### Loop-Free Alternate Fast Reroute

The purpose of FRR is to allow a router to quickly redirect traffic after detecting a link or adjacent node failure. The router, called the *protecting node*, accomplishes this task by precomputing a loop-free alternate (LFA) repair path prior to the primary path actually failing. Immediately following a primary path failure, traffic is rerouted over the repair path while the routing protocol converges around the failure and calculates a new best path.

LFA FRR does not perform signaling between neighboring routers and only provides local protection for traffic forwarding. The feature may be deployed on one router, a selection of routers, or on every router in the network.

Figure 21-15 illustrates a link failure between R1 and XR2. R1 has LFA FRR enabled and immediately redirects traffic to R3 without having to wait for the neighboring routers to recompute the network topology or even be aware that a failure has occurred. LFA FRR minimizes the local failure reaction time by eliminating the hundreds milliseconds of routing protocol delay that is normally present with IGP flooding and best path calculations. LFA FRR can redirect traffic to the precomputed backup path in less than 50 milliseconds once the failure event is detected.

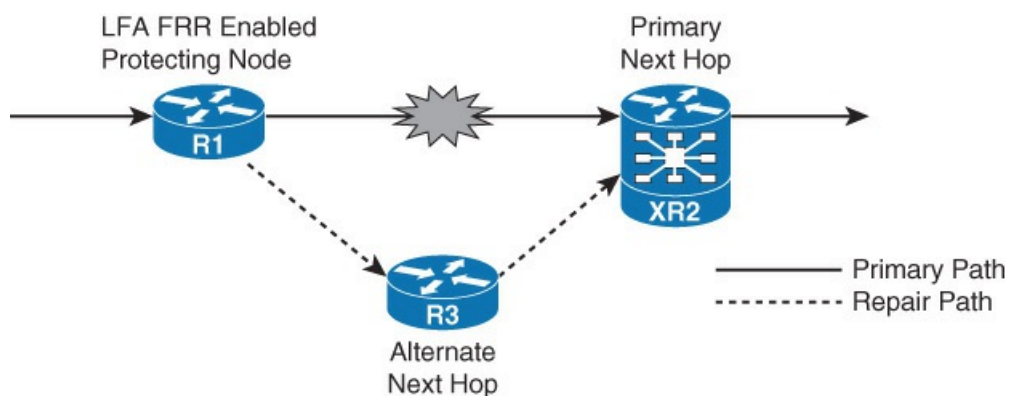


Figure 21-15 IP Loop-Free Alternate Fast Reroute

#### Note

Enabling LFA FRR does not impact the primary path convergence speed because the router always gives CPU priority to the primary path SPF calculation. The LFA FRR SPF run typically begins 500 ms after the network has converged.

### Loop-Free Condition Rules

A network with multiple redundant links may not necessarily meet the criteria for building a precomputed repair path. RFC 5286 outlines a set of formulas for determining whether a path meets the loop-free condition required for a repair path.

The following terms are used to describe the inequality condition formulas:

**N** = Neighbor router

**D** = Destination

**S** = Source router performing calculation

**E** = Neighbor router being protected

**PN** = Pseudonode

**Inequality 1: Loop Free**

Inequality condition 1 is the minimum requirement for precomputing a repair path and meeting the LFA condition. The neighboring router (N) should not expect that the protecting node (S) has the better path to reach the destination prefix (D).

**Formula:**  $\text{Distance (N, D)} < \text{Distance (N, S)} + \text{Distance (S, D)}$

Figure 21-16 demonstrates that the LFA repair path coverage depends on both the interface metric/cost and logical topology. The figure on the left passes the LFA available rule criteria, whereas the figure on the right does not because the metric between routers N, D is higher than the sum between routers N, S and S, D. The higher metric makes it impossible for S to use N as the repair path without temporarily introducing a short duration routing loop between the two routers while waiting for the routing protocol to converge using normal routing methods.

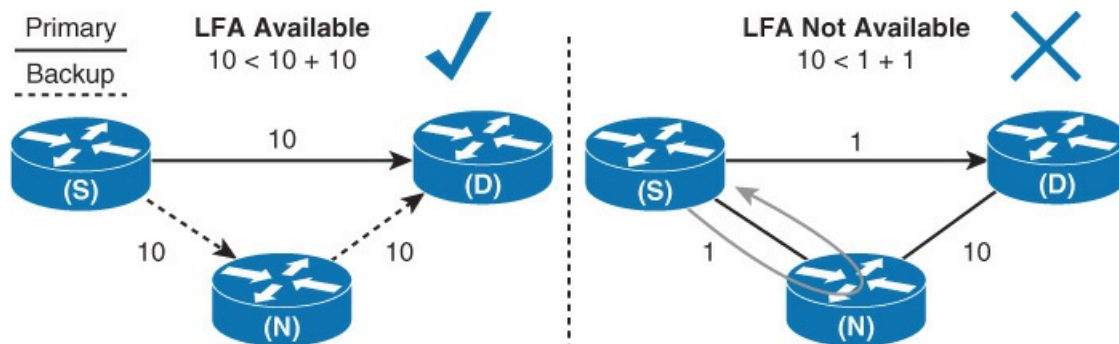


Figure 21-16 Inequality Condition 1: Loop-Free Alternate Protection

Note

A short duration routing loop is commonly called a *microloop* or *μloop*. The duration of the loop usually depends the convergence time of the slowest router in the network.

The repair path can be selected based on a set of more restrictive inequality conditions, which cover a wider range of failure scenarios. The more conditions that are met, the more resilient the repair path.

**Inequality 2: Downstream Path**

The neighboring router (N) is closer to the destination than the local router. This condition helps ensure that if there are multiple failures the neighbor router (N) will not form a loop and send traffic back to the computing router (S).

**Formula:**  $\text{Distance (N, D)} < \text{Distance (S, D)}$

Figure 21-17 illustrates that the downstream path between N, D must be a smaller overall metric



than the path between S, D.

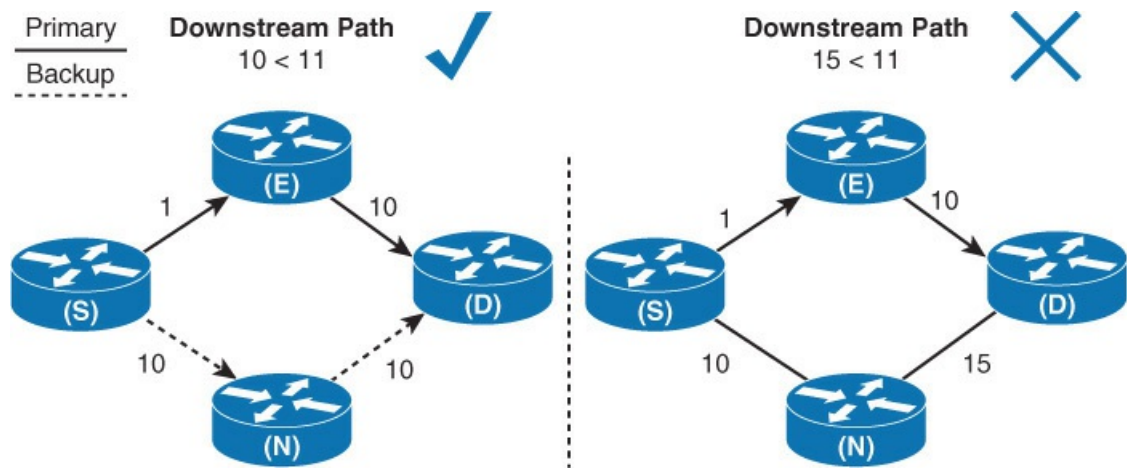


Figure 21-17 Inequality Condition 2: Downstream Path Protection

**Inequality 3: Node-Protecting Loop-Free Alternate**

The primary path between the source router (S) and the destination network (D) passes through the neighboring router (E). To ensure that a failure on E does not disrupt the repair path, the traffic directed to the next-hop protecting router (N) should not go through E.

**Formula:** Distance (N, D) < Distance (N, E) + Distance (E, D)

Figure 21-18 demonstrates that the repair path should not flow through router E to protect against a node failure on E.

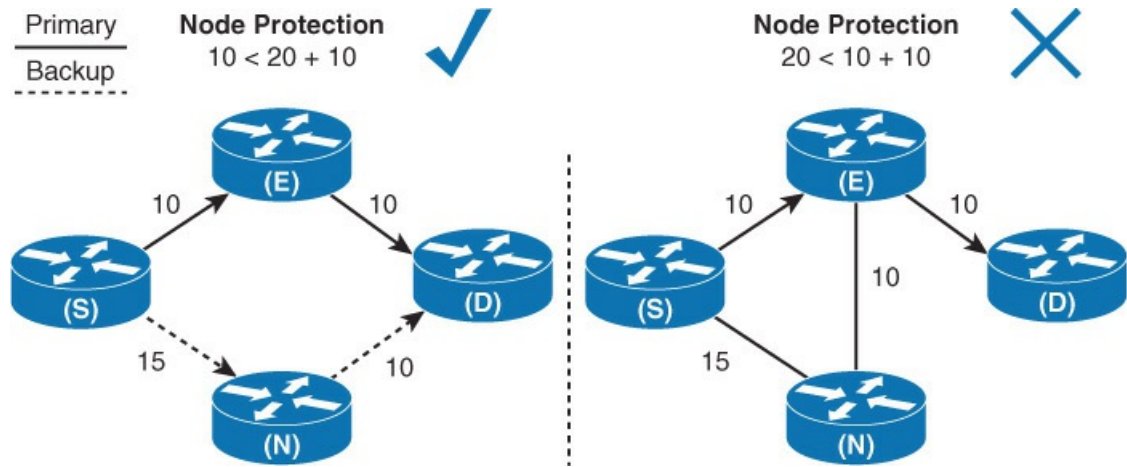


Figure 21-18 Inequality Condition 3: Node Protection

**Inequality 4: Loop-Free Link-Protecting for Broadcast Links**

A broadcast interface may have multiple attached routers to the same switch, and therefore it is possible to have different next-hop addresses for the primary path and repair path via the same link. The network failure may be the switch connecting the routers; therefore, the repair path should not cross the same broadcast network as the primary path.

**Formula:** Distance (N, D) < Distance (N, PN) + Distance (PN, D)

Figure 21-19 demonstrates that the repair path should not try to use the broadcast network to form a repair path. The pseudonode router (PN) represents the IS-IS DIS or OSPF DR for the LAN segment.

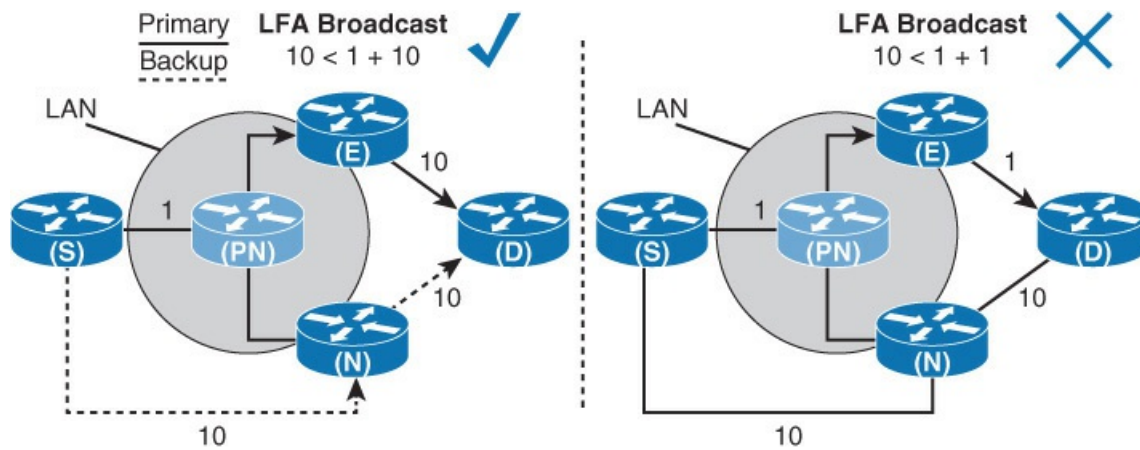


Figure 21-19 Inequality Condition 4: LFA Broadcast Protection

### LFA Protection Modes

The two IP configuration modes available for creating the LFA repair path are per-link and per-prefix.

■ **Per-link:** Per-link LFA protection examines the next-hop address of the primary link to determine whether a packet can be forwarded to another neighboring router without that neighbor in turn sending it back and forming a temporary routing loop while the IGP converges. Per-link LFA performs only a very simple inequality condition 1 “loop-free” check, and therefore there are minimal router CPU and memory requirements. All routes reachable through the primary protected link’s next-hop address share the same repair path. Therefore, either all the prefixes using the net-hop address of the primary link are protected or none of the prefixes are protected. Spreading the load of multiple prefixes over diverse repair paths is not possible using per-link mode, and there is no guarantee that the repair path provides path or node protection for the destination network.

■ **Per-prefix:** Per-prefix LFA performs a repair path computation for every prefix. This allows for an optimal repair path to be created for each destination network. Each prefix may have multiple candidate repair paths so a set of tiebreaker rules determine the best backup path. A tiebreaker attribute that eliminates all repair path candidates is skipped. Tiebreaker rule processing is sequential and finishes once a single path remains. If multiple candidate repair paths exist after processing the tiebreaker policy, the eligible repair paths are distributed among the prefixes to provide load sharing. Table 21-13 lists the repair path attributes available when using per-prefix LFA.

