Proceedings of the 1983
Winter Simulation Conference
S. Roberts, J. Banks, B. Schmeiser (eds.)

ROUTING TABLE UPDATE EPOCH IN PACKET-SWITCHING NETWORKS*

K. A. Dassel, W. Chou, A. A. Nilsson
P. O. Box 5490
North Carolina State University
Raleigh, N. C. 27650

A computer communication network must be capable of managing its resources efficiently using a routing procedure, flow control, and buffer management techniques. A general network simulator has been developed as a tool in evaluating the many possible combinations of such techniques. It can also be used in evaluating existing strategies in operating networks and investigating new techniques. In this paper the simulator has been used to investigate several strategies of determining an optimum routing table epoch.

1. INTRODUCTION

A computer communication network must be capable of managing its resources efficiently using a routing procedure, flow control, and buffer management techniques. The network routing procedure directs the information, or message, being sent through the network to the correct destination "node." Flow control schemes regulate the acceptance of new messages into the network. The goal is to balance the amount of network traffic congestion and the measure of system performance. Closely related to the routing and flow control strategies is the buffer management. Without a well-designed strategy, buffer overflow and buffer deadlock problems will occur.

Each network must select a routing strategy, flow control scheme, and method of buffer management. A multitude of such combinations exists. TRANSPAC, TELENET, TYMNET, and ARPANET are existing networks. TRANSPAC (Dreyfack 1979, Pouzin 1981, Schwartz 1980) provides a virtual circuit service. The flow of data on each virtual circuit is controlled on each individual logical channel between the source node and the network, or between adjacent nodes (Pouzin 1981). Buffer space at a node is dynamically allocated to the individual virtual circuit queues. Paths through the network for switched virtual circuits are determined by a "call request" packet sent from the source node to the destination node. Routing tables at each node indicate which outgoing line is to be used, dependent on the packet's destination. The routing tables are constructed at a central Network Management Control Center. Link "costs" are defined in terms of the line capacity and number of link buffers. Updates occur whenever a change occurs in a link cost estimate.

TELENET (Mathison 1975, Sun 1982) also provides virtual circuit service. A window scheme is used on both the link and end-to-end level to restrict the flow of data into and within the network. Buffers at a node are grouped into a common pool and shared by the input process, the receive process, and transmission process. No limitation is placed on the maximum number of buffers which can be allocated per link nor per virtual circuit. Routing at the nodes is accomplished using a routing table provided by the Network Control Center. A new table is distributed by the NCC when a topological change occurs in the network.

TYMNET (Price 1977, Rajaraman 1978, Rosen 1980, Schwartz 1977, Tymes 1971, Tymes 1981) transports packets through the network using virtual circuit service. Flow control is managed using a quota system (Tymes 1981). Each channel on a link has an assigned quota of the maximum number of packets it is permitted to send at one time. As part of its buffer management strategy, each port at a node has associated with it a pair of buffers. The routing of the virtual circuit is done when a session request is issued. All routing is performed by the supervisor node. Costs are assigned to each link based on several factors: link bandwidth, type of link, type of transaction, link load, and possibly legal considerations.

*Research supported by Army Research Office under Contract No. DAAG29-81-K-0176.

CH1953-9/83/0000-0219 $01.00 © 1983 IEEE

ARPANET (McQuillan 1977; McQuillan and Walden 1977; McQuillan, Richer, and Rosen 1980; Rosen 1980; Schwartz 1977; Schwartz and Stern 1980; Tanenbaum 1971) routes packets through the network on a datagram basis. Each node maintains its own routing table. All nodes measure the actual delay of each packet sent over each of its outgoing lines (McQuillan, Richer, Rosen 1980), and calculates average delay every 10 seconds. If the delay has changed "significantly" from the previous average, the routing table is updated, and the new delay estimates reported to all other nodes in the network. After each 10-second period, if the difference between the new and old average delay has not exceeded the significance level, the level is decreased by a fixed amount. The significance level is reset to its original level whenever an update is performed. When the level reaches zero, an update is automatically performed. The flow of information into and within the network is regulated using a window scheme. When a source node has a multi-packet message to send, it must first send a request for buffer allocation to the destination node. If the destination node responds with an "allocate" message, the transmission may begin.

