Performance Analysis of Distributed Routing Strategies Free of Ping-Pong-Type Looping

KANG G. SHIN, SENIOR MEMBER, IEEE, AND MING-SYAN CHEN

Abstract—**This paper deals with a distributed adaptive routing strategy which is very simple and effective, and is free of a ping-pong-type looping in the presence of network failures. Using the number of time intervals required for a node to recover from a network failure as the measure of network's adaptability, performance of this strategy and the ARPANET's previous routing strategy (APRS) is comparatively analyzed without resorting to simulation. Formulas of the exact number of time intervals required for failure recovery under both strategies are also derived. We show that i) the performance of the strategy is always better than, or at least as good as, that of APRS, and ii) network topology has significant effects on the performance of both strategies.**

We also extend our analytical results to routing strategies which are free of loops with more than two nodes. (Such a loop is called a *multinode loop*.) Finally, numerical examples are presented to demonstrate the utility of our results.

Index Terms—**ARPANET, distributed adaptive routing strategy, high-order routing strategy, local and global information, multinode loops, network delay table, ping-pong-type loops.**

I. Introduction

SINCE routing is a key factor to the performance of packet switching networks, various control strategies for routing have been proposed [1]-[5]. Nonadaptive routing strategies, such as random routing and fixed routing, make no attempt to adjust themselves to changes in network conditions and are found to be too inefficient and unreliable to be useful. Centralized adaptive routing strategies—which use a routing control center to keep information on the entire network and make the routing decisions for individual nodes—also suffer from such drawbacks as the vulnerability to single point failures and the need for a routing control center [3]. On the other hand, distributed adaptive routing strategies distribute network information among all computer nodes and, thus, avoid the disadvantages of the other strategies [6]. Hence, we shall focus only on distributed adaptive routing in this paper.

To ensure each node will have sufficient information for routing decisions, we must equip each node with enough information about the entire network. Obviously, it will be very costly for each node to keep and update complete information of the entire network, although, in that case, routing decisions should always be correct. Instead, we would like to incorporate minimal information into each node to make correct or almost correct routing decisions. In other words, each node keeps only a piece of information (i.e., *local information*) of the network rather than information on the entire network.

Furthermore, the network's adaptability should be taken into account when a distributed routing strategy is utilized. When the local information kept in a node is changed, e.g., some links connected to it become faulty, the node must be able to inform the change to all of its neighbors and let their local information be updated to include this change.

Among all distributed adaptive routing strategies reported in the literature, the ARPANET's previous routing strategy (APRS) appears to be acceptable for most packet switching networks [7]. However, this routing strategy has been replaced by the ARPANET's current routing strategy (ACRS) [8] after being in service for over a decade, due to its poor adaptability to network changes and unnecessary looping in case of network failures, i.e., packets are returned to a node from which they were previously transmitted [8], [9].

The reduction of the looping effects is the subject of this paper. The network's adaptability is improved by incorporating more information in routing messages. Specifically, we shall first consider a routing scheme which turns out to be similar to the one implemented in the TIDAS network [10],[1] and then generalizations thereof. Both our routing strategy and the one in [10] are in principle similar to APRS except that some additional features are included in routing messages to speed up the recovery process in case of a network failure.

Performance analysis of routing strategies is known to be very difficult due to the diversified nature of network structures and is usually carried out via simulation [10]. In this paper, the performance of both ours and APRS is rigorously analyzed without resorting to simulation, and formulas of the exact numbers of time intervals required for failure recovery under both strategies are developed. Also, the effects of network topology on the performance of both strategies are described in theorem form. More importantly, routing strategies free of loops with more than two nodes (called *high-order routing strategies*) are proposed and their performances are analyzed.

The paper is organized as follows. In Section II, we describe both APRS and our strategy whose performance is comparatively analyzed in Section III. In Section IV, we extend our strategy to a more general form which is free of multinode loops. Numerical examples are given to illustrate our analytic results in Section V and the paper concludes with Section VI.

Manuscript received September 19, 1985; revised May 27, 1986. The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109.

This work was supported in part by the Office of Naval Research under Contract N00014-85-K-0122.

IEEE Log Number 8611716.

[1] The existence of the strategy in [10] was brought to the authors' attention by an anonymous referee.

II. Description of the Routing Strategies

For a computer network, let N_i, $1 \le i \le n$, be computer nodes and $L_{i,j}$, $1 \le i, j \le n$, be the communication link from N_i to N_j where n is the total number of nodes in the network. Let SP_{i_0,i_m} be the set of all paths from N_{i_0} to N_{i_m} in the network. A path $P_i \in SP_{i_0,i_m}$ is expressed by an *ordered* sequence representation $(N_{i_0}, N_{i_1}, N_{i_2}, \cdots, N_{i_m})$, $1 \le i_j \le n$ for all $j \in \{0, 1, 2, \cdots, m\}$, and nodes on the path are visited in that order. The relation $L_{i_j,i_{j+1}} \in P_i$ means that the link $L_{i_j,i_{j+1}}$ is a part of the path P_i. Let A_i denote the set of all nodes adjacent to N_i, i.e., $\forall N_j \in A_i$, there exists a communication link $L_{i,j}$.