<b>Tiebreak Attribute</b>	<b>Description</b>
Shared risk link groups	Avoids candidate repair paths that belong to the same shared-risk link group (SRLG) as the primary path. An SRLG may be used to enhance repair path selection by identifying a set of links that share common resources and may fail simultaneously. For example, a set of interfaces may share the same fiber-optic path.
Equal-cost multi-path (ECMP) primary path	Avoids candidate repair paths that are not ECMPs. This option may be desirable when bandwidth overutilization over a single link is a concern.
ECMP secondary path	Avoids candidate repair paths that are ECMPs.
Interface disjoint	Avoid repair paths that share the same interface as the primary path.
Lowest repair path metric	Avoids candidate repair paths with a high metric. This option may not be desirable when high metric repair path provides more LFA protection coverage.
Line card disjoint	Avoids candidate repair paths that are on the same line cards as the primary path.
Node protecting	Avoids candidate repair paths that do not bypass the next-hop router of the primary path.
Broadcast interface disjoint	Avoids candidate repair paths that do not bypass the directly connected broadcast network of the primary path next hop.
Downstream	Avoids candidates repair paths that have downstream nodes with metrics to the destination higher than the protecting node metric to the destination.

**Table 21-13** LFA Repair Path Attributes

RFC 6571 provides an excellent analysis of per-link and per-prefix LFA coverage for 11 of the most common service provider network topologies. The findings indicate that, on average, per-link LFA provided 67 percent repair path coverage, whereas per-prefix LFA provided 89 percent coverage. In general, per-prefix LFA provides the best overall LFA coverage and is the recommended mode of operation in pure IP environments.

**Note**

Remote LFA (RQ) is a third method for calculating a repair path. Remote LFA is beyond the scope of this text because it not a pure native IP feature and requires Multiprotocol Label Switching (MPLS) to provide the LFA coverage.

Figure 21-20 is a reference network topology for the LFA FRR configuration examples.

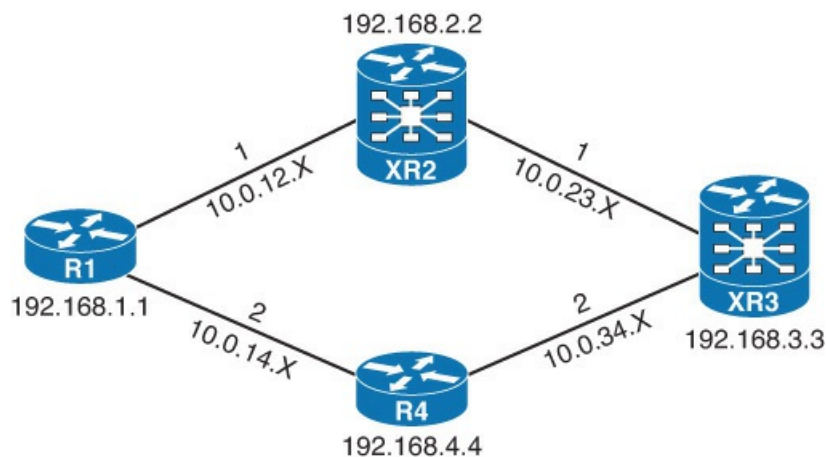


Figure 21-20 LFA FRR Topology Example

## OSPF LFAFRR

The protecting OSPF router has the following restrictions for selecting a repair path:

- The candidate repair path's interface must participate in the same area as where the primary path's interface exists.
- The backup route must be of the same type, such as intra-area, interarea, external, or external NSSA, and the prefix must use the same metric type as the primary path.
- Repair paths over virtual links that terminate on the protecting router (headend) are not supported.

In IOS, there is only one required step for enabling LFA IPFRR for OSPF.

### Step 1. Enable loop-free alternate fast reroute.

IOS routers support only per-prefix LFA. The OSPF command **fast-reroute per-prefix enable prefix-priority [area area-id] priority-level** enables per-prefix LFA. LFA may be enabled for the entire OSPF domain or for a specific area with the optional **area** keyword. The prefix priority level **low** enables repair path protection for all prefixes, and priority level **high** provides repair paths for /32 prefixes and or prefixes matched by an optional prefix-priority route map.

### Step 2. Modify repair path selection policy (optional).

The OSPF command **fast-reroute per-prefix tie-break attribute [required] index index-level** configures the repair path selection policy. The attributes are evaluated in sequential order, starting with the lowest index number. The default IOS repair path policy selection order is displayed here. Modifying a tiebreak command removes the default rules, requiring a new manual configuration for all desired rules:

1. srlg index 10
2. primary-path index 20
3. interface-disjoint index 30

4. lowest-metric index 40
5. linecard-disjoint index 50
6. node-protecting index 60
7. broadcast-interface-disjoint index 70

Default repair path selection policy is viewable with the command **show ip ospf fast-reroute**.

### **Step 3. Exclude an interface from the repair path (optional).**

The interface command **ip ospf fast-reroute per-prefix candidate disable** prevents an interface from being selected as next hop in a repair path.

### **Step 4. Disable protection on an interface (optional).**

The interface command **ip ospf fast-reroute per-prefix protection disable** disables LFA next-hop protection for an interface. Primary routes that point to this interface will not be protected.

IOS XR supports per-prefix or per-link LFA:

### **Step 1. Enable loop-free alternate fast reroute.**

LFA may be enabled in OSPF router, area, or interface configuration mode with the command **fast-reroute {per-link | per-prefix} [disable]**. The location at which the command is applied determines the scope of interfaces that are protected. Prefix protection can be selectively-enabled by priority levels with the command **fast-reroute per-prefix priority-limit [critical | high | medium]**. The default priority level for /32 prefixes is medium. All other prefixes default to low.

### **Step 2. Modify repair path selection policy (optional).**

The OSPF command **fast-reroute per-prefix tiebreaker attribute index index-number** configures the repair path selection policy. The attributes are evaluated in sequential order starting with the lowest index number.

The default IOS XR repair path policy selection order is as follows:

1. Primary path index 10
2. Lowest metric index 20
3. Line card disjoint index 30
4. Node protection index 40

The command **show ospf** displays the default per-prefix tiebreaker rules.

### Step 3. Modify repair path interface candidate list (optional).

The OSPF command **fast-reroute {per-prefix | per-link} use-candidate-only enable** excludes all interfaces from being candidate backup interfaces. Individual backup interfaces for the protected link are defined with the OSPF interface mode command **fast-reroute {per-prefix | per-link} lfa-candidate interface interface-type interface-number**.

[Example 21-40](#) demonstrates how to figure OSPF per-prefix LFA FRR protection for routers R1 and XR3 in the topology example. All prefixes are eligible for protection in the example configuration.

#### Example 21-40 OSPF LFA FRR Configuration

[Click here to view code image](#)

```
IOS
router ospf 100
  fast-reroute per-prefix enable prefix-priority low
```

```
IOS XR
router ospf 100
  fast-reroute per-prefix
```

The IOS command to verify LFA prefix coverage is **show ip ospf fast-reroute prefix-summary**, and the IOS XR command is **show ospf statistics fast-reroute**.

[Example 21-41](#) demonstrates how to view the LFA coverage for the example topology. Notice in the output that all of the networks are eligible for LFA protection, but only 66 percent have repair paths. The logic for 100 percent LFA coverage is described later in this section.

#### Example 21-41 LFA Coverage for [Figure 21-20](#)

[Click here to view code image](#)

```
R1#show ip ospf fast-reroute prefix-summary
! Output omitted for brevity
Interface          Protected   Primary paths   Protected paths   Percent protected
                  Yes        All  High  Low  All  High  Low  All High  Low
Lo0                Yes        0    0    0    0    0    0    0%  0%  0%
Gi2                Yes        4    2    2    3    1    2    75% 50% 100%
Gi1                Yes        2    1    1    1    0    1    50%  0% 100%

Area total:                6    3    3    4    1    3    66% 33% 100%

Process total:                6    3    3    4    1    3    66% 33% 100%
Process total:                5    3    2    3    1    2    60% 33% 100%
Process total:                6    3    3    4    1    3    66% 33% 100%
```

```
RP/0/0/CPU0:XR3#show ospf statistics fast-reroute
ospf_show_stats_ipfrr
OSPF 100 IPFRR Statistics:
  Number of paths: 6
  Number of paths-enabled for protection : 6 (100%)
  Number of paths protected: 4 (66%)
```

The IOS command to view the OSPF repair paths is **show ip ospf rib** or **show ip route repair-paths**. In IOS XR, the global route table includes (!) for the precomputed repair path. [Example 21-42](#) provides a summary view of the OSPF repair paths.

### Example 21-42 Repair Path Summary

[Click here to view code image](#)

```
R1#show ip ospf rib
      OSPF Router with ID (192.168.1.1) (Process ID 100)

      Base Topology (MTID 0)

OSPF local RIB
Codes: * - Best, > - Installed in global RIB
! Output omitted for brevity
*> 10.0.23.0/24, Intra, cost 2, area 0
    via 10.0.12.2, GigabitEthernet2
    repair path via 10.0.14.4, GigabitEthernet1, cost 5
*> 10.0.34.0/24, Intra, cost 4, area 0
    via 10.0.12.2, GigabitEthernet2
    repair path via 10.0.14.4, GigabitEthernet1, cost 4
    via 10.0.14.4, GigabitEthernet1
    repair path via 10.0.12.2, GigabitEthernet2, cost 4
```

```
RP/0/0/CPU0:XR3#show route
Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local, G - DAGR
       A - access/subscriber, a - Application route, (!) - FRR Backup path
! Output omitted for brevity
O   10.0.12.0/24 [110/0] via 10.0.34.4, 00:00:01, GigabitEthernet0/0/0/0 (!)
    [110/2] via 10.0.23.2, 00:00:01, GigabitEthernet0/0/0/1
O   10.0.14.0/24 [110/4] via 10.0.34.4, 00:00:01, GigabitEthernet0/0/0/0
    [110/4] via 10.0.23.2, 00:00:01, GigabitEthernet0/0/0/1
O   192.168.1.1/32 [110/0] via 10.0.34.4, 00:00:01, GigabitEthernet0/0/0/0 (!)
    [110/3] via 10.0.23.2, 00:00:01, GigabitEthernet0/0/0/1
```

[Example 21-43](#) demonstrates the forwarding entries for a single route. Notice that both the RIB and FIB are populated with the primary path (protected) and the repair path (backup).

## Example 21-43 Routing Table Entry For 192.168.3.3

[Click here to view code image](#)

```
! RIB ENTRY
R1#show ip route 192.168.3.3
Routing entry for 192.168.3.3/32
  Known via "ospf 100", distance 110, metric 3, type intra area
  Last update from 10.0.12.2 on GigabitEthernet2, 11:43:39 ago
  Routing Descriptor Blocks:
  * 10.0.12.2, from 192.168.3.3, 11:43:39 ago, via GigabitEthernet2
    Route metric is 3, traffic share count is 1
    Repair Path: 10.0.14.4, via GigabitEthernet1
```

```
! FIB ENTRY
R1#show ip cef 192.168.3.3
192.168.3.3/32
  nexthop 10.0.12.2 GigabitEthernet2
  repair: attached-nexthop 10.0.14.4 GigabitEthernet1
```

```
! RIB ENTRY
RP/0/0/CPU0:XR3#show route 192.168.1.1
Routing entry for 192.168.1.1/32
  Known via "ospf 100", distance 110, metric 3, type intra area
  Routing Descriptor Blocks:
  10.0.34.4, from 192.168.1.1, via GigabitEthernet0/0/0/0, Backup
    Route metric is 0
  10.0.23.2, from 192.168.1.1, via GigabitEthernet0/0/0/1, Protected
    Route metric is 3
```

```
! FIB ENTRY
RP/0/0/CPU0:XR3#show cef ipv4 192.168.1.1
192.168.1.1/32, version 56, internal 0x4000001 0x0 (ptr 0xacbe2e24) [1], 0x0
(0xacbde394), 0x0 (0x0)
local adjacency 10.0.23.2
  Prefix Len 32, traffic index 0, precedence n/a, priority 1
  via 10.0.34.4, GigabitEthernet0/0/0/0, 8 dependencies, weight 0, class 0, backup
[flags 0x300]
  path-idx 0 NHID 0x0 [0xacafc8c4 0x0]
  next hop 10.0.34.4
  local adjacency
  via 10.0.23.2, GigabitEthernet0/0/0/1, 8 dependencies, weight 0, class 0, pro-
tected [flags 0x400]
  path-idx 1 bkup-idx 0 NHID 0x0 [0xacf2a3b0 0x0]
  next hop 10.0.23.2
```

A review of the network topology reveals that the 192.168.x.x/32 networks advertised from directly adjacent neighbors fail the loop-free condition requirement rule 1.

Rule 1: Distance (N, D) < Distance (N, S) + Distance (S, D)

3 (R4, XR2) < 2 (R4, R1) + 1 (R1, XR2) – **Fails Condition**

3 (XR2, R4) < 1 (XR2, R1) + 2 (R1, R4) – **Fails Condition**



Figure 21-21 demonstrates that to provide 100 percent prefix repair path coverage, a link between XR2 and R4 is necessary.

