AN OVERVIEW OF THE NEW ROUTING ALGORITHM FOR THE ARPANET

John M. McQuillan
Ira Richer
Eric C. Rosen
Bolt Beranek and Newman Inc.
Cambridge, MA

(Originally published in: Proc. Sixth Data Communications Symposium, November, 1979)

Summary*

The original routing algorithm of the ARPANET, in service for over a decade, has recently been removed from the ARPANET and replaced with a new and different algorithm. Although the new algorithm, like the old, is a distributed, adaptive routing algorithm, it is not similar to the old in any other important respect. In the new algorithm, each node maintains a data base describing the delay on each network line. A shortest-path computation is run in each node which explicitly computes the minimum-delay paths (based on the delay entries in the data base) from that node to all other nodes in the network. The average delay on each network line is measured periodically by the nodes attached to the lines. These measured delays are broadcast to all network nodes, so that all nodes use the same data base for performing their shortest-path computations. The new routing algorithm was extensively tested on the ARPANET before being released. This paper describes the algorithm and summarizes the results of these tests.

Introduction

The last decade has seen the design, implementation, and operation of several routing algorithms for distributed networks of computers. The first such algorithm, the original routing algorithm for the ARPANET, has served remarkably well considering how long ago (in the history of packet switching) it was conceived. This paper describes the new routing algorithm we installed recently in the ARPANET. Readers not familiar with our earlier activities may consult [1] for a survey of the ARPANET design decisions, including the previous routing algorithm, readers interested in a survey of routing algorithms for other computer networks and current research in the area may consult [2].

A distributed, adaptive routing scheme typically has a number of separate components, including (1) a measurement process for determining pertinent network characteristics, (2) a protocol for disseminating information about these characteristics, and (3) a calculation to determine how traffic should be routed. A routing "algorithm" or "procedure" is not specified until all these components are defined. In the present paper we discuss these components of the new ARPANET algorithm. We begin with a brief outline of the shortcomings of the original algorithm and then provide a description of the new procedure. The algorithm has undergone extensive testing in the ARPANET under operational conditions; the final section gives a summary of the test results. The present paper is a summary of our conclusions only; for more complete descriptions of our research findings, see our internal reports on this project [3, 4, 5].

Problems with the Original Algorithm

The original ARPANET routing algorithm and the new version both attempt to route packets along paths of least delay. The total path is not determined in advance; rather, each node decides which line to use in forwarding the packet to the next node. In the original approach, each node maintained a table of estimated delay to each other node, and sent its table to all adjacent nodes every 128 as. When node I received the

* This research was sponsored by the Defense Advanced Research Projects Agency under ARPA Order No. 3941, and by the Defense Communications Agency (OoD), Contract No. MDA903-78-C-0129, monitored by DSSW. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either express or implied, of the Defense Advanced Research Projects Agency, the Defense Communications Agency, or the United States Government

table from an adjacent node J. it would first measure the delay from itself to J. (We will shortly discuss the procedure used for measuring the delay.) Then it would compute its delay via J to all other nodes by adding to each entry in J's table its own delay to J. Once a table was received from all adjacent nodes, node I could easily determine which adjacent node would result in the shortest delay to each destination node in the network.

In recent years we began to observe a number of problems with the original ARPANET routing algorithm [6] and have come to the conclusion that a complete redesign was the only way to solve some of them. In particular, we decided that a new algorithm was necessary to solve the following problems:

- Although the exchange of routing tables consumed only a small fraction of line bandwidth, the packets containing the tables were long, and the periodic transmission and processing of such long, high-priority packets can adversely affect the flow of network traffic [7]. Moreover, since the updates contain an entry for each network node, the routing packets would become corresponding larger (or more frequent) as the ARPANET grows to 100 or more nodes, thereby exacerbating the problem.
- The route calculation is performed in a distributed manner, with each node basing its calculation on local information together with calculations made previously at every other node. With such a scheme, it is difficult to ensure that routes used by different nodes are consistent,
- The rate of exchange of routing tables and the distributed nature of the calculations causes a dilemma: the network is too slow in adapting to congestion and to important topology changes, yet it can respond too quickly (and perhaps inaccurately) to minor changes.