Each of these networks has implemented a different combination of routing strategy and resource management. Is one approach superior or is each appropriate for different kinds of network environments? Can performance be improved by modifying any or each choice? What interactions exist between the choice of routing strategy and resource management? What impact does each have on the measure of system performance? As a means of exploring these questions, a general purpose network simulation program was developed. In particular, this paper examines a method of determining an optimum routing table epoch. The approach is compared with three other update strategies.

2. THE SIMULATION PROGRAM MODEL

Typically, when a particular network and its associated protocols are investigated, a simulation program is developed specific to that network. However, if several networks or combinations of routing strategies and resource management techniques are to be studied, generating a different simulation model for each individual network or strategy combination is infeasible and impractical. The model described was developed as a tool for such comparative studies.

The key to the simulator is its highly generalized and structured design. The simulation program contains two types of routines: 1) bookkeeping and data management routines found in most simulation, and 2) routing and resource management dependent routines. Each individual event which contributes to the realization of the network simulation is coded in a separate, self-contained subroutine. The main body of the program consists of calls to subroutines initiating the program, realizing the events, and summarizing the results. The input to the model would include the following:

(1) network characteristics--number of nodes, neighboring nodes, communicating node pairs, line speeds

(2) initial routing table

(3) buffer distribution per node

(4) control information--number of statistics to be collected, program stopping conditions

(5) general information (comparison dependent)--message sizes, buffer limit per output line, conditions determining busy nodes, message interarrival, window size, routing parameters

There are four main events to be simulated:

(1) arrival of a message to the network

(2) movement of the message within the network

(3) servicing the message at each intermediary network node

(4) managing the routing information

Because the subroutines are completely structured (duplicating code if necessary) and self-contained, each appears as a black box to other subroutines. A call is made, a certain response is expected, e.g., the routing table is updated, and the method used to obtain the response is immaterial to the rest of the program. Simulating different routing strategies requires the modification of the routing parameters. Simulating end-to-end flow control schemes affects only events (1) and (3) above. Because the program is highly parameterized it may mean no coding changes are needed; instead, particular variables only need to be re-initialized. Similarly, flow control on the links affects only events (2) and (3). By providing this flexibility, it becomes possible to simulate many network environments and resource management strategies using a single program.

3. STRATEGIES USED IN DETERMINING THE UPDATE EPOCH

To illustrate the utility of this network simulator, determination of an optimum routing table epoch was investigated. The frequency of updating the routing table is a compromise between reporting changes as soon as they are detected and the overhead on the line caused by the routing packets. If the overhead did not exist or was ignored, then it would be reasonable to update the tables whenever any change was detected. Performance would be improved, and any poor routing decisions made during an update would be in effect only a short time. However, overhead does exist and cannot be ignored. The question then is, "When should the routing table be updated?".

The routing strategy can be characterized by two features (Chou, Powell, and Nilsson 1981): 1) metric function used to determine routes and entries in the routing tables, and 2) frequency of updating the routing tables. The metric (Chou, Powell, and Bragg 1979; Chou, Bragg, and Nilsson 1981) is a polynomial function of traffic parameters. When evaluated for a particular

network link, the resulting value is an estimate of the delay or "distance" over the link. The traffic parameter used in this paper is the number of packets queued at each of a node's output lines.