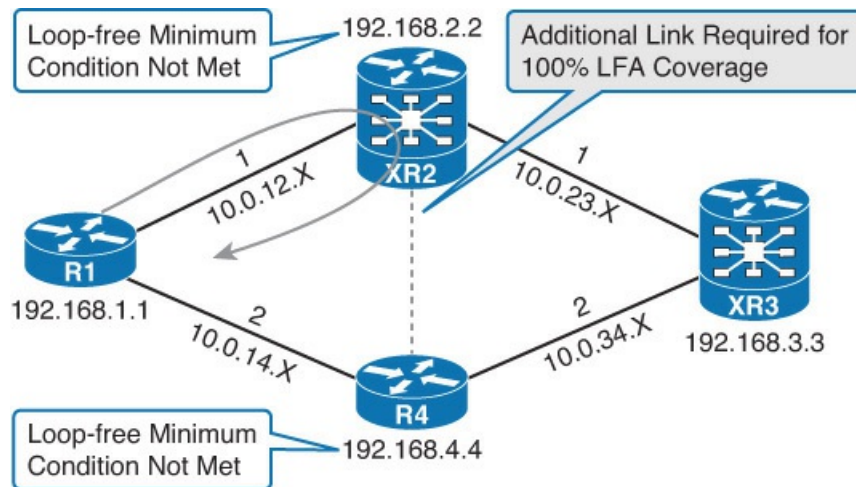


Figure 21-21 Requirements for 100 Percent LFA Coverage

Note

Network modeling tools, such as Cisco Cariden, are helpful for identifying gaps in LFA repair path coverage.

### IS-IS LFA FRR

Loop-free alternate FRR has the following requirements:

- Repair path links on the protecting node need to belong to the same level or area as the primary path interface.
- Cisco's IS-IS implementation of LFA protects only paths that go through point-to-point interfaces.

The IOS configuration procedure for IS-IS LFA FRR is as follows:

#### Step 1. Enable loop-free alternate fast reroute.

IOS routers only support per-prefix LFA. The IS-IS command **fast-reroute per-prefix {level-1 | level-2} {all | route-map route-map-name}** enables per-prefix LFA. All prefixes for a level may be protected using the **all** keyword or a subselection based on a route map match criteria. Both L1 and L2 routes can be protected by entering the command twice, once for each level.

#### Step 2. Ensure protected interfaces are point-to-point.

The interface command **isis network point-to-point** changes a broadcast interface to a point-to-point link.

#### Step 3. Modify repair path selection policy (optional).

The ISIS command **fast-reroute per-prefix tie-break attribute index-level** configures the

repair path selection policy. The default IOS repair path policy selection order is as follows:

1. srlg-disjoint index 10
2. primary-path index 20
3. lowest-backup-path-metric index 30
4. linecard-disjoint index 40
5. node-protecting index 50

Default repair path selection policy is viewable within the configuration with the command **show run all | i tie-break**.

#### **Step 4. Exclude an interface from the repair path (optional).**

The interface command **isis fast-reroute candidate {level-1 | level-2} disable** prevents an interface from being selected as next hop in a repair path.

On the primary protected interface, it is also possible to exclude a specific interface from being a candidate repair path using the command **isis fast-reroute exclude {level-1 | level-2} interface interface-type interface-number**.

#### **Step 5. Disable protection on an interface (optional).**

The interface command **isis fast-reroute protection {level-1 | level-2} disable** disables LFA next-hop protection for an interface. Primary routes that point to this interface will not be protected.

IOS XR supports per-prefix or per-link LFA for IS-IS:

#### **Step 1. Enable loop-free alternate fast reroute.**

LFA is-enabled in the IS-IS interface address family configuration mode with the command **fast-reroute {per-link | per-prefix}**. Similar to OSPF, IS-IS LFA prefix protection can be selectively enabled by priority level with the IS-IS address family command **fast-reroute per-prefix priority-limit [critical | high | medium] {level-1 | level-2}**. The default priority level for /32 prefixes is medium. All other prefixes default to low.

#### **Step 2. Ensure that protected interfaces are point-to-point.**

The IS-IS interface mode command **point-to-point** changes a broadcast interface to a point-to-point link.

#### **Step 3. Modify repair path selection policy (optional).**

The IS-IS command **fast-reroute per-prefix tiebreaker attribute index index-number level**

{1 | 2} configures the repair path selection policy.

The default IOS XR repair path policy selection order is as follows:

1. Primary path 10
2. Lowest metric 20
3. Line card disjoint 30
4. Node protection 40

#### Step 4. Modify repair path interface candidate list (optional).

The IS-IS address-family command **fast-reroute {per-prefix | per-link} use-candidate-only level {1 | 2}** excludes all interfaces from being candidate backup interfaces. Individual backup interfaces for the protected link are configurable within the IS-IS interface address family with the command **fast-reroute {per-prefix | per-link} lfa-candidate interface interface-type interface-number level {1 | 2}**.

Example 21-44 demonstrates how to configure IS-IS to use LFA FRR. In the configuration example, the routers are all in the same area and protect L1 routes. Notice that the interface network type is also set to point-to-point to fulfill the LFA requirements.

#### Example 21-44 IS-IS LFA FRR Configuration

[Click here to view code image](#)

```
IOS
interface GigabitEthernet1
  ip address 10.0.14.1 255.255.255.0
  no ip redirects
  ip router isis LAB
  isis network point-to-point
  isis metric 2
  bfd interval 50 min_rx 50 multiplier 3
!
interface GigabitEthernet2
  ip address 10.0.12.1 255.255.255.0
  no ip redirects
  ip router isis LAB
  isis network point-to-point
  isis metric 1
  bfd interval 50 min_rx 50 multiplier 3
!
router isis LAB
  net 49.0001.0000.0000.0001.00
  is-type level-1
  fast-reroute per-prefix level-1 all
  passive-interface Loopback0
  bfd all-interfaces
```

## IOS XR

```
router isis LAB
 is-type level-1
 net 49.0001.0000.0000.0003.00
 address-family ipv4 unicast
 !
 interface Loopback0
  passive
  address-family ipv4 unicast
  !
 !
 interface GigabitEthernet0/0/0/0
  bfd minimum-interval 50
  bfd multiplier 3
  bfd fast-detect ipv4
  point-to-point
  address-family ipv4 unicast
   fast-reroute per-prefix
   metric 2
 !
 interface GigabitEthernet0/0/0/1
  bfd minimum-interval 50
  bfd multiplier 3
  bfd fast-detect ipv4
  point-to-point
  address-family ipv4 unicast
   fast-reroute per-prefix
   metric 1
```

The IOS and IOS XR command **show isis fast-reroute summary** provides an overview of the prefix LFA coverage.

The repair path information is viewable in the local RIB with the IOS command **show isis rib** [*ip-address* | *ip-address-mask*] or **show ip route repair-paths** [*ip-address* | *ip-address-mask*]. In IOS XR, the repair paths are viewable with the command **show route** or **show isis fast-reroute detail** [*prefix/prefix-length*]

### Shared Risk Link Group

A shared-risk link group (SRLG) is a group of interfaces that share common physical paths or have a high likelihood of failing at the same time.

**Figure 21-22** illustrates the physical connectivity and logical routed connectivity between R1 and XR2. Notice that the top two links share the same optical transport path. If the L1 network supporting these links has a failure then two out of the three L3 routing paths will also go down. To minimize packet loss, the LFA FRR routers should select the bottom path as the precomputed FRR repair path by assigning shared risk link group to the interfaces.

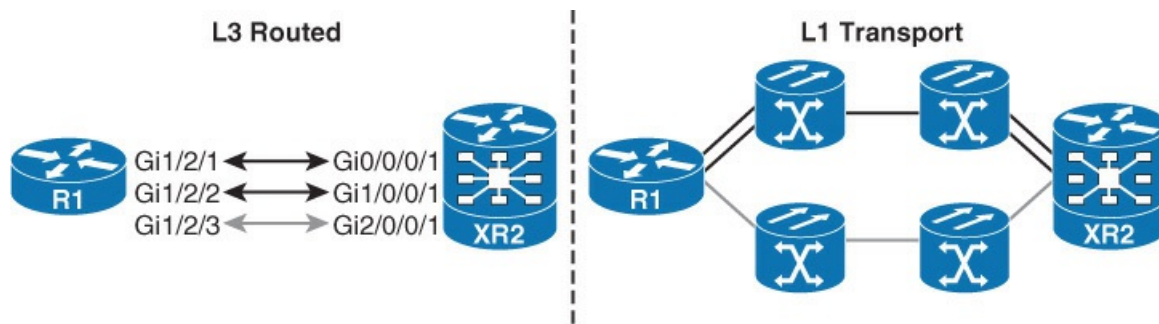


Figure 21-22 L3 Routed Network and L1 Transport Network

In IOS, an SRLG value is assigned with the interface command **srlg gid srlg-id**. In IOS XR, an SRLG value is assigned in SRLG configuration mode with the command **interface interface-type interface-number value value**. An interface may belong to zero, one, or multiple SRLGs.

Example 21-45 demonstrates how to configure SRLG value 12 for labeling the shared transport path between R1 and XR2. The tiebreak rules for OSPF and IS-IS LFA automatically prefer repair path candidates that are not members of the same SRLGs. LFA FRR only examines SRLG values that are locally configured on the protecting node and not the full path to the destination prefix.

### Example 21-45 Assigning SRLG to Interfaces

[Click here to view code image](#)

#### IOS

```
interface GigabitEthernet1/2/1
  srlg gid 12
!
interface GigabitEthernet1/2/2
  srlg gid 12
```

#### IOS XR

```
srlg
  interface GigabitEthernet0/0/0/0
    value 12
!
  interface GigabitEthernet0/0/0/1
    value 12
```

### BGP Prefix-Independent Convergence

Prefix-independent convergence (PIC) improves convergence by installing a backup/alternate path in the RIB and CEF forwarding tables for BGP networks. The precomputed backup path allows for subsecond convergence by immediately redirecting traffic to the backup path upon detection of a primary path failure or withdrawal. Traffic remains on the alternate path until the network reconverges around the failure and a new best path is identified.

The BGP PIC fast reroute solution consists of two key modules, called *PIC core* and *PIC edge*:

- PIC core is a failure in the IGP path toward the BGP prefix next hop. PIC core uses a hierarchical FIB table and depends on the IGP for fast convergence.
- PIC edge is a fault at the edge of the network that may be caused by either a link or node failure,

which results in the removal of the BGP prefix next-hop address. When this occurs, the PIC edge router redirects the traffic to the alternate BGP path.

Figure 21-23 illustrates the location for a PIC core or a PIC edge failure.

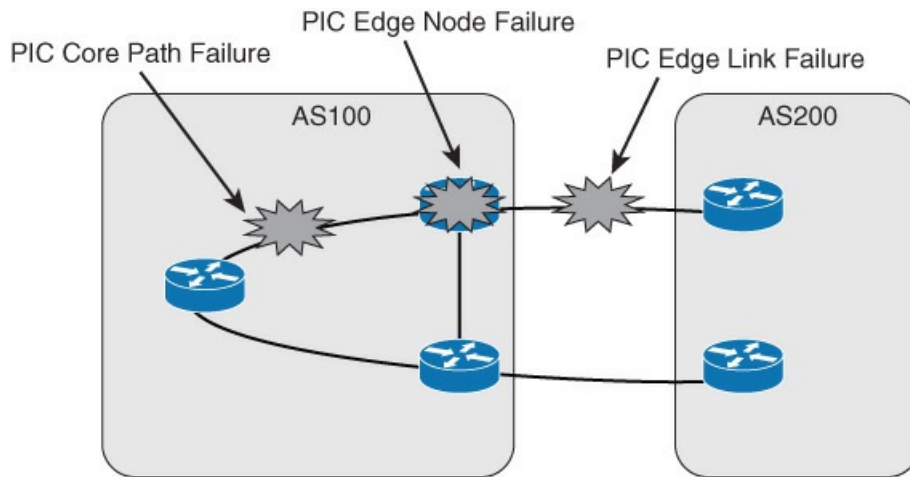


Figure 21-23 BGP PIC Overview

Note

BGP PIC does not support SSO with NSF.

### BGP PIC Core

BGP Internet tables can contain hundreds of thousands of prefix entries. The process of updating the FIB for such a large table can take many minutes to compile, as each individual prefix is evaluated and reprogrammed.

Figure 21-24 demonstrates a high-level view of a flat FIB table. Notice that each prefix has its own forwarding information directly associated to an outgoing interface (OIF) in a one-to-one correlation, which instructs the router how to forward the packet.

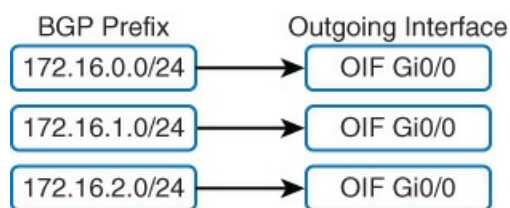


Figure 21-24 Flat FIB

BGP PIC core uses a hierarchical FIB that incorporates the concept of a shared path list and its in-place modification. In most environments, the majority of BGP prefixes point to a small number of IGP next-hop forwarding addresses. BGP PIC incorporates a pointer indirection between BGP and IGP entries so that a BGP prefix next hop that recurses to an IGP entry is immediately updated after the IGP converges.

Figure 21-25 demonstrates a hierarchical FIB shared path list table. Both single path and multipath BGP prefixes are represented to demonstrate how indirection and in place modification works. Only the objects of the FIB that are directly impacted by the topology change are updated. The shared path pointer enhancement in the hierarchical FIB allows convergence to

remain constant regardless of the size of the BGP table, which is why the feature is aptly named prefix-independent convergence.