The delay measurement Procedure of the old ARPANET routing algorithm is quite simple. Periodically, an IMP counts the number of packets queued for transmission on its lines and adds a constant to these counts; the resulting number is the "length" of the line for purposes of routing. This delay measurement procedure has three serious defects:

1) If two lines have different speeds, or different propagation delays, then the fact that the same number of packets is queued for each line does not imply that packets can expect equal delays over the two lines. Even if two Lines nave the same speed and propagation delay, difference in the sizes of the packets which are queued for each line may cause different delays on the two lines.
2) In the ARPANET, where the queues are constrained to have a (short) maximum length, queue length is a boor indicator of delay. The constraints on queue Length are imposed by the software in order to fairly resolve contention for a limited amount of resources. There are a number of such resources which must be obtained before a packet can even be queued for an output line. If a packet must wait a significant amount of time to get these resources, it may experience a long delay, even though the queue for its output line is quite short.
3) An instantaneous measurement of queue length does not accurately predict average delay because there is a significant real-time fluctuation in queue lengths at any traffic level. Our measurements show that under a high constant offered load the average delay is high, but many individual packets show low delays, and the queue length often falls to zero! This variation may be due to variation in the utilization of the CPU, or to other bottlenecks, the presence of which is not accurately reflected by measuring queue lengths.

These three defects are all reflections of a single point, namely that the length of an output queue is only one of many factors that affect a packet's delay. A measurement procedure that takes into account only one such factor cannot give accurate results.

The new routing algorithm is an improvement over the old one in that it uses fewer network resources, operates on more realistic estimates of network conditions, reacts faster to important network changes, and does not suffer from long-term loops or oscillations.

The New Routing Procedure

The routing procedure we have developed contains several basic components. Each node an the network maintains a data base describing the network topology and the line delays. Using this data base, each node independently calculates the best paths to all other nodes, routing outgoing packets accordingly. Because the traffic in the network can be quite variable, each node periodically measures the delays along its outgoing lines and forwards this information (as a "routing update") to all other nodes. A routing update generated by a particular node contains information only about the delays on the lines emanating from that node. Hence an update packet is quite small, and its size is independent of the number of nodes in the network. An update generated by a particular node travels unchanged to all nodes in the network (not just to the immediate neighbors of the originating node, as an

many other routing algorithms.) Since the updates need not be processed before being forwarded, and since they are small, they propagate very quickly through the network, so that all nodes can update their data bases rapidly and continue to route traffic in a consistent and efficient manner.

Many algorithms have been devised for finding the shortest path through a network. Several of these are based on the concept of computing the entire tree of shortest paths from a given node, the root of the tree. A recent article [8] discusses some of these algorithms and references several survey articles. The algorithm we have implemented is based on an algorithm attributed to Dijkstra [9]; because of its search rule, we call it the shortest-path-first (SPF) algorithm.

The basic SPF algorithm uses a data base describing the network to generate a tree representing the minimum delay paths, from a given root node to every other network node. Figure 1 shows a simplified flow chart of the algorithm. The tree initially consists of just the root node.

The tree is then augmented to contain the node that is closest (in delay) to the root and that is adjacent to a node already on the tree. The process continues by repetition of this last step. LIST denotes a data structure containing nodes that have not yet been placed on the tree but are neighbors of nodes that are on the tree. The tree is built up shortest-paths-first — hence the name of the algorithm. Eventually the furthest node from the root is added to the tree, and the algorithm terminates. We have made important additions to this basic algorithm so that changes in network topology or characteristics require only an incremental calculation rather than a complete re- calculation of all shortest paths.