For example, consider a source-destination node pair (a,b) with a minimum of m hops between them and a maximum of eight packets permitted to wait for any output line along the path (Chou, Powell, and Bragg 1979). For packets to be directed onto an alternate path of n hops, the delay on the m-hop path must be "significantly" greater than the delay on the n-hop path. What is significant depends on the values of m and n. If m=n or m+1=n, a small difference in delay estimates indicates that both paths are carrying approximately the same traffic load. When the load is low, any small difference which exists would be expected to last only briefly. By immediately performing an update, upon its completion the situation will probably have reversed itself, necessitating another routing table update before some network nodes will have received routing packets from the first update. When the traffic load is high, the alternate path will also have a high traffic load. Any packets shifted onto this route will encounter highly utilized lines. Thus, if only a small difference existed between the delay estimates, it is expected that packets traveling the alternate route will actually have longer delays. As the difference between m and n increases, the level of significance also needs to increase. For example, if m=2 and n=5, it is desirable that a relatively large difference exist in the delay estimates before sending a packet along a path which will need three extra hops. The more hops the alternative contains, the greater the probability conditions along this path will change before the packet arrives at its destination. A large difference in the delay estimates also indicates that the traffic creating the difference is not brief in duration. In this paper five algorithms for determining the routing table epoch were compared.

3.1 The Absolute Algorithm

Using the network in Figure 1 from Chou, et. al.,

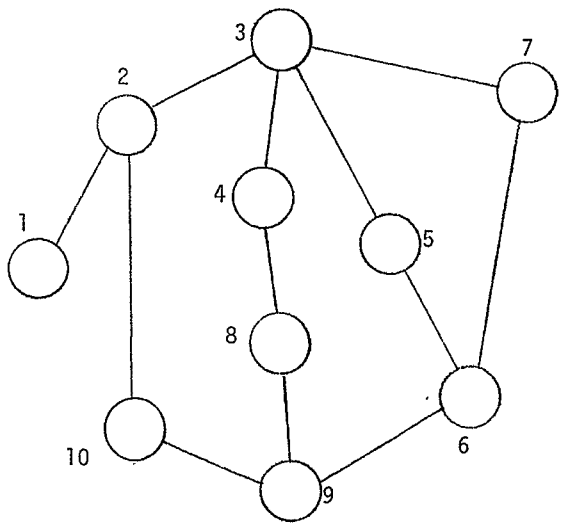

Figure 1: The Network Topology

(1979, 1981), the minimum hop path and second shortest hop path were found for the designated communicating node pairs given in Table 1. For

Table 1: Source-Destination Pairs for the Traffic Environment

Traffic Pair	Source	Destination
1	2	7
2	10	5
3	5	8
4	7	9
5	9	3
6	6	2
7	1	10
8	3	6
9	8	1
10	2	6
11	2	5

example, pair 1 has a minimum hop path between nodes 2 and 7 of 2 hops (from 2 to 3, from 3 to 7). The second shortest path is 4 hops (e.g., from 2 to 10, from 10 to 9, from 9 to 6, from 6 to 7). The average minimum hop path and average second shortest hop path was then calculated. Depending on the message size, the link transmission time is calculated and multiplied by the number of hops on the minimum hop path (if all links in the network were not the same speed, this calculation would need to be modified accordingly). The number of hops corresponding to this path is added to the transmission delay measurement. The process is repeated for the second-shortest path average. For example, for the node pairs in Table 1, the average minimum path was 2.7 hops. The average second shortest path was 3.1 hops. An 800 bit message takes 0.083 seconds to traverse the 9600 bits per second link, or 0.225 seconds to travel the minimum path without processing or queueing delay. To this value is added the hop count, giving 0.225+2.7=2.925 units of delay. Repeating the process for the second-shortest path gives a value of 3.358 units. If the entire procedure were repeated for a 1600 bit message, a value of 3.15 units would be obtained for the minimum hop path and a value of 3.62 units for the second-shortest path. The previous calculations are given in Table 2. These units of measurement will be used to determine a critical value with respect to estimate differences.