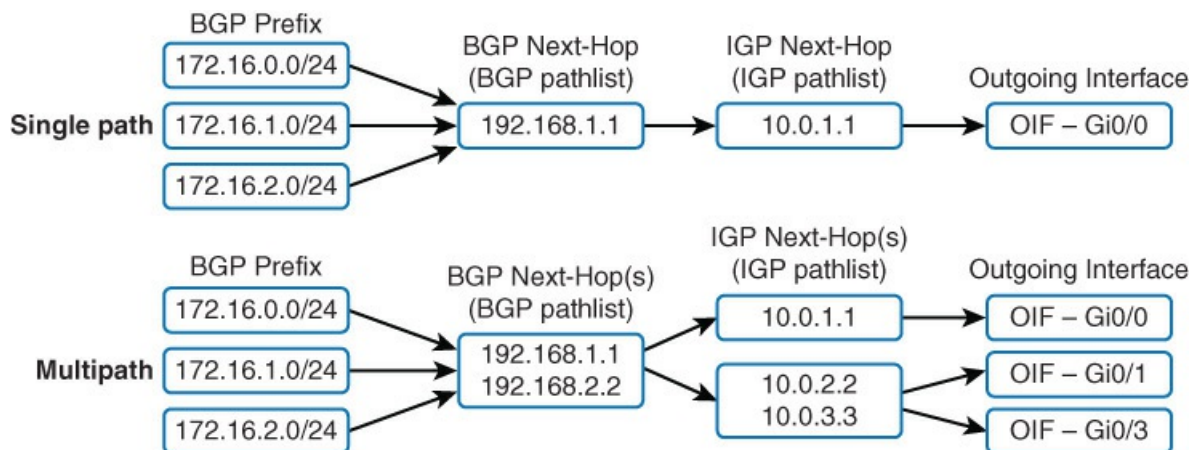


Figure 21-25 Hierarchical FIB Shared Path List

The IOS global configuration command **cef table output-chain build favor convergence-speed** enables BGP PIC core and in place FIB modifications on IOS platforms. IOS XR platforms do not require any additional configuration to enable PIC Core because they already use a hierarchical FIB by default.

Example 21-46 demonstrates how to configure BGP PIC Core on an IOS router.

### Example 21-46 BGP PIC Core

[Click here to view code image](#)

```
IOS
cef table output-chain build favor convergence-speed
```

### BGP PIC Edge

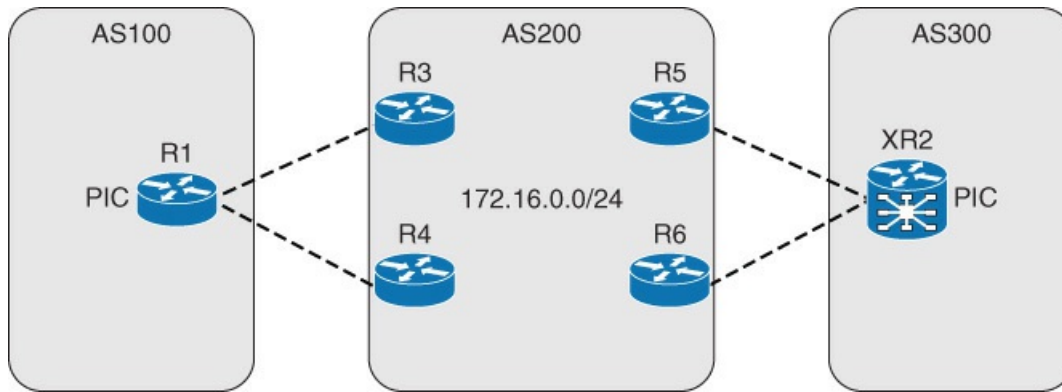
PIC edge redirects traffic to a backup path if the next-hop address of the primary path is lost due to a failure.

PIC edge is configured in IOS with the BGP address family command **bgp additional-paths install** and in IOS XR with the command **additional-paths selection route-policy route-policy-name**. Within the RPL, the **set** command **set path-selection backup 1 install** enables the installation of the backup path.

When the next-hop forwarding address fails, BGP normal routing behavior is to attempt to find the next longest matching path to reach the prefix. This behavior is not desirable when trying to achieve subsecond convergence and should be disabled when using PIC to ensure that traffic is immediately redirected to the backup path. The IOS BGP address family command to disable CEF recursion is **bgp recursion host**, and the IOS XR command is **nexthop resolution prefix-length minimum 32**.

Figure 21-26 provides a topology example that demonstrates the BGP PIC edge feature. The PIC configurations for R1 and XR2 are included in the example. BFD is not a strict configuration requirement for PIC edge, but is recommended for fast failure detection. For true bidirectional

fast convergence, the routers in AS200 also need to have PIC enabled.



**Figure 21-26** PIC for Protecting Multihomed Network

Example 21-47 demonstrates the configuration for BGP PIC edge.

### Example 21-47 BGP PIC Edge Configuration

[Click here to view code image](#)

```
R1
router bgp 100
 neighbor 10.0.13.3 remote-as 200
 neighbor 10.0.13.3 fall-over bfd single-hop
 neighbor 10.0.14.4 remote-as 200
 neighbor 10.0.14.4 fall-over bfd single-hop
 !
 address-family ipv4
  bgp additional-paths install
  bgp recursion host
  network 192.168.1.1 mask 255.255.255.255
  neighbor 10.0.13.3 activate
  neighbor 10.0.14.4 activate
 exit-address-family
```



**XR2**

```
route-policy ALTERNATE
  set path-selection backup 1 install
end-policy
!
router bgp 300
  address-family ipv4 unicast
    additional-paths selection route-policy ALTERNATE
    nexthop resolution prefix-length minimum 32
  network 192.168.2.2/32
  !
  neighbor 10.0.25.5
    remote-as 200
    bfd fast-detect
    bfd multiplier 3
    bfd minimum-interval 50
    address-family ipv4 unicast
      route-policy PASS in
      route-policy PASS out
  !
  neighbor 10.0.26.6
    remote-as 200
    bfd fast-detect
    bfd multiplier 3
    bfd minimum-interval 50
    address-family ipv4 unicast
      route-policy PASS in
      route-policy PASS out
```

Similar to LFA, the BGP PIC backup path is viewable in the global route table using the IOS command **show ip route repair-paths** [*ip-address* | *subnet-mask*] or the with the IOS XR command **show route**.

Example 21-48 demonstrates that the alternate route is viewable in the BGP table with the command **show bgp ipv4 unicast ip-address/prefix-length**.

**Example 21-48 BGP Prefix Primary and Backup Path**

[Click here to view code image](#)

```
R1#show bgp ipv4 unicast 172.16.0.0/24
BGP routing table entry for 172.16.0.0/24, version 14
Paths: (2 available, best #2, table default)
  Additional-path-install
  Advertised to update-groups:
    1
  Refresh Epoch 1
  200
  10.0.14.4 from 10.0.14.4 (192.168.4.4)
    Origin IGP, localpref 100, valid, external, backup/repair , recursive-via-con-
connected
    rx pathid: 0, tx pathid: 0
  Refresh Epoch 1
  200
  10.0.13.3 from 10.0.13.3 (192.168.3.3)
    Origin IGP, localpref 100, valid, external, best , recursive-via-connected
    rx pathid: 0, tx pathid: 0x0
```

```
RP/0/0/CPU0:XR2#show bgp ipv4 unicast 172.16.0.0/24
BGP routing table entry for 172.16.0.0/24
Paths: (2 available, best #1)
  Advertised to update-groups (with more than one peer):
    0.2
  Path #1: Received by speaker 0
  Advertised to update-groups (with more than one peer):
    0.2
  200
  10.0.25.5 from 10.0.25.5 (192.168.5.5)
    Origin IGP, localpref 100, valid, external, best, group-best, import-candidate
    Received Path ID 0, Local Path ID 1, version 7
    Origin-AS validity: not-found
  Path #2: Received by speaker 0
  Not advertised to any peer
  200
  10.0.26.6 from 10.0.26.6 (192.168.6.6)
    Origin IGP, metric 0, localpref 100, valid, external, backup, add-path,
import-candidate
    Received Path ID 0, Local Path ID 2, version 12
    Origin-AS validity: not-found
```

**Example 21-49** demonstrates that the IOS BGP table includes an additional *b* backup path status code for the alternate path for the prefix.

### **Example 21-49** IOS BGP Table

[Click here to view code image](#)

```

R1#show bgp ipv4 unicast
BGP table version is 13, local router ID is 192.168.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

```

	Network	Next Hop	Metric	LocPrf	Weight	Path
*b	172.16.0.0/24	10.0.14.4			0	200 i
*>		10.0.13.3			0	200 i
*>	192.168.1.1/32	0.0.0.0	0		32768	i
*b	192.168.2.2/32	10.0.14.4			0	200 300 i
*>		10.0.13.3			0	200 300 i

### BGP PIC Edge Link and Node Protection

The level of protection provided by BGP PIC edge depends on the location of the failure. If the failure is a link fault, the only routers that require backup paths are the autonomous system edge routers. However, if the edge router fails, the core routers will require a prepopulated alternate path in order to direct traffic around the failure.

Figure 21-27 illustrates link and node protection for BGP PIC edge. For PIC to succeed when there is an edge failure, the directly connected device requires a prepopulated alternate path to route around the failure.

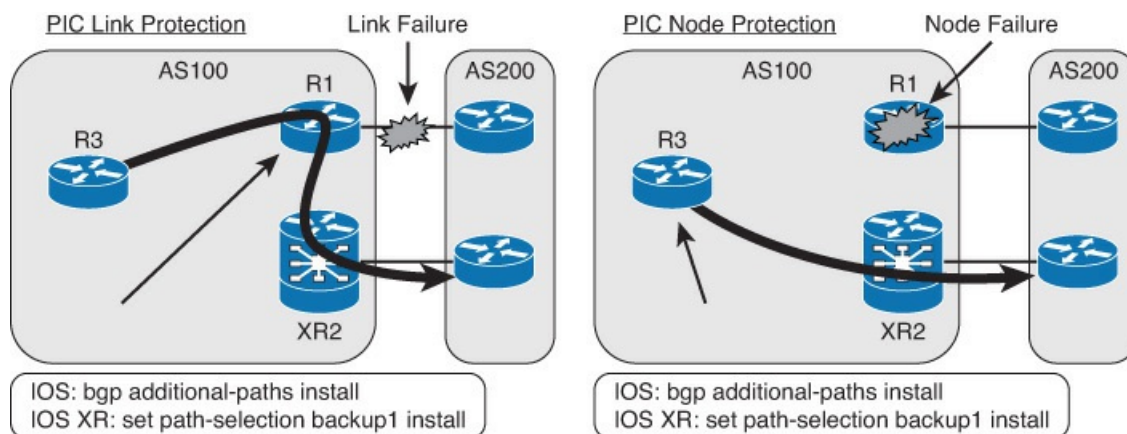


Figure 21-27 BGP PIC Link and Node Protection

### BGP PIC Edge with Next-Hop-Self

BGP edge routers that use next-hop-self to advertise prefixes into the network may inadvertently hide link failures and prevent fast reroute from succeeding.

Figure 21-28 illustrates how a routing loop may form when using next-hop-self to advertise prefixes. The internal network is blind to the reachability state of the link connected to R1. When this link fails, the forwarding address of the BGP prefix from AS200 remains unchanged because the next-hop-self feature is using the loopback address of R1. R1's alternate path is via XR2. The internal routers continue forwarding traffic to R1 after the failure event. Because R1 does not have a direct link to XR2, traffic is sent back to the core network forming a continuous loop until routing convergence completes using normal BGP mechanisms.

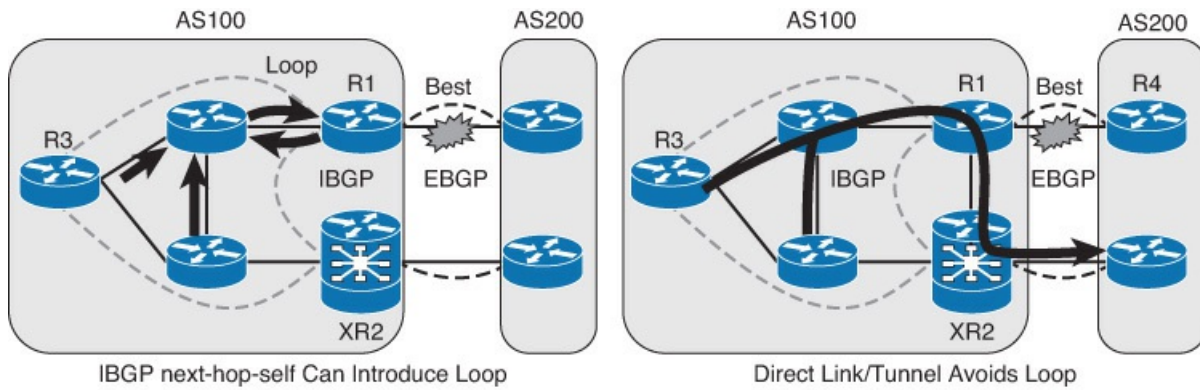


Figure 21-28 BGP PIC with Next-Hop-Self

The solution to this problem is to have a direct link or to tunnel the traffic directly to XR2 so that it may forward the traffic off-network without encountering the routing loop of the internal network.