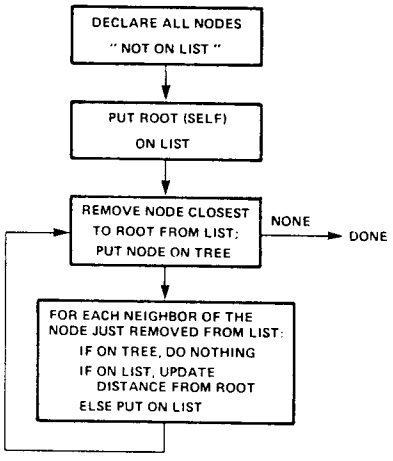


Figure 1

Figure 2 shows a seven-node network and the corresponding shortest path tree for node 1. The figure also shows the routing directory which is produced by the algorithm and which would be used by node 1 to dispatch traffic. For example, traffic for node 4 is routed via node 2. Only the routing directory is used in forwarding packets; the tree is used only in creating the directory.

The two other important components of the routing procedure are the mechanism for measuring delay and the scheme for propagating information. The routing algorithm must have some way of measuring the delay of a packet at each hop. This aspect of the routing algorithm is quite crucial; an algorithm with poor delay measurement facilities will perform poorly, no matter how sophisticated its other features are.

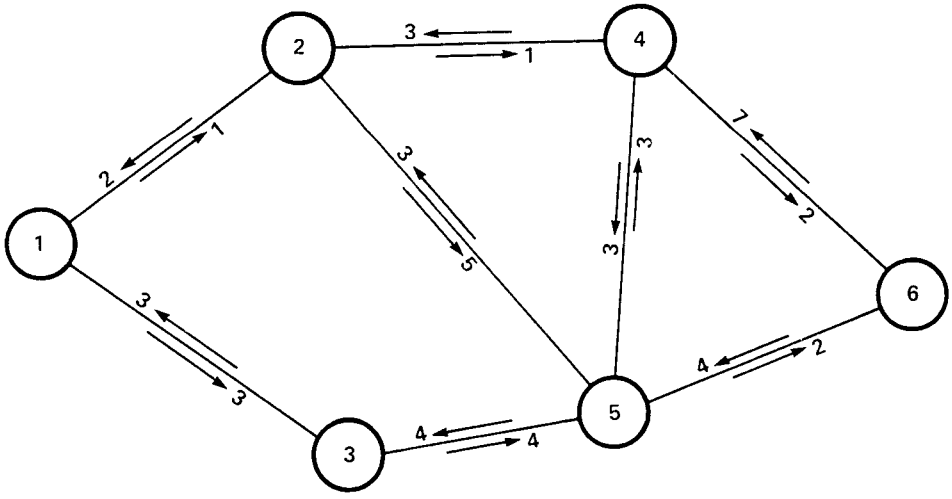
Each node measures the actual delay of each packet flowing over each of its outgoing lines, and calculates the average delay every 10 seconds. If this delay is significantly different from the previous delay, it is reported to all other nodes. The choice of 10 seconds as the measurement period represents a significant departure from the old routing algorithm. Since it takes 10 seconds to produce a measurement, the delay estimate for a given line cannot change more often than once every 10 seconds. The old routing algorithm, on the other hand, would allow the delay estimate to change as often as once every 128 msec. We now believe, however, that there is no point to changing the estimate so often, since it is not possible to obtain an accurate estimate of delay in the ARPANET in less than several seconds. Figure 3 shows some actual delays; an artificially induced traffic load was applied between minutes 5 and 17. The procedure of directly measuring the packet delays cannot fail to yield a more accurate result than any procedure which attempts to infer the delays by measuring something else which is merely expected to correlate with the delays (such as queue lengths.)

The updating procedure for propagating delay information is of critical importance because it must ensure that each update is actually received at all nodes, so that identical data bases of routing information are maintained at all nodes. When an update is generated, it is assigned a sequence number. Each update is transmitted to all nodes by the simple but reliable method of transmitting it on all lines. When a node receives an update, it first checks to see whether it has processed that update (or an update which originated from the same node, but which had a later sequence number) before. If so, it is discarded. If not, it is immediately forwarded to all adjacent nodes. In this way the update quickly flows to all other nodes. The fact that an update flows once in each direction over each network line is the basis for a reliable transmission procedure for the updates. Because the updates are short and are generated infrequently, this procedure uses very little line or node bandwidth. We have augmented this basic procedure with a mechanism to ensure that data bases at nodes are correctly updated when a new node or line is installed, or when a whole set of previously disconnected nodes joins the network. The updating protocol is discussed in detail in [10] .