The units of delay calculated above differ by ABSDIF=0.433 units. Therefore, if the delay on the minimum delay path has increased by more than 0.4333 units over the next shortest path, a routing change should be made and a routing table update triggered. Periodically (every 250 milliseconds in the simulation program), a node calculates its delay estimates to the other network nodes. As it does, the absolute difference between the old delay estimate and new delay estimate is found, i.e., DIFF=ABS(DELAY_OLD-DELAY_NEW). If DIFF>ABSDIF a flag is set. After

Table 2: Calculations for Traffic Environment

Pair	Source	Destination	Min. No. Hops	Second Shortest
1	2	7	2	4
2	10	5	3	3
3	5	8	3	3
4	7	9	2	4
5	9	3	3	3
6	6	2	3	3
7	1	10	2	2
8	3	6	2	2
9	8	1	4	4
10	2	6	3	3
			Avg. 2.7	Avg. 3.1

Transmission Time/Link, 800 bit message: $\frac{800}{9600} = 0.083$

Transmission Time/Link, 1600 bit message: $\frac{1600}{9600} = 0.167$

Units of Measurement:

800 bit message: 2.7(0.083)+2.7=2.925 units
3.1(0.083)+3.1=3.358 units

1600 bit message: 2.7(0.167)+2.7=3.15 units
3.1(0.167)+3.1=3.62 units

the delay estimates to all other network nodes have been compared, the flag is checked. If the flag is set, a routing table update is performed. If not, the node waits another 250 milliseconds before repeating the process.

This algorithm is similar in its approach to the new ARPANET (McQuillan, Richer, and Rosen 1980) update algorithm. However, rather than using the average delay experienced over the time interval as ARPANET does, this "absolute" algorithm looks at the difference between the actual delay at the time the update decision is to be made, and the previous estimate of the delay. If the absolute value of the difference exceeds the critical value, an update is performed. Whereas the ARPANET algorithm decrements the critical value by a fixed amount each time an update is not performed, the critical value remains constant in the absolute algorithm.

3.2 Modified Absolute Algorithm

Can the absolute algorithm be improved? As it was described, the network-wide critical difference value was determined by the absolute value of the difference between the delay measurement on the average shortest delay path and the delay measurement on the average second-shortest path. The question becomes one of how well the averages are representative of path length in the network. If the network is designed such that alternate paths between source-destination node pairs differ by a relatively constant amount, then using a network critical value is reasonable. However, if this condition is not met, a separate critical value can be calculated for each node pair combination. For example, in Figure 1 the network has been designed in such a way that regardless of how delay estimates fluctuate, node 1 has only one possible output link to choose from; therefore, there is no need to update its corresponding routing table. By using individual critical values, this can be achieved by setting node 1's critical values to a large positive value. However, when alternate paths are the same length, rather than use a critical value of zero, a network-wide critical value is substituted. A critical value of zero is too sensitive to random traffic fluctuation. At any instant delay differences are expected to exist on alternate paths. Using a critical value of zero would trigger an update each time the delay estimates are checked. It was in an attempt to avoid such frequent updating that the absolute algorithm was developed.

3.3 Deterministic Update Algorithm

This algorithm generates an update to the routing table on a fixed interval basis. Regardless of changes in packet delay, the routing table is updated at the end of every epoch. The epoch used in this paper is 250 milliseconds.

3.4 "Buffer" Update Algorithm

Periodically (every 250 msec. in this paper), a node compared the queue size at each output link with the measurements taken at the last routing table update. If the queue size had increased or decreased by at least two packets at any queue, a routing table update was performed.

3.5 Factor Algorithm

This algorithm used the delay measurements calculated in Table 2. Recall that an 800 bit message would experience an average of 2.925 units of delay on the minimum hop path, and 3.358 units of delay on the next shortest path. The average delay on the second-shortest path is a factor of FACTOR=3.358/2.925=1.15 times greater than the average delay on the minimum hop path. Periodically, a node calculates its delay estimates to the other network nodes. As it does, the minimum of the old delay and new delay estimate is found, i.e., MIN=MIN(DELAY_OLD,DELAY_NEW). If (MIN*FACTOR)<MAX(DELAY_OLD,DELAY_NEW), then a flag is set. After the delay estimates to all other network nodes have been compared, the flag is checked. If the flag is set, a routing table update is performed.