### BGP Advertise Best External

The default behavior of BGP is to advertise only the single best path for a BGP prefix. Unfortunately, a router that has knowledge of only one path cannot participate in PIC because there is not an alternate path to converge to.

Figure 21-29 demonstrates a problem when using local preference to set a primary and backup path to reach AS200. XR2 has applied a route policy that sets the local preference for the 172.16.0.0/24 network to 5000. R1 observes that XR2 has the most desirable path to reach 172.16.0.0/24, so it withdraws its path to reach the network. Now the only known path to reach 172.16.0.0/24 on R3 and XR2 is via XR2, preventing the installation of a backup path for fast convergence.

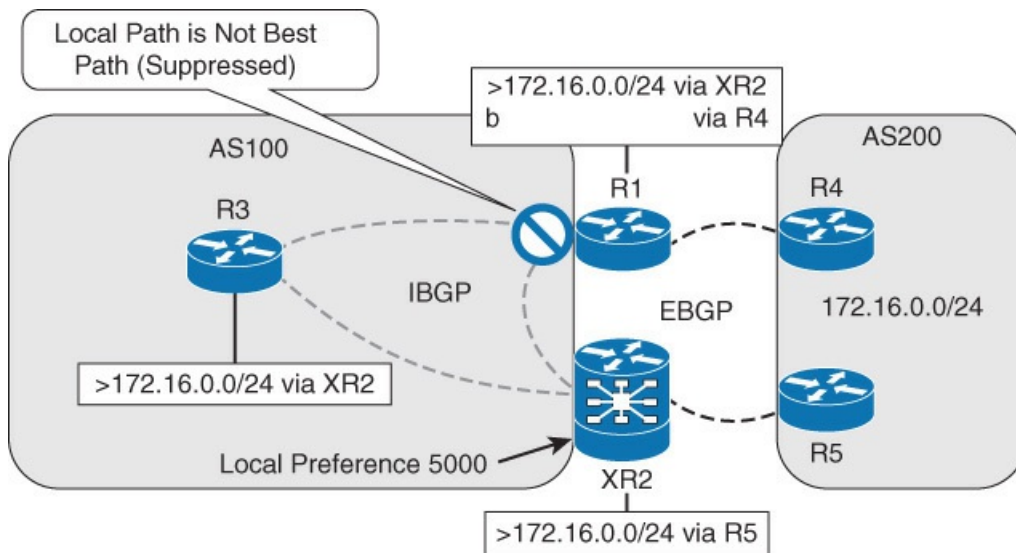


Figure 21-29 No Secondary Path Information

The IOS BGP address family command **bgp advertise-best-external** feature may be used by R1 to ensure that the additional external path is still advertised to the internal neighbor routers. The feature ensures that the external best paths are still advertised internally even when the iBGP neighbors have advertised a better internal path to reach the destination prefix.

In IOS XR, the address family command **advertise best-external** ensures the best external

path is still advertised internally when a better iBGP route is known.

BGP PIC is automatically enabled on the router when BGP **advertise-best-external** is configured.

### BGP Additional Path

One of the design challenges engineers face with BGP PIC is having the necessary alternate paths in the BGP table for precomputing backup repair paths. The regular BGP best path algorithm dictates that only the best path for a prefix is advertised to neighbor routers, so it is not uncommon for internal routers to have only one path for a prefix.

Figure 21-30 demonstrates how a route reflector can inadvertently hide alternate path information by advertising only the best path to clients.

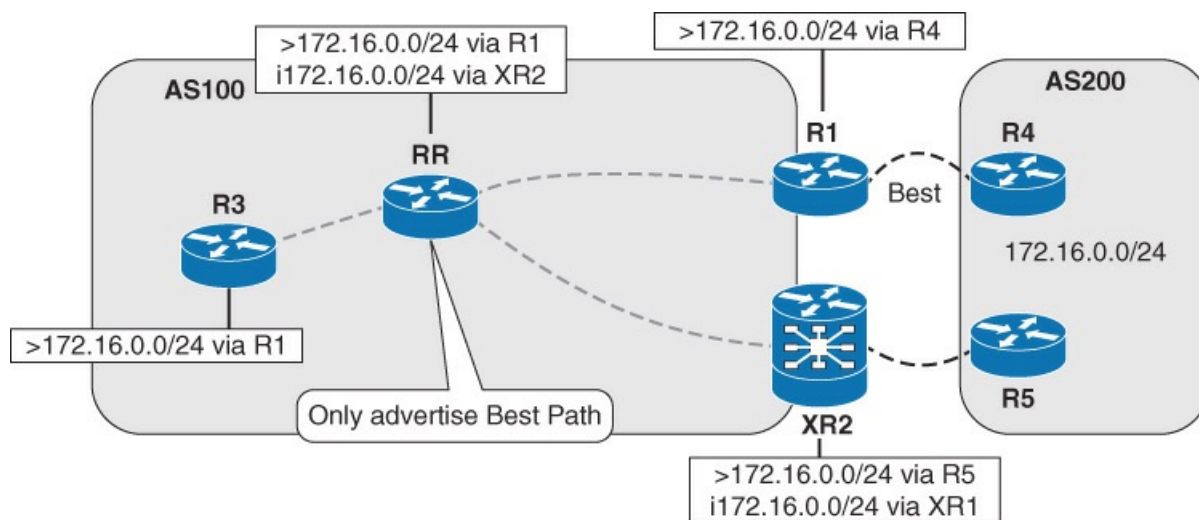


Figure 21-30 Route Reflector Suppresses Secondary Path Information

The BGP additional paths feature allows the advertisement of multiple paths and can be deployed along with BGP PIC to ensure that an alternate precomputed path is available for fast reroute.

#### Note

The BGP additional path feature increases memory utilization because the additional backup path increases the BGP table size.

There are three steps for enabling BGP additional path:

### Step 1. Enable add path capabilities.

BGP additional paths capability is negotiated between neighbors. A neighbor may send additional paths, receive additional paths, or both. The additional path for all neighbors is enabled in the BGP address family configuration mode:

IOS: **bgp additional-paths {send [receive] | receive}**

IOS XR: **additional-paths receive, additional-paths send**

The capability may also be enabled or disabled for individual neighbors:

IOS: **neighbor ip-address additional-paths disable**

IOS XR: **capability additional-paths send disable**

## Step 2. Identify candidate paths for advertisement.

In IOS, three candidate selection policies are available for identifying paths for advertisement:

- **bgp additional-paths select best number:** The best 2 or best 3 command option means that the best path along with the second or second and third best paths are advertised to the neighbor router.

- **bgp additional-paths select group-best:** The **group-best** option chooses the best path option from among the same autonomous system. So, if there are multiple paths to reach a prefix from an autonomous system, only one path is selected. If there are multiple paths to reach the prefix from two neighboring autonomous systems, two best paths are chosen, one best path for each autonomous system.

- **bgp additional-paths select all:** All paths with a unique next-hop forwarding address are advertised to the neighbor.

In IOS XR, the address family command **additional-paths selection route-policy route-policy-name** advertises the additional paths. The route policy may match a specific prefix or all prefixes based on the match criteria. Within the RPL, the add path capability is set with the command **set path-selection {backup number | group-best | all | best-path} [install] [multipath-protect] [advertise]**. IOS XR only allows the advertisement of the best and second best path for a prefix.

The RPL policy set condition has the following selection criteria:

- **Backup:** The second best path

- **Group best:** The best path for each neighboring autonomous system

- **All:** All BGP paths

- **Best-path:** Only the best path

Based on the selection, the router can either install a backup path, advertise the additional prefix, or both:

- **Install:** Installs backup path (PIC)

- **Multipath protect:** Install backup path for a multipath (PIC) and advertise

- **Advertise:** Advertise the additional path

The following example demonstrates how to install a backup path for PIC if the local community

of the prefix is 1:1. If the prefix is either 172.16.0.0/16 or 172.22.0.0/16, a backup path is installed, and the best and second best backup path for the prefix are advertised to the neighbor routers.

[Click here to view code image](#)

```
IOS XR
route-policy BACKUP
  if community matches-any (1:1) then
    set path-selection backup 1 install
  elseif destination in (172.16.0.0/16, 172.22.0.0/16) then
    set path-selection backup 1 advertise
    set path-selection backup 1 install
  endif
end-policy
```

### Step 3. Advertise paths to neighbors.

The final step is to activate the additional path selection policy for each neighbor with the IOS command **neighbor ip-address advertise additional-paths [best number] [group-best] [all]**. IOS XR routers do not require Step 3.

Figure 21-31 illustrates the same network, but with the additional path feature enabled. With add path feature enabled, every router now has a backup path to reach the prefix 172.16.0.0/24 in AS200.

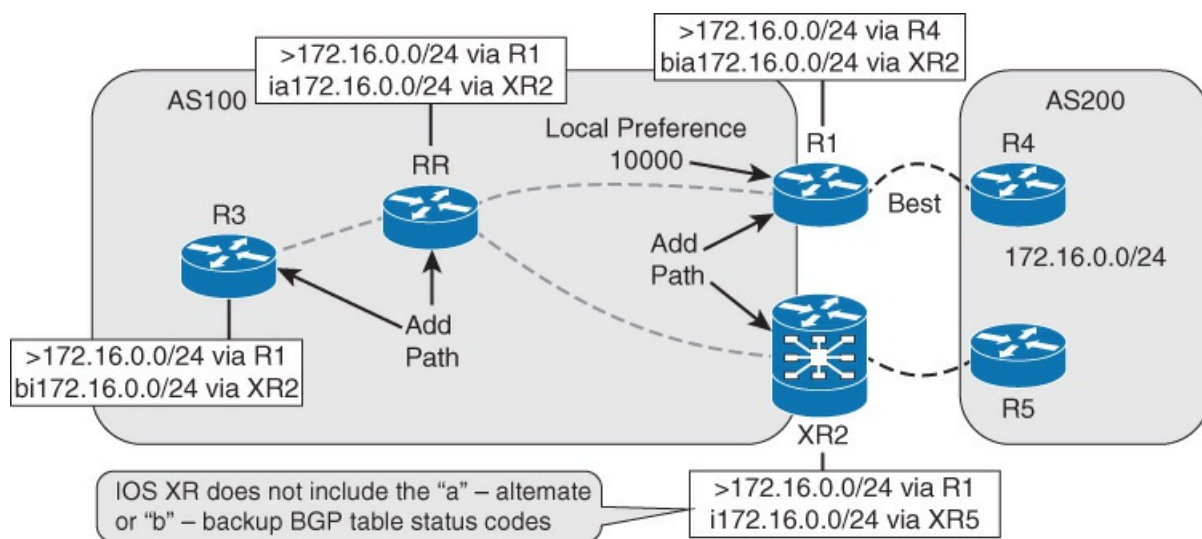


Figure 21-31 Full PIC protection with the Help of BGP Add Path

#### Note

The IOS BGP table includes special status codes to indicate that a PIC backup path is installed or an a additional path is advertised.

Example 21-50 demonstrates the BGP add path configuration for every router in AS100. In the topology, every iBGP router is configured to support the additional path capability. The route reflector is not part of the data forwarding path, so it has not been configured to install a BGP PIC backup path.

## Example 21-50 BGP PIC Edge and Additional Path Configuration

[Click here to view code image](#)

### R1

```
! Output omitted for brevity
router bgp 100
  neighbor 10.0.1.2 remote-as 200
  neighbor 192.168.100.100 remote-as 100
  neighbor 192.168.100.100 update-source Loopback0
  !
  address-family ipv4
    bgp additional-paths select all
    bgp additional-paths send receive
    bgp additional-paths install
    bgp recursion host
  neighbor 10.0.1.2 activate
  neighbor 192.168.100.100 activate
  neighbor 192.168.100.100 next-hop-self
  neighbor 192.168.100.100 advertise additional-paths all
  exit-address-family
```

### XR2

```
! Output omitted for brevity

route-policy BACKUP
  set path-selection backup 1 advertise install
end-policy
!
router bgp 100
  address-family ipv4 unicast
    additional-paths receive
    additional-paths send
    additional-paths selection route-policy BACKUP
    nexthop resolution prefix-length minimum 32
  !
  neighbor 10.0.3.5
    remote-as 200
    address-family ipv4 unicast
      route-policy PASS in
      route-policy PASS out
    !
  !
  neighbor 192.168.100.100
    remote-as 100
    update-source Loopback0
    address-family ipv4 unicast
      next-hop-self
    !
  !
```



### R3

```
! Output omitted for brevity
router bgp 100
  neighbor 192.168.100.100 remote-as 100
  neighbor 192.168.100.100 update-source Loopback0
  !
  address-family ipv4
    bgp additional-paths select all
    bgp additional-paths send receive
    bgp additional-paths install
    bgp recursion host
  neighbor 192.168.100.100 activate
  neighbor 192.168.100.100 additional-paths receive
exit-address-family
```

### RR

```
! Output omitted for brevity
interface Loopback0
  ip address 192.168.100.100 255.255.255.255
  !
router bgp 100
  neighbor CLIENT peer-group
  neighbor CLIENT remote-as 100
  neighbor CLIENT update-source Loopback0
  neighbor 192.168.1.1 peer-group CLIENT
  neighbor 192.168.2.2 peer-group CLIENT
  neighbor 192.168.3.3 peer-group CLIENT
  !
  address-family ipv4
    bgp additional-paths select all best 2
    bgp additional-paths send receive
  neighbor CLIENT route-reflector-client
  neighbor CLIENT advertise additional-paths best 2
  neighbor 192.168.1.1 activate
  neighbor 192.168.2.2 activate
  neighbor 192.168.3.3 activate
exit-address-family
```

**Example 21-51** demonstrates how to view the BGP neighbor relationship to confirm that the additional path capability has successfully negotiated.