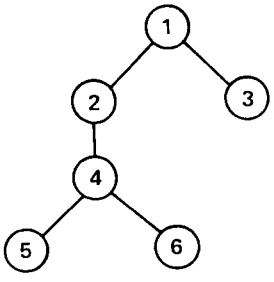
Since all nodes perform the same calculation on an identical data base, there are no long-lasting routing loops. Of course transient loops may occasionally form for a few packets when a change is being processed, but that is quite acceptable, since it has no significant impact on the average delay in the network.

Performance

We next describe some analytical and empirical results on the performance of the new routing algorithm. One important measure of the efficiency of the SPF algorithm is the average time required to process changes in the delays along network lines, since such changes comprise the bulk of the processing requirements. When a given node receives an update message indicating that the delay along some line has increased, the running time of the SPF algorithm is



A) EXAMPLE NETWORK (LINE LENGTHS ARE INDICATED BY THE NUMBERS BESIDE THE ARROW HEADS)



B) SHORTEST PATH TREE

DESTINATION NODE	2	3	4	5	6
ROUTE TRAFFIC VIA NODE	2	3	2	2	2

C) ROUTING DIRECTORY

Figure 2

roughly proportional to the number of nodes in that line's subtree; that is, it is roughly proportional to the number of nodes to which the delay has become worse. When a given node receives an update message indicating that the delay along some line has decreased, the amount of time it takes to run the incremental SPF algorithm is roughly proportional to the number of nodes in that line's subtree after the algorithm is run; that is, it is roughly proportional to the number of nodes to which the delay got better. Thus, in either case, the SPF running time is directly related to the subtree size.

Since the average subtree size provides a measure of SPF performance, it is useful to understand how this quantity varies with the size of the network. Let N denote the number of network nodes, and let h. represent the number of hops on the path from the source node, i = 1, to node i; in other words, if the length of each line is 1, then h_i is the length of the path to node i. Clearly, node i appears in i's subtree and in the subtrees of all the nodes along the path to i. Thus hi is equal to the number of subtrees in which node E is present, so that