4. SIMULATION RESULTS

Using the simulation program, the "absolute" algorithm using both a network-wide critical value and individual critical values was compared with the three other algorithms in determining the update epoch for the network in Figure 1. Effectiveness of the algorithms was measured in packet response time as a function of throughput. The results of the comparison are shown graphically in Figures 2 and 3. For both an 800-bit message and a 1600-bit message, either of the two versions of the "absolute" algorithm provided

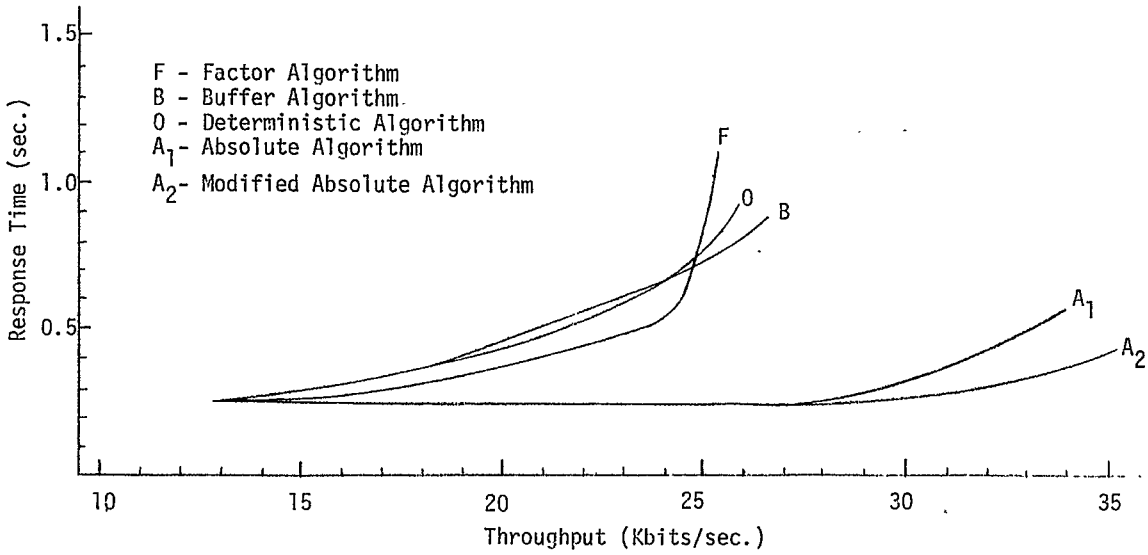

Figure 2: Comparison of Routing Table Update Algorithms in the Network of Figure 1 With a Message Size of 800 Bits