### **Example 21-51** *Confirming BGP Add Path Capability Exchange*

**Click here to view code image**

```
R4-RR#show bgp ipv4 unicast neighbors 192.168.100.100
! Output omitted for brevity
For address family: IPv4 Unicast
  Additional Paths send capability: advertised and received
  Additional Paths receive capability: advertised and received
```

```
RP/0/0/CPU0:XR3#show bgp ipv4 unicast neighbors 192.168.100.100
```

```
! Output omitted for brevity
```

```
AF-dependent capabilities:
```

```
Additional-paths Send: advertised and received
```

```
Additional-paths Receive: advertised and received
```

```
Additional-paths operation: Send and Receive
```

**Example 21-52** demonstrates how to view which paths are advertised to the neighbor routers. In the example, the route reflector is advertising two paths for the prefix 172.16.0.0/16, while the edge router XR2 is advertising the local eBGP path.

### **Example 21-52** *Viewing Additional Path Prefix*

**Click here to view code image**

```
RR#show bgp ipv4 unicast 172.16.0.0/24
```

```
BGP routing table entry for 172.16.0.0/24, version 76
```

```
Paths: (2 available, best #2, table default)
```

```
Path advertised to update-groups:
```

```
4
```

```
Refresh Epoch 1
```

```
200, (Received from a RR-client)
```

```
192.168.3.3 (metric 3) from 192.168.3.3 (192.168.3.3)
```

```
Origin IGP, metric 0, localpref 100, valid, internal, best2, all
```

```
rx pathid: 0x1, tx pathid: 0x1
```

```
Path advertised to update-groups:
```

```
4
```

```
Refresh Epoch 2
```

```
200, (Received from a RR-client)
```

```
192.168.1.1 (metric 3) from 192.168.1.1 (192.168.1.1)
```

```
Origin IGP, metric 0, localpref 10000, valid, internal, best
```

```
rx pathid: 0x0, tx pathid: 0x0
```

```

RP/0/0/CPU0:XR2#show bgp ipv4 unicast 172.16.0.0/24
BGP routing table entry for 172.16.0.0/24
Versions:
  Process          bRIB/RIB   SendTblVer
  Speaker          51         51
Paths: (2 available, best #2)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Advertised to peers (in unique update groups):
    192.168.100.100
  200
    10.0.3.5 from 10.0.3.5 (172.16.2.1)
      Origin IGP, metric 0, localpref 100, valid, external, backup, add-path
      Received Path ID 0, Local Path ID 3, version 51
      Origin-AS validity: not-found
  Path #2: Received by speaker 0
  Not advertised to any peer
  200
    192.168.1.1 (metric 3) from 192.168.100.100 (192.168.1.1)
      Origin IGP, metric 0, localpref 10000, valid, internal, best, group-best,
import-candidate
      Received Path ID 0, Local Path ID 1, version 44
      Originator: 192.168.1.1, Cluster list: 192.168.100.100

```

## SUMMARY

The aim of high availability is to achieve continuous network uptime by designing a network to avoid single points of failure, incorporate deterministic network patterns, and use event-driven failure detection to provide fast network convergence. The high availability features outlined in this chapter can minimize the impact of a network failure.

The following features prevent packet loss by routing through a failure or proactively routing around a failure:

- NSF and NSR separate the control plane from the forwarding plane, allowing traffic to continue to flow even if the RP experiences a hardware or software failure.
- RFD suppresses unstable routes and can be used to avoid sending traffic over sections of the network that have a recent history of unreliability.

The features that can accelerate network routing convergence are as follows:

- Event-driven failure detection, such as carrier delay and BFD, may be used in place of routing protocol keepalive timers to accelerate failure detection from tens of seconds to milliseconds.
- The link-state routing convergence time can be optimized from 1 to 5 seconds to under 1 second through aggressive LSP/LSA propagation and SPF process tuning.
- BGP convergence can be improved through fast peering deactivation, next-hop tracking, accelerated advertisement updates, and tuning the underlying TCP protocol for faster update exchanges. A precomputed backup path provides a fast reroute forwarding option for packets while waiting for the routing protocols to converge around a network failure.

■ Loop-free avoidance and BGP PIC provide fast reroute capabilities by calculating a backup route before the failure occurs. When a failure is detected, traffic is quickly redirected to the repair path within tens of milliseconds.

## REFERENCES IN THIS CHAPTER

Cisco. Cisco IOS Software Configuration Guides, <http://www.cisco.com>

Cisco. Cisco IOS XR Software Configuration Guides, <http://www.cisco.com>

Böhmer, Oliver. “Deploying BGP Fast Convergence / BGP PIC” (presented at Cisco Live, London, 2013).

Luc De Ghein. “IP LFA (Loop-Free Alternative): Architecture and Troubleshooting” (presented at Cisco Live, Milan, 2014).

Pete Lumbis. “Routed Fast Convergence” (presented at Cisco Live, San Francisco, 2014).

White, Slice, Retana, *Optimal Routing Design*. Indianapolis: Cisco Press, 2005.

Moy, J., P. Pillay-Esnault, and A. Lindem. RFC 3623, *Graceful OSPF Restart*. IETF, <http://www.ietf.org/rfc/rfc3623.txt>, November 2003

Nguyen, L., A. Roy, and A. Zinin. RFC 4811, *OSPF Out-of-Band Link State Database (LSDB) Resynchronization*. IETF, <http://www.ietf.org/rfc/rfc4811.txt>, March 2007

Nguyen, L., A. Roy, and A. Zinin. RFC 4812, *OSPF Restart Signaling*. IETF, <http://www.ietf.org/rfc/rfc4812.txt>, March 2007

Zinin, A., A. Roy, L. Nguyen, B. Friedman, and D. Yeung. RFC 5613, *OSPF Link-Local Signaling*. IETF, <http://www.ietf.org/rfc/rfc4812.txt>, August 2009

Shand, M. and L. Ginsberg. RFC 3847, *Restart Signaling for Intermediate System to Intermediate System (IS-IS)*. IETF, <http://www.ietf.org/rfc/rfc3847.txt>, July 2004

Sangli, S., E. Chen, R. Fernando, J. Scudder, and Y. Rekhter. RFC 4724, *Graceful Restart Mechanism for BGP*. IETF, <http://www.ietf.org/rfc/rfc4724.txt>, January 2007

Katz, D. and D. Ward. RFC 5880, *Bidirectional Forwarding Detection (BFD)*. IETF, <http://www.ietf.org/rfc/rfc5881.txt>, June 2010

Katz, D. and D. Ward. RFC 5881, *Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)*. IETF, <http://www.ietf.org/rfc/rfc5881.txt>, June 2010

Katz, D. and D. Ward. RFC 5882, *Generic Application of Bidirectional Forwarding Detection (BFD)*. IETF, <http://www.ietf.org/rfc/rfc5882.txt>, June 2010

Katz, D. and D. Ward. RFC 5883, *Bidirectional Forwarding Detection (BFD) for Multihop Paths*. IETF, <http://www.ietf.org/rfc/rfc5883.txt>, June 2010

RFC 2328, *OSPF Version 2*, J. Moy, IETF, <http://www.ietf.org/rfc/rfc2328.txt>, April 1998

Rao, S., A. Zinin, and A. Roy. RFC 3883, *Detecting Inactive Neighbors over OSPF Demand Circuits (DC)*. IETF, <http://www.ietf.org/rfc/rfc3883.txt>, October 2004

Atlas, A. and A. Zinin. RFC 5286, *Basic Specification for IP Fast Reroute: Loop-Free Alternates*. IETF, <http://www.ietf.org/rfc/rfc5286.txt>, September 2008

Filsfil, C. and P. Francois. RFC 6571, *Loop-Free Alternate (LFA) Applicability in Service Provider (SP) Networks*. IETF, <http://www.ietf.org/rfc/rfc6571.txt>, June 2012

# Appendixes

## Appendix A. Decimal to Hex to Binary Tables

This appendix provides a reference conversion chart between decimal, hexadecimal, and binary for all numbers between 1 and 255. The chart acts as a reference when converting between any of the values.

Decimal	Hex	Binary
1	01	00000001
2	02	00000010
3	03	00000011
4	04	00000100
5	05	00000101
6	06	00000110
7	07	00000111
8	08	00001000
9	09	00001001
10	0A	00001010
11	0B	00001011
12	0C	00001100
13	0D	00001101
14	0E	00001110
15	0F	00001111
16	10	00010000
17	11	00010001

18	12	00010010
19	13	00010011
20	14	00010100
21	15	00010101
22	16	00010110
23	17	00010111
24	18	00011000
25	19	00011001
26	1A	00011010
27	1B	00011011
28	1C	00011100
29	1D	00011101
30	1E	00011110
31	1F	00011111
32	20	00100000
33	21	00100001
34	22	00100010
35	23	00100011
36	24	00100100
37	25	00100101
38	26	00100110
39	27	00100111



40	28	00101000
41	29	00101001
42	2A	00101010
43	2B	00101011
44	2C	00101100
45	2D	00101101
46	2E	00101110
47	2F	00101111
48	30	00110000
49	31	00110001
50	32	00110010
51	33	00110011
52	34	00110100
53	35	00110101
54	36	00110110
55	37	00110111
56	38	00111000
57	39	00111001
58	3A	00111010
59	3B	00111011
60	3C	00111100

61	3D	00111101
62	3E	00111110
63	3F	00111111
64	40	01000000
65	41	01000001
66	42	01000010
67	43	01000011
68	44	01000100
69	45	01000101
70	46	01000110
71	47	01000111
72	48	01001000
73	49	01001001
74	4A	01001010
75	4B	01001011
76	4C	01001100
77	4D	01001101
78	4E	01001110
79	4F	01001111
80	50	01010000
81	51	01010001
82	52	01010010
83	53	01010011
84	54	01010100
85	55	01010101
86	56	01010110
87	57	01010111
88	58	01011000
89	59	01011001
90	5A	01011010
91	5B	01011011
92	5C	01011100
93	5D	01011101
94	5E	01011110
95	5F	01011111
96	60	01100000
97	61	01100001
98	62	01100010
99	63	01100011
100	64	01100100

101	65	01100101
102	66	01100110
103	67	01100111
104	68	01101000
105	69	01101001
106	6A	01101010
107	6B	01101011
108	6C	01101100
109	6D	01101101
110	6E	01101110
111	6F	01101111
112	70	01110000
113	71	01110001
114	72	01110010
115	73	01110011
116	74	01110100
117	75	01110101
118	76	01110110
119	77	01110111
120	78	01111000
121	79	01111001
122	7A	01111010
123	7B	01111011
124	7C	01111100
125	7D	01111101
126	7E	01111110
127	7F	01111111
128	80	10000000
129	81	10000001
130	82	10000010
131	83	10000011
132	84	10000100
133	85	10000101
134	86	10000110
135	87	10000111
136	88	10001000
137	89	10001001
138	8A	10001010
139	8B	10001011
140	8C	10001100
141	8D	10001101
142	8E	10001110

143	8F	10001111
144	90	10010000
145	91	10010001
146	92	10010010
147	93	10010011
148	94	10010100
149	95	10010101
150	96	10010110
151	97	10010111
152	98	10011000
153	99	10011001
154	9A	10011010
155	9B	10011011
156	9C	10011100
157	9D	10011101
158	9E	10011110
159	9F	10011111
160	A0	10100000
161	A1	10100001
162	A2	10100010
163	A3	10100011
164	A4	10100100
165	A5	10100101
166	A6	10100110
167	A7	10100111
168	A8	10101000
169	A9	10101001
170	AA	10101010
171	AB	10101011
172	AC	10101100
173	AD	10101101
174	AE	10101110
175	AF	10101111
176	B0	10110000
177	B1	10110001

178	B2	10110010
179	B3	10110011
180	B4	10110100
181	B5	10110101
182	B6	10110110
183	B7	10110111
184	B8	10111000
185	B9	10111001
186	BA	10111010
187	BB	10111011
188	BC	10111100
189	BD	10111101
190	BE	10111110
191	BF	10111111
192	C0	11000000
193	C1	11000001
194	C2	11000010
195	C3	11000011
196	C4	11000100
197	C5	11000101
198	C6	11000110
199	C7	11000111
200	C8	11001000
201	C9	11001001
202	CA	11001010
203	CB	11001011
204	CC	11001100
205	CD	11001101
206	CE	11001110
207	CF	11001111
208	D0	11010000
209	D1	11010001
210	D2	11010010
211	D3	11010011
212	D4	11010100
213	D5	11010101
214	D6	11010110
215	D7	11010111
216	D8	11011000
217	D9	11011001
218	DA	11011010
219	DB	11011011

220	DC	11011100
221	DD	11011101
222	DE	11011110
223	DF	11011111
224	E0	11100000
225	E1	11100001
226	E2	11100010
227	E3	11100011
228	E4	11100100
229	E5	11100101
230	E6	11100110
231	E7	11100111
232	E8	11101000
233	E9	11101001
234	EA	11101010
235	EB	11101011
236	EC	11101100
237	ED	11101101
238	EE	11101110
239	EF	11101111
240	F0	11110000
241	F1	11110001
242	F2	11110010
243	F3	11110011
244	F4	11110100
245	F5	11110101
246	F6	11110110
247	F7	11110111
248	F8	11111000
249	F9	11111001
250	FA	11111010
251	FB	11111011
252	FC	11111100
253	FD	11111101
254	FE	11111110
255	FF	11111111

## Appendix B. BGP Attributes

BGP attaches path attributes (PAs) associated with each network path. The path attributes provide Border Gateway Protocol (BGP) with granularity and control of routing policies within BGP. The BGP prefix attributes are classified as follows:

- Well-known mandatory
- Well-known discretionary
- Optional transitive
- Optional nontransitive

Per RFC 4271, well-known attributes must be recognized by all BGP implementations. Well-known mandatory attributes must be included with every prefix advertisement, whereas well-known discretionary attributes may or may not be included with the prefix advertisement. Optional attributes do not have to be recognized by all BGP implementations. Optional attributes can be set so that they are *transitive* and stay with the route advertisement from autonomous system to autonomous system. Other PAs are *nontransitive* and cannot be shared from autonomous system to autonomous system.