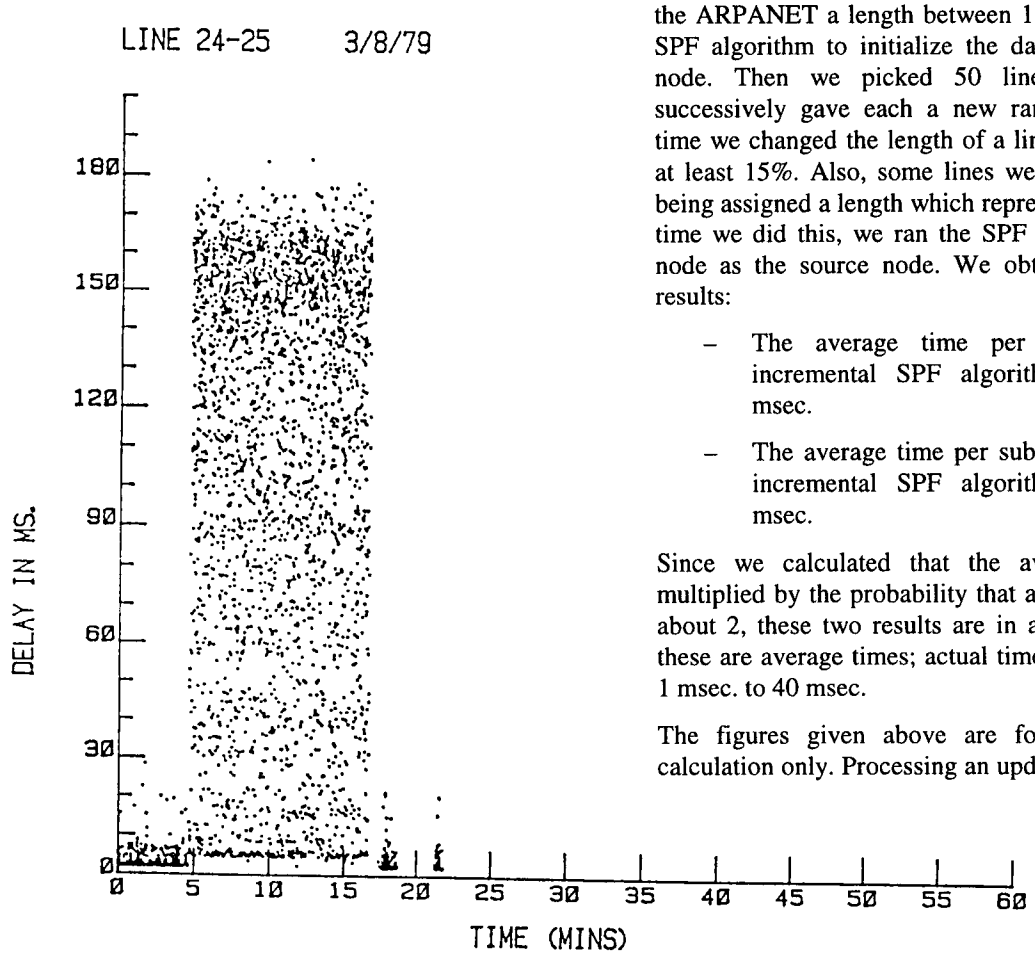


Figure 3

$$\text{total number of all subtree nodes} = \sum_{i=2}^{N} h_i$$

and since there are N-1 subtrees (the complete tree from the source node is not considered to be a "subtree"), the average subtree size is given by

$$\text{average subtree size} = \frac{1}{N-1} \sum_{i=2}^{N} h_i$$

But this expression is identical to the average hop length of all paths, and thus we have the remarkable result that in any tree, the average subtree size is equal to the average hop length from the root to all nodes. This result is significant because the average hop length generally increases quite slowly as the number of nodes increases. (For a network with uniform connectivity c>2, the average hop length increases roughly as log N/log(c-1).)

To establish some estimate of the running time of the algorithm, we programmed a stand-alone version for the ARPANET nodes. We randomly assigned each line in the ARPANET a length between 1 and 20. We ran the SPF algorithm to initialize the data structure in each node. Then we picked 50 lines at random and successively gave each a new random length. Every time we changed the length of a line, we changed it by at least 15%. Also, some lines were brought down by being assigned a length which represented infinity. Each time we did this, we ran the SPF algorithm with each node as the source node. We obtained the following results:

- The average time per node to run the incremental SPF algorithm was about 2.2 msec.
- The average time per subtree node to run the incremental SPF algorithm was about 1.1 msec.

Since we calculated that the average subtree size multiplied by the probability that a line is in the tree is about 2, these two results are in agreement. Note that these are average times; actual times varied from under 1 msec. to 40 msec.

The figures given above are for the shortest path calculation only. Processing an update invokes a routine

to maintain the topology data base (including the ability to dynamically add or delete lines and nodes), and a routine to determine which nodes can be reached from the root node. These modules increase the running time by about a factor of two; and the total storage requirement, including these modules, the topology data base, and the measurement and updating packages, is about 2000 16-bit words.