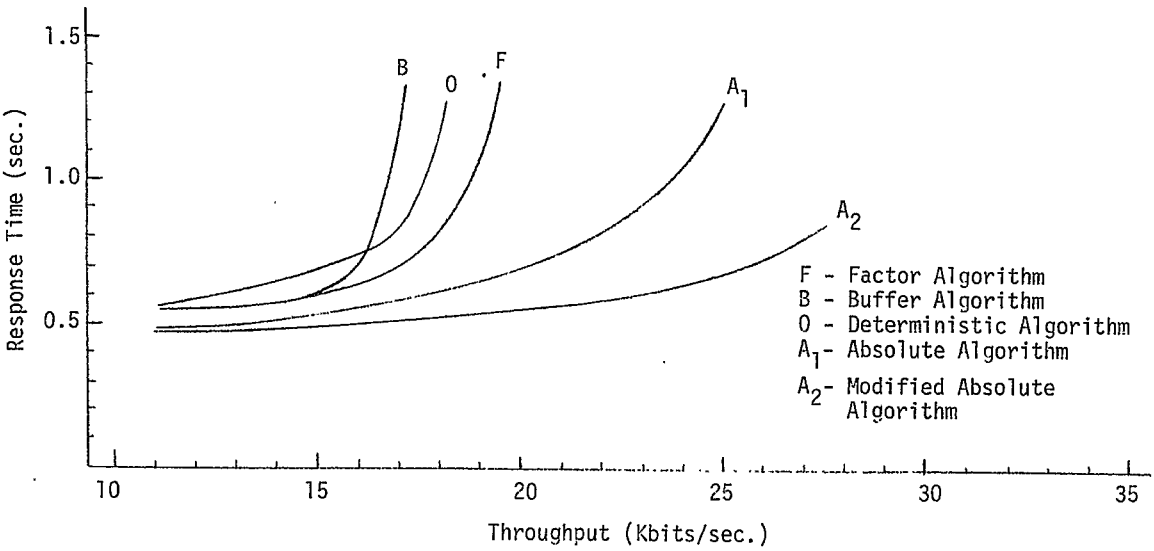

Figure 3: Comparison of Routing Table Update Algorithms in the Network of Figure 1 With a Message Size of 1600 Bits

better service, i.e., response time as a function of throughput, than did the other update algorithms. In addition, for this network, using individual critical values in the "absolute" algorithm provided better service than using a network average. The experiment was repeated on the 10-node network used by Price (1977) in his data network simulations (Figure 4), and on a 15-node modification of this network (Figure 5). In these runs three of the update strategies were simulated: the deterministic update algorithm, the factor algorithm, and the "absolute" algorithm. The "absolute" algorithm used a network-wide critical value. The results are shown graphically in Figures 6 and 7. Again, the "absolute" algorithm provided better performance. For each network simulated, operating under a low level of throughput, all the update strategies performed approximately equally well. However, as the throughput level of the network increased, the superiority of the "absolute" algorithm became evident. For each network, the traffic environment remained stable significantly longer using the "absolute" algorithm to determine the routing table update epoch.

5. SUMMARY

In order for any communication to take place between nodes in a computer network, the information being sent must be able to find its way through the network to the correct destination node. This is accomplished by the routing strategy. In addition to transporting information between a source and destination node, the network

must be capable of managing its resources efficiently. Flow control schemes and buffer management have this responsibility. A multitude of routing-flow control-buffer management strategy combinations exist. A necessary tool to evaluate the combinations is simulation. A very general network simulator has been developed to aid such comparative studies. In this paper, the simulation program was used in comparing several routing table update strategies to determine an optimum update epoch. It was shown that a strategy using absolute differences in packet delay estimates to trigger routing table updates provided the best service. Service was measured by packet response time as a function of throughput.