The following table documents the most common BGP attributes in use today.

Attribute	Attribute Code	Attribute Type	Advertised to EGP Peers	Advertised to IBGP Peers	Function
Origin	1	Well-known mandatory	Yes	Yes	Defines the route's source of origin for the prefix advertisement. There are three options: IGP: The route is interior to the originating autonomous system. EGP: The route is exterior to the originating autonomous system. <b>Incomplete:</b> The route cannot be identified as interior or exterior to the originating autonomous system. IGP is preferred over EGP and EGP is preferred over Incomplete.
AS_Path	2	Well-known mandatory	Yes	Yes	This attribute lists a combination of autonomous systems that the prefix has traveled through. The AS_Path consists of AS_Sequence: Ordered set of autonomous systems a prefix has traveled since its origination AS_Set: Unordered set of combined autonomous systems taken from networks within the aggregation range AS_Confed_Set: Unordered set of member autonomous systems that the prefix has traveled AS_Confed_Sequence: Ordered set of member autonomous system numbers in the local confederation that the prefix has traversed.
Next Hop	3	Well-known mandatory	Yes	Yes	Identifies the IP address used to reach the advertising router of the prefix.



Multi-Exit Discriminator (MED)	4	Well-known mandatory	Yes	Yes	Provides a suggestion to an external autonomous system to which path should be used to reach a network. The lower value is preferred to a higher value.
Local Preference	5	Well-known discretionary	No	Yes	Indicates to routers within an autonomous system the preference that traffic leaves an autonomous system. Typically, local preference is set on edge routers when the prefix is received because the local preference is advertised to other routers within the autonomous system. The higher value is preferred to a lower value.
Atomic Aggregate	6	Well-known discretionary	Yes	Yes	Informs BGP routers that some of the BGP path attributes have been lost due to route aggregation.
Aggregator	7	Optional Transitive	Yes	Yes	Identifies the autonomous system and IP address that created the aggregate route.
Community	8	Optional Transitive	Yes, after configuration	Yes, automatic for IOS XR; IOS requires configuration	Provides a method of grouping or tagging destinations prefixes so that routing policies can be applied by other routers.
Originator ID	9	Optional nontransitive	No	Yes	Identifies the router that initially advertised the route as a loop-prevention mechanism when the route is reflected by a route reflector. Route reflectors add this attribute if the attribute does not already exist and is discarded

					if the router sees itself in the originator ID.
Cluster_List	10	Optional nontransitive	No	Yes	Route reflectors use this attribute to prevent routing loops. Route reflectors will append their cluster ID to the cluster list upon route reflection. In the event that a route reflector receives a route with its cluster ID in the cluster list, the route is discarded.
AS4_Path	17	Optional transitive	Yes	Yes	This attribute maintains the AS_Path information when 4-byte autonomous system numbers (ASNs) need to traverse an older BGP router that does not support 4-byte ASNs. AS4_Path cannot contain AS_Confed_Sequence or AS_Confed_Set.
AS4_Aggregator	18	Optional transitive	Yes	Yes	This attribute maintains the information as the aggregator attribute except that it allows for 4-byte ASNs that need to traverse an older BGP router that does not support 4-byte ASNs.
Accumulated Interior Gateway Protocol (AIGP) Metric	26	Optional nontransitive	Yes, but must be explicitly configured	Yes	AIGP enables the BGP to maintain and calculate a conceptual path metric in environments that use multiple autonomous systems with unique IGP routing domains in each autonomous system. The ability for BGP to make routing decisions based on a path metric is a viable option because all the autonomous systems are under the control of a single domain with consistent routing policies for BGP and IGP protocols.
Weight	Not applicable. Weight is not advertised.	Not applicable. Weight is not advertised.	No	No	Weight is a Cisco-defined attribute that is assigned locally on the router and is not advertised to other routers. The path with the higher weight is preferred. The attribute influences only outbound traffic from a router or autonomous system and should be used when other attributes should not influence the best path for a specific network.

## Code Snippets

```
R1#show version
Cisco IOS Software, c7600rsp72043_rp Software (c7600rsp72043_rp-ADVIPSERVICES-M),
Version 15.2(2)S2, RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2012 by Cisco Systems, Inc.
Compiled Tue 07-Aug-12 09:17 by prod_rel_team
ROM: System Bootstrap, Version 12.2(33r)SRC3, RELEASE SOFTWARE (fc1)
BOOTLDR: Cisco IOS Software, c7600rsp72043_rp Software (c7600rsp72043_rp-ADVIPSER-
VICES-M), Version 15.2(2)S2, RELEASE SOFTWARE (fc1)
F340.10.08-7600-1 uptime is 9 minutes
Uptime for this control processor is 9 minutes
System returned to ROM by power cycle (SP by power on)
System image file is "disk0:c7600rsp72043-advipservices-mz.152-2.S2.bin"
Last reload type: Normal Reload
Last reload reason: power-on
```

R1#show version

Cisco IOS Software, C2900 Software (C2900-UNIVERSALK9-M), Version 15.1(4)M2, RELEASE SOFTWARE (fc1)

Technical Support: <http://www.cisco.com/techsupport>

Copyright (c) 1986-2011 by Cisco Systems, Inc.

Compiled Mon 26-Sep-11 17:37 by prod\_rel\_team

ROM: System Bootstrap, Version 15.0(1r)M1, RELEASE SOFTWARE (fc1)

R1 uptime is 26 weeks, 4 days, 18 hours, 24 minutes

System returned to ROM by reload at 00:01:53 UTC Wed Jun 5 2013

System restarted at 20:01:53 EDT Tue Jun 4 2013

System image file is "flash:c2900-universalk9-mz.SPA.151-4.M2.bin"

Last reload type: Normal Reload

Last reload reason: Reload Command

! Output omitted for brevity

Technology Package License Information for Module:'c2900'

```
-----  
Technology      Technology-package      Technology-package  
                Current      Type                Next reboot  
-----  
ipbase          ipbasek9                Permanent          ipbasek9  
security        securityk9               Permanent          securityk9  
uc              uck9                    Permanent          uck9  
data            datak9                   Permanent          datak9
```

Configuration register is 0x0

```
XR1#show install active summary
```

```
Default Profile:
```

```
SDRs:
```

```
Owner
```

```
Active Packages:
```

```
disk0:asr9k-mini-px-4.3.2
```

```
disk0:asr9k-px-4.3.2.CSCuj99070-1.0.0
```

```
disk0:asr9k-px-4.3.2.CSCuj19861-1.0.0
```

```
disk0:asr9k-fpd-px-4.3.2
```

```
disk0:asr9k-mpis-px-4.3.2
```

```
disk0:asr9k-mgbl-px-4.3.2
```

```
disk0:asr9k-bng-px-4.3.2
```

```
disk0:asr9k-doc-px-4.3.2
```

```
disk0:asr9k-px-4.3.2.CSCuj22149-1.0.0
```

```
RP/0/0/CPU0:PE1#show ospf trace hello
```

```
Traces for OSPF 100 (Tue Jan 15 18:27:33)
```

```
Traces returned/requested/available: 2048/2048/2048
```

```
Trace buffer: hello
```

```
1   Jan 5 17:51:48.651  ospf_rcv_hello: intf Gi0/1/0/0 area.0 from 100.1.1.1  
1.10.10.254  
2   Jan 5 17:51:48.651  ospf_check_hello_events: intf Gi0/1/0/0 area.0 from  
1.10.10.1  
3   Jan 5 17:51:49.066  ospf_send_hello: area.0 intf Gi0/1/0/0 from 1.10.10.1  
4   Jan 5 17:51:54.833  ospf_send_hello: area.0 intf Gi0/1/0/3 from 14.1.1.1  
5   Jan 5 17:51:54.932  ospf_rcv_hello: intf Gi0/1/0/2 area.0 from 2.100.2.2  
12.1.1.2
```

Router>

Router>?

Exec commands:

<1-99>	Session number to resume
clear	Reset functions
connect	Open a terminal connection
credential	load the credential info from file system
crypto	Encryption related commands.
disconnect	Disconnect an existing network connection
dot11	IEEE 802.11 commands
emm	Run a configured Menu System
enable	Turn on privileged commands
ethernet	Ethernet parameters
exit	Exit from the EXEC
help	Description of the interactive help system
ips	Intrusion Prevention System
lat	Open a lat connection
lig	LISP Internet Groper
login	Log in as a particular user
logout	Exit from the EXEC
modemui	Start a modem-like user interface
name-connection	Name an existing network connection
ping	Send echo messages
radius	radius exec commands
release	Release a resource
renew	Renew a resource
set	Set system parameter (not config)
ssh	Open a secure shell client connection
tclquit	Quit Tool Command Language shell
telnet	Open a telnet connection
terminal	Set terminal line parameters
tn3270	Open a tn3270 connection
trm	Trend Registration Module

```
Router>  
Router>enable  
Router#
```



```
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
! Global configuration prompt is shown below
Router(config)#
Router(config)#interface Gi0/0
! Interface configuration submode prompt is shown below
Router(config-if)#router eigrp 100
! Router configuration submode prompt is shown below
Router(config-router)#router bgp 100
! Router configuration submode prompt. Notice the prompt is the same even
! though the routing protocol is different
Router(config-router)#exit
Router(config)#
```

```
Router (config)#hostname IOS-ROUTER
```

```
IOS-ROUTER (config)#
```

---

```
R1(config)#archive
R1(config-archive)#path disk0:
R1(config-archive)#time-period 1440
! Time period of 1440 (1 day) was selected to show changes, you will need to
! identify what is best for your organization.
R1(config-archive)#write memory
R1(config)#hostname IOS-Router1
IOS-Router1(config)#exit
```

```
IOS-Router1#show archive
```

```
The maximum archive configurations allowed is 10.
```

```
There are currently 8 archive configurations saved.
```

```
The next archive file will be named disk0:CONFIG-<timestamp>-8
```

Archive #	Name
1	disk0:CONFIG-Dec-16-21-42-33.356-0
2	disk0:CONFIG-Dec-17-21-43-33.410-1
3	disk0:CONFIG-Dec-18-21-44-33.439-2
4	disk0:CONFIG-Dec-19-21-45-33.466-3
5	disk0:CONFIG-Dec-20-21-46-30.840-4
6	disk0:CONFIG-Dec-21-21-46-33.496-5
7	disk0:CONFIG-Dec-22-21-47-33.548-6
8	disk0:CONFIG-Dec-23-21-47-40.742-7 <- Most Recent

```
IOS-Router1#show archive config differences disk0:CONFIG-Dec-20-21-46-30.840-4
!Contextual Config Diffs:
+hostname R1
-hostname IOS-Router1
line con 0
  -length 0
IOS-Router1#
```

---

```
IOS-Router1#configure replace disk0:PreSaved-Config.txt
This will apply all necessary additions and deletions
to replace the current running configuration with the
contents of the specified configuration file, which is
assumed to be a complete configuration, not a partial
configuration. Enter Y if you are sure you want to proceed. ? [no]: y
Total number of passes: 1
Rollback Done
R1#
*Dec 18 01:19:48.435: Rollback:Acquired Configuration lock.
R1#
```

RP/0/RP0/CPU0:XR1#

RP/0/RP0/CPU0:XR1#admin

RP/0/RP0/CPU0:XR1(admin)#



```
RP/0/RP0/CPU0:XR1#configure terminal
```

```
RP/0/RP0/CPU0:XR1(config)#
```

```
RP/0/0/CPU0:ios#show running-config
Building configuration...
!! IOS XR Configuration
interface GigabitEthernet0/0/0/0
 shutdown
!
interface GigabitEthernet0/0/0/1
 shutdown
!
end
RP/0/0/CPU0:ios#
```

```
RP/0/0/CPU0:ios#conf terminal
RP/0/0/CPU0:ios(config)#hostname XR1
RP/0/0/CPU0:ios(config)#cdp
RP/0/0/CPU0:ios(config)#int gigabitEthernet 0/0/0/0
RP/0/0/CPU0:ios(config-if)#cdp
RP/0/0/CPU0:ios(config-if)#int gigabitEthernet 0/0/0/1
RP/0/0/CPU0:ios(config-if)#cdp
RP/0/0/CPU0:ios(config-if)#show configuration
Building configuration...
!! IOS XR Configuration
hostname XR1
cdp
interface GigabitEthernet0/0/0/0
  cdp
!
interface GigabitEthernet0/0/0/1
!
End
```

```
RP/0/0/CPU0:ios(config-if)#show configuration merge
Building configuration...
!! IOS XR Configuration
!! Last configuration change at Fri Sep 20 15:42:00 2014 by xr
!
hostname XR1
cdp
interface GigabitEthernet0/0/0/0
  cdp
  shutdown
!
interface GigabitEthernet0/0/0/1
  cdp
  shutdown
!
End
```

```
RP/0/0/CPU0:ios(config-if)#commit
```

```
RP/0/0/CPU0:Sep 21 00:26:25.360 : config[66391]: %MGBL-CONFIG-6-DB_COMMIT :  
Configuration committed by user 'JCHAMBR'. Use 'show configuration commit changes  
1000000638' to view the changes.
```

```
RP/0/0/CPU0:XR1(config-if)#
```