We designed and programmed the new routing procedure over a period of about six months. We then began an extensive series of tests on the ARPANET, at off-peak hours but under actual network conditions [5]. Our tests involved a great deal more than simply turning the new routing algorithm on to see whether it would run. The tests were specifically designed to stress the algorithm, by inducing those situations which would be most difficult for it to handle well. To stress its ability to react properly to topological changes, we induced line and node failures in as many different ways as we could think of, including multiple simultaneous failures. We also generated large amounts of test traffic in order to see how the algorithm performs under heavy load. (In this respect, it should be noted that the periods during which we were testing were "off-peak" only with respect to the amount of ordinary user traffic in the network. The amount of test traffic we generated far exceeds the amount of traffic generated by users, even during peak hours.) We experimented with many different traffic patterns, in order to test the algorithm under a wide variety of heavily loaded conditions. In particular, we tried to induce those situations which would be most likely to result in loops or in wild oscillations. We also designed and implemented a sophisticated set of measurement and instrumentation tools, so that we could evaluate the routing algorithm's performance. Some of these tools enabled us to monitor the utilization of resources used by the algorithm. Others enabled us to monitor changes in delay (as measured by the routing algorithm), as well as changes in the routing trees themselves at particular network nodes. One of our most important tools was the "tagged packet". A tagged packet is a packet which, as it travels through the network, receives an imprint from each node through which it travels. When such a packet reaches its destination, it contains a list of all the nodes it has traversed, as well as the delay it experienced at each node. These packets provided us with a very straightforward indication of the routing algorithm's performance. Of course, since the network was also in use by ordinary users during our tests, we cannot claim to have performed "controlled" experiments, in the strict scientific sense. However, all our experiments were repeated many times before being used to draw conclusions. Some of our main results are:

1) Utilization of resources (line and processor bandwidth) by the new routing algorithm is as expected, and compares quite favorably with the old algorithm. Line overhead and CPU overhead are both less than two percent.

2) The new algorithm responds quickly and correctly to topological changes; most nodes learn of an update within 100 msec.

3) The new algorithm is capable of detecting congestion, and will route packets around a congested area.

4) The new algorithm tends to route traffic on minimum hop paths, unless there are special circumstances which make other paths more attractive.

5) The new algorithm does not show evidence of serious instability or oscillations due to feedback effects.

6) Routing loops occur only as transients, affecting only packets that are already in transit at the time when there is a routing change. The few packets that we have observed looping have not traversed any node more than twice. However, the loop can be many hops long.

7) Under heavy load, 'he new algorithm will seek out paths where there is excess bandwidth, in order to try to delivery as much traffic as possible to the destination.

Of course, the new routing algorithm does not generate optimal routing — no single-path algorithm with statistical input data could do that. It has performed well, and is successful in eliminating many of the problems associated with the old routing scheme. After several months of careful testing during which both old and new routing algorithms were resident in the network and used for experiments [5], we began to operate the ARPANET with the new routing scheme in May 1979, and removed the old routing program. Since that time we have continued to monitor the performance of the algorithm. the results obtained during our test periods have continued to hold, even during peak hours, and no new or unforeseen problems have yet arisen.

References

1. "The ARPANET Design Decisions," J.M. McQuillan and D.C. Walden, Computer Networks, Vol.1, No.5, August 1977.

2. "Routing Algorithms for Computer Networks -- A Survey", J.M. McQuillan, 1977 National Telecommunications Conference, December 1977.

3. "ARPANET Routing Algorithm Improvements -- First Semiannual Technical Report," J.M. McQuillan, I. Richer, E.C. Rosen, BBN Report No. 3803, April 1978.

4. "ARPANET Routing Algorithm Improvements — Second Semiannual Technical Report, J.M. McQuillen, I. Richer, E.C. Rosen, and D.P. Bertsekas, BBN Report No. 3940, October 1978.

5. "ARPANET Routing Algorithm Improvements — Third Semiannual Technical Report," E.C. Rosen, J. Herman, I. Richer, and J.M. McQuillan, BBN Report No. 4088, April 1979.

6. "ARPANET Routing Study — Final Report," J.M. McQuillan, I. Richer, and E. Rosen, BBN Report No. 3641, September 1977.

7. "On the Effects of Periodic Routing Updates in Packet Switched Networks," W. E. Naylor and L. Kleinrock, Conference Record, National Telecommunications Conference, Dallas, Texas, November, 1976, pp. 16.2.1 - 16.2.7.

8. "Efficient Algorithms for Shortest Paths in Sparse Network," D.B. Johnson, J. ACM, Vol 24, pp 1-13, January 1977.

9. "A Note on Two Problems in Connection with Graphs", E. Dijkstra, Numer. Math., Vol. 1, pp 269-271, 1959.

10. "The Update Protocol of the New ARPANET Routing Algorithm", E.C. Rosen, submitted to Fourth Berkeley Conference on Distributed Data Management and Computer Networks.