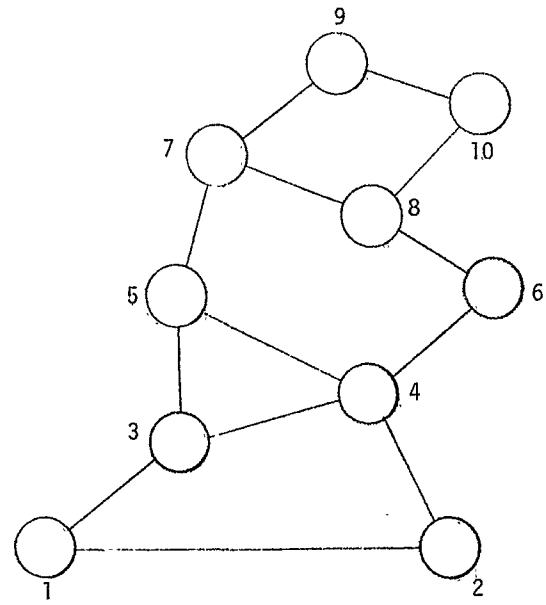

Figure 4: Network Example 2

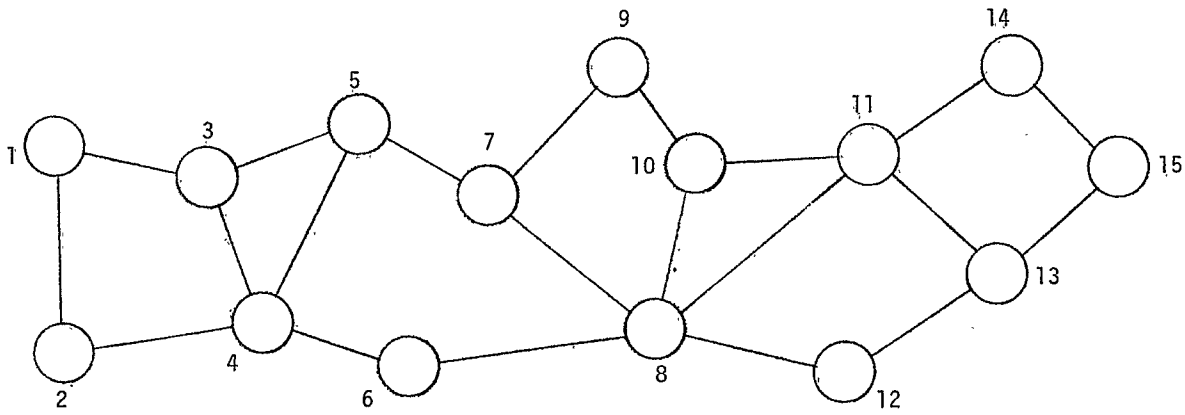

Figure 5: 15-Node Network Example

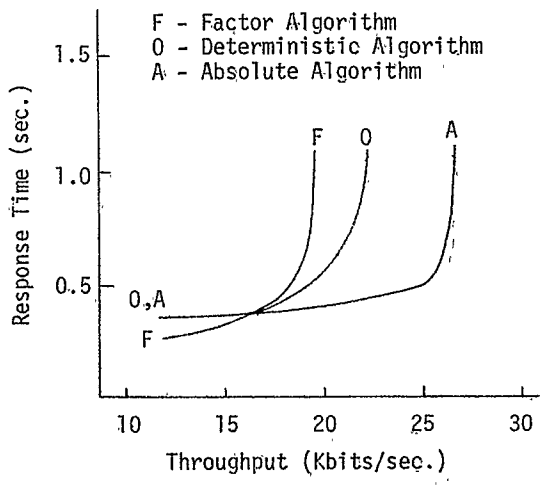

Figure 6: Comparison of Routing Table Update Algorithms in the Network of Figure 4 With a Message Size of 800 Bits