RP/0/0/CPU0:XR1#show configuration commit list

No.	Label/ID	User	Line	Client	Time Stamp
1	1000000038	JCHAMBR	vty3:node0_0_CPU0	CLI	Fri Sep 13 11:06:35 2014
2	1000000037	KJOHNS	vty3:node0_0_CPU0	CLI	Fri Sep 13 11:05:33 2014
3	1000000036	BEDGEW	vty3:node0_0_CPU0	CLI	Fri Sep 13 11:00:41 2014
4	1000000035	RAMIROG	vty3:node0_0_CPU0	CLI	Fri Sep 13 10:59:39 2014
5	1000000034	JCHAMBR	vty3:node0_0_CPU0	CLI	Tue Aug 27 15:08:04 2014
6	1000000033	KJOHNS	vty1:node0_0_CPU0	CLI	Tue Jul 16 15:32:27 2014
7	1000000032	RAMIROG	vty3:node0_0_CPU0	CLI	Mon Jul 15 16:22:54 2014
8	1000000031	BEDGEW	vty3:node0_0_CPU0	CLI	Mon Jul 15 16:21:14 2014
9	1000000030	JCHAMBR	vty3:node0_0_CPU0	CLI	Mon Jul 15 16:02:07 2014
10	1000000029	KJOHNS	vty4:node0_0_CPU0	CLI	Mon Jul 8 16:06:58 2014
11	1000000028	RAMIROG	vty4:node0_0_CPU0	CLI	Mon Jul 8 16:04:55 2014
12	1000000027	BEDGEW	vty4:node0_0_CPU0	CLI	Mon Jul 8 15:55:08 2014
13	1000000026	JCHAMBR	vty4:node0_0_CPU0	CLI	Mon Jul 8 15:54:03 2014
14	1000000025	KJOHNS	vty4:node0_0_CPU0	CLI	Mon Jul 8 15:49:53 2014
15	1000000024	FOSS	vty4:node0_0_CPU0	CLI	Mon Jul 8 15:48:49 2014

```
RP/0/0/CPU0:XR1#show configuration commit changes 1000000025
Building configuration...
!! IOS XR Configuration
!
no route-policy RPL-L3-IPv4-IN-BETA
end
```

```
RP/0/0/CPU0:XR1#show configuration commit changes last 3
Building configuration...
!! IOS XR Configuration
no cdp
!
no interface Loopback0
!
no router ospf 1
end
```



```
RP/0/0/CPU0:XR1#configure terminal
```

```
RP/0/0/CPU0:XR1(config)#hostname XR1-ROUTER
```

```
RP/0/0/CPU0:XR1(config)#commit label ROUTERNAMECHANGE
```

```
RP/0/0/CPU0:XR1-ROUTER(config)#show configuration commit list
```

SNo.	Label/ID	User	Line	Client	Time Stamp
1	ROUTERNAME	JCHAMBR	con0_0_CPU0	CLI	Fri Sep 20 16:02:22 2014
2	1000000014	JCHAMBR	con0_0_CPU0	CLI	Fri Sep 20 15:48:35 2014
3	1000000013	JCHAMBR	con0_0_CPU0	CLI	Fri Sep 20 15:48:25 2014

```
! Change of the Hostname
RP/0/0/CPU0:XR1#conf terminal
RP/0/0/CPU0:XR1(config)#hostname XR1-COMMITREPLACE
! Examine the target configuration
RP/0/0/CPU0:XR1(config)#show configuration
Building configuration...
!! IOS XR Configuration
hostname XR1-COMMITREPLACE
end
! Examine the target configuration merged with the running configuration
RP/0/0/CPU0:XR1(config)#show configuration merge
Building configuration...
!! IOS XR Configuration
!! Last configuration change at Fri Sep 20 16:07:55 2014 by xr
!
hostname XR1-COMMITREPLACE
cdp
interface GigabitEthernet0/0/0/0
  cdp
  shutdown
!
interface GigabitEthernet0/0/0/1
  cdp
  shutdown
!
end
```

```
RP/0/0/CPU0:XR1(config)#commit replace
This commit will replace or remove the entire running configuration. This
operation can be service affecting.
Do you wish to proceed? [no]: y
RP/0/0/CPU0:Sep 20 16:08:57.284 : ifmgr[220]: %PKT_INFRA-LINK-3-UPDOWN :
Interface GigabitEthernet0/0/0/1, changed state to Down
RP/0/0/CPU0:XR1-COMMITREPLACE(config)#
RP/0/0/CPU0:Sep 20 16:08:57.284 : ifmgr[220]: %PKT_INFRA-LINK-3-UPDOWN :
Interface GigabitEthernet0/0/0/0, changed state to Down
RP/0/0/CPU0:Sep 20 16:08:57.314 : ifmgr[220]: %PKT_INFRA-LINK-3-UPDOWN :
Interface GigabitEthernet0/0/0/1, changed state to Up
RP/0/0/CPU0:Sep 20 16:08:57.314 : ifmgr[220]: %PKT_INFRA-LINK-3-UPDOWN :
Interface GigabitEthernet0/0/0/0, changed state to Up
! Examine the running-configuration now that the change has been committed.
RP/0/0/CPU0:XR1-COMMITREPLACE(config)#show running-config
Building configuration...
!! IOS XR Configuration
!
hostname XR1-COMMITREPLACE
end
```

```
RP/0/0/CPU0:XR1#conf terminal
RP/0/0/CPU0:XR1(config)#router bgp 100
RP/0/0/CPU0:XR1(config-bgp)#commit
RP/0/0/CPU0:XR1(config-bgp)#router bgp 200
RP/0/0/CPU0:XR1(config-bgp)#commit
% Failed to commit one or more configuration items during a pseudo-atomic
operation. All changes made have been reverted. Please issue 'show configuration
failed' from this session to view the errors
RP/0/0/CPU0:XR1(config-bgp)#show configuration failed
!! SEMANTIC ERRORS: This configuration was rejected by
!! the system due to semantic errors. The individual
!! errors with each failed configuration command can be
!! found below.
router bgp 200
!!% The process 'bpm' rejected the operation but returned no error
!
End
```

RP/0/0/CPU0:XR1-COMMITREPLACE#rollback configuration last 3

Loading Rollback Changes.

Loaded Rollback Changes in 1 sec

Committing..

10 items committed in 2 sec (4)items/sec

Updating.

Updated Commit database in 1 sec

Configuration successfully rolled back 3 commits.

RP/0/0/CPU0:XR1#show configuration commit list

Fri Sep 20 16:37:20.748 UTC

SNo.	Label/ID	User	Line	Client	Time Stamp
1	1000000021	JCHAMBR	con0_0_CPU0	Rollback	Fri Sep 20 16:37:10 2014
2	1000000020	JCHAMBR	con0_0_CPU0	CLI	Fri Sep 20 16:08:57 2014
3	1000000019	JCHAMBR	con0_0_CPU0	CLI	Fri Sep 20 16:07:55 2014
4	ROUTERNAME	JCHAMBR	con0_0_CPU0	CLI	Fri Sep 20 16:07:14 2014

RP/0/0/CPU0:XR1#rollback configuration to ROUTERNAME

Loading Rollback Changes.

Loaded Rollback Changes in 1 sec

Committing.

13 items committed in 1 sec (12)items/sec

Updating.

Updated Commit database in 1 sec

Configuration successfully rolled back to 'ROUTERNAMECHANGE'.

RP/0/0/CPU0:XR1#show configuration commit list

SNo.	Label/ID	User	Line	Client	Time Stamp
1	1000000022	JCHAMBR	con0_0_CPU0	Rollback	Fri Sep 20 16:47:08 2014
2	1000000021	JCHAMBR	con0_0_CPU0	Rollback	Fri Sep 20 16:37:10 2014
3	1000000020	JCHAMBR	con0_0_CPU0	CLI	Fri Sep 20 16:08:57 2014
4	1000000019	JCHAMBR	con0_0_CPU0	CLI	Fri Sep 20 16:07:55 2014

```

RP/0/0/CPU0:XR1(config)#hostname XR1-COMMIT-CONFIRM
RP/0/0/CPU0:XR1(config)#commit confirmed 30
RP/0/0/CPU0:Sep 16 13:46:53.374 : config[66625]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration committed by user 'BEDGEW'. Use 'show configuration commit changes
1000000042' to view the changes.
RP/0/0/CPU0:XR1-COMMIT-CONFIRM(config)#
RP/0/0/CPU0:Sep 16 13:47:24.075 : cfgmgr_trial_confirm[66653]: %MGBL-CONFIG-6-DB_
COMMIT : Configuration committed by user 'BEDGEW'. Use 'show configuration commit
changes
1000000043' to view the changes.
RP/0/0/CPU0:XR1-COMMIT-CONFIRM#show configuration commit list
Mon Sep 16 13:59:44.908 EDT
SNo. Label/ID      User      Line           Client         Time Stamp
----
1    1000000043    BEDGEW    vty3:node0_0_CPU0  Rollback      Mon Sep 16 13:47:23 2014
2    1000000042    BEDGEW    vty3:node0_0_CPU0  CLI           Mon Sep 16 13:46:53 2014

```

```

RP/0/0/CPU0:XR1(config)#hostname XR1-COMMIT-CONFIRM
RP/0/0/CPU0:XR1(config)#commit confirmed 30
RP/0/0/CPU0:Sep 16 13:51:47.414 : config[66850]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration
  committed by user 'BEDGEW'. Use 'show configuration commit changes 1000000044' to
view
  the changes.
RP/0/0/CPU0:XR1-COMMIT-CONFIRM(config)#
RP/0/0/CPU0:XR1-COMMIT-CONFIRM(config)#commit
Mon Sep 16 13:51:55.705 EDT
% Confirming commit for trial session.
RP/0/0/CPU0:XR1-COMMIT-CONFIRM(config)#exit
RP/0/0/CPU0:XR1-COMMIT-CONFIRM#show configuration commit list
SNo. Label/ID      User      Line      Client      Time Stamp
~~~~ ~~~~~~      ~~~~      ~~~~      ~~~~~~      ~~~~~~
1    1000000044    BEDGEW    vty3:node0_0_CPU0    CLI      Mon Sep 16 13:51:47 2014
4    1000000043    BEDGEW    vty3:node0_0_CPU0    Rollback Mon Sep 16 13:47:23 2014
5    1000000042    BEDGEW    vty3:node0_0_CPU0    CLI      Mon Sep 16 13:46:53 2014

```



```
RP/0/0/CPU0:XR1(config)#commit label MULTIPLE confirmed 30
```

```
RP/0/0/CPU0:Sep 20 16:59:44.006 : config[65704]: %MGBL-CONFIG-6-DB_COMMIT :  
Configuration committed by user 'BEDGW'. Use 'show configuration commit changes  
1000000023' to view the changes.
```

```
RP/0/0/CPU0:XR-MULTIPLE(config)#commit
```

```
RP/0/0/CPU0:Sep 20 17:00:18.423 : config[65704]: %MGBL-SYS-5-CONFIG_I : Configured  
from console by BEDGW% Confirming commit for trial session.
```

```
RP/0/0/CPU0:XR1#conf terminal
RP/0/0/CPU0:XR1(config)#load bootflash:PRECONFIG-FILE.txt
Loading.
105 bytes parsed in 1 sec (104)bytes/sec
RP/0/0/CPU0:XR1(config)#show configuration
Building configuration...
!! IOS XR Configuration
hostname XR1-PRECONFIG
cdp
interface GigabitEthernet0/0/0/0
  ipv4 address 10.12.1.1 255.255.255.0
!
!
route-policy PASS-ALL
  pass
end-policy
!
End
```

```
RP/0/0/CPU0:XR1#conf terminal
RP/0/0/CPU0:XR1(config)#router bgp 100
RP/0/0/CPU0:XR1(config-bgp)#neighbor 1.1.1.1
RP/0/0/CPU0:XR1(config-bgp-nbr)#address-family ipv4 unicast
RP/0/0/CPU0:XR1(config-bgp-nbr-af)#pwd
router bgp 100 \n neighbor 1.1.1.1 \n address-family ipv4 unicast
```

```
RP/0/0/CPU0:XR1(config-bgp-nbr-af)#exit
```

```
RP/0/0/CPU0:XR1(config-bgp-nbr)#exit
```

```
RP/0/0/CPU0:XR1(config-bgp)#exit
```

```
RP/0/0/CPU0:XR1(config)#
```

```
RP/0/0/CPU0:XR1(config-bgp-nbr-af)#root
```

```
RP/0/0/CPU0:XR1(config)#
```