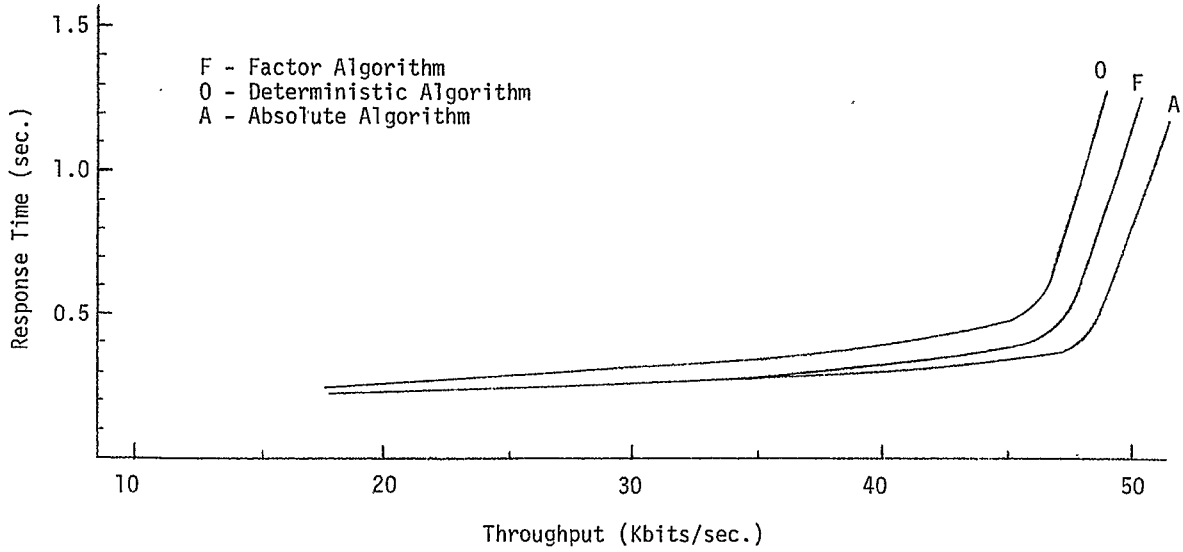

Figure 7: Comparison of Routing Table Update Algorithms in the Network of Figure 5 With a Message Size of 1600 Bits

REFERENCES

Chou W, Powell JD, Bragg AW (1979), Comparative evaluation of deterministic and adaptive routing. In: Proceedings of Flow Control in Computer Networks, pp. 257-279.

Chou W, Bragg AW, Nilsson AA (1981), The need for adaptive routing in the chaotic and unbalanced traffic environment, IEEE Transactions on Communications, Vol. COM-29, No. 4, pp. 481-490.

Dreyfack H (1979), TRANSPAC starts up, wants small users too, Electronics, April 12, 52: 70-72.

Mathison SL (1975), TELENET inaugurates service, Computer Communication Review, Oct., Vol. 5, No. 4, pp. 24-28.

McQuillan JM (1977), Routing algorithms for computer networks--a survey. In: Proceedings 1977 National Telecommunications Conference, December, pp. 28:1-1 - 28:1-6.

McQuillan JM, Walden DC (1977), The ARPA network design decisions, Computer Networks, August, Vol. 1, pp. 243-289.

McQuillan JM, Richer I, Rosen EC (1980), The new routing algorithm for the ARPANET, IEEE Transactions on Communications, May, Vol. COM-28, No. 5, pp. 711-719.

Pouzin L (1981), Methods, tools and observations on flow control in packet-switched data networks, IEEE Transactions on Communications, April, Vol. COM-29, No. 4, pp. 413-426.

Price WL (1977), Data network simulation, Experiments at the National Physical Laboratory 1968-76, Computer Networks, May, Vol. 1, No. 4, pp. 199-210.

Rajaraman A (1978), Routing in TYMNET. Presented at the European Computing Conference, London, England, May.

Rinde J (1977), Routing and control in a centrally-directed network. In: 1977 National Computer Conference, AFIPS Conference Proceedings, Vol. 46, pp. 603-608.

Rosen EC (1980), The updating protocol of ARPANET's new routing algorithm, Computer Networks, No. 4, pp. 11-19.

Schwartz M (1977), Computer communication network design and analysis, Prentice-Hall, Inc.

Schwartz M, Stern TE (1980), Routing techniques used in computer communication networks, IEEE Transactions on Communications, April, Vol. COM-28, No. 4, pp. 539-552.

Sun MK (1982), Personal communication.

Tanenbaum AS (1971), Computer networks, Prentice-Hall, Inc.

Tymes L (1971), TYMNET-a terminal oriented communication network. In: 1971 Spring Joint Computer Conference, AFIPS Conference Proceedings, Vol. 38, pp. 211-216.

Tymes L (1981), Routing and flow control in TYMNET, IEEE Transactions on Communications, Vol. COM-29, No. 4, pp. 392-398.