Under APRS, the path from one node to every other node is not determined in advance. Instead, every node maintains a *network delay table* to record the shortest delay via each link emanating from the node. A *minimal delay table* in a node — which contains the delays of the optimal paths (i.e., the path requiring the minimal delay) from that node to all other nodes—is passed to all of its adjacent nodes as a routing message every fixed time interval (i.e., 128 ms in APRS).

Let $D^A_{i/j,d}(m)$ denote the delay from N_i via N_j to N_d in the network delay table of N_i under APRS during the time interval $[m, m + 1)$ and $DL_{i,j}(m)$ be the delay of the link $L_{i,j}$ at time m. Also, let $OP^A_{j,d}(m)$ be the minimum delay path from N_j to N_d in the network delay table of N_j during the interval $[m, m + 1)$ and denote the delay of $OP^A_{j,d}(m)$ by $DOP^A_{j,d}(m)$. According to the operations in APRS, relationships among these quantities can be expressed as follows.

$$D^A_{i\setminus j,d}(m) = DL_{i,j}(m) + DOP^A_{j,d}(m-1) \forall N_j \in A_i \tag{1}$$

$$DOP^A_{i,d}(m) = \min_{N_j \in A_i} \{D^A_{i\setminus j,d}(m)\}. \tag{2}$$

On the other hand, under our strategy as well as the one in the TIDAS network, every node keeps a network delay table and exchanges minimal delay tables with all its adjacent nodes as in APRS except for the following modification. If the routing message is passed from N_j to N_i which is the second node in the optimal path from N_j to some destination node N_d, we replace the delay of the optimal path from N_j to N_d with the delay of its *second optimal path* (i.e., the path requiring the second shortest delay to N_d among all paths in the network delay table of N_j) in the routing message passed to N_i.

For convenience, the APRS and our strategy will henceforth be referred to as strategy A and strategy B, respectively. Let $SOP^B_{j,d}(m)$, $DSOP^B_{j,d}(m)$, $D^B_{i\setminus j,d}(m)$, $OP^B_{j,d}(m)$ and $SOP^B_{j,d}(m)$ be defined to have the same meanings as $SOP^A_{j,d}(m)$, $DSOP^A_{j,d}(m) D^A_{i\setminus j,d}(m)$, $OP^A_{j,d}(m)$ and $SOP^A_{j,d}(m)$, respectively, except that they are the notation under strategy B. Also, let 2nd (P_i) denote the second node on the path P_i. The operations in strategy B can then be expressed in an algorithm form as follows:

```
For every node $N_j \in A_i$
if $N_i = \text{2nd}(OP^B_{j,d}(m-1))$ then $D^B_{i\setminus j,d}(m) = DL_{i,j}(m) + DSOP^B_{j,d}(m-1)$
                                      else $D^B_{i\setminus j,d}(m) = DL_{i,j}(m) + DOP^B_{j,d}(m-1)$
```

where

$$DOP^B_{i,d}(m) = \min_{N_j \in A_i} \{D^B_{i\setminus j,d}(m)\} \tag{3}$$

$$DSOP^B_{i,d}(m) = \min_{\substack{N_j \in A_i \\ N_j \ne \text{2nd}(OP^B_{i,d}(m))}} \{D^B_{i\setminus j,d}(m)\}. \tag{4}$$

Performance of both strategies A and B will be comparatively analyzed first in the following section. In Section IV, the analytic results for strategy B will then be extended to the more general case: routing strategies which are free of multinode loops.

III. Comparative Analysis of Network Performance

We begin with an introduction of necessary assumptions and definitions in Section III-A, which is then followed by the performance analysis of strategies A and B in Section III-B. For both strategies the network's adaptability is measured by the number of time intervals required for a node to recover from a network failure.

A. Assumptions and Definitions

Since node failures can be represented as the failure of all the links attached to it, without loss of generality, only link failures will be considered. Assume that the faulty link L_{f,f^*} is a part[2] of the optimal path from N_s to N_d, and the link failure is detected at time 0 while the delays of other links remain unchanged. We assume further that there always exists at least one nonfaulty path from N_s to N_d even after a link failure. In what follows, m_A and m_B are used to denote the numbers of time intervals required for N_s to obtain a new nonfaulty optimal path to N_d in the presence of a link failure under strategy A and strategy B, respectively.

Also needed are the following definitions.

Definition 1: A path P_i is said to be *faulty* iff the path contains a faulty link L_{f,f^*}, i.e., $L_{f,f^*} \in P_i$.

Definition 2: A *loop*, denoted by L_i, $1 \le i \le n$, is a path which starts and ends at the same node N_i, i.e., $L_i \in SL \equiv \bigcup_{1 \le i \le n} SP_{i,i}$.

Definition 3: A path is said to *contain a loop* iff there is one or more nodes appearing more than once in its ordered sequence representation.

Definition 4: A path is said to *contain a ping-pong-type loop* iff there exists a node N_i in its ordered sequence representation which is sandwiched between two occurrences of another node N_j.

Definition 5: If a path $P_i = (N_{i_0}, N_{i_1}, \cdots, N_{i_m})$ is faulty and the pair $(N_{i_k}, N_{i_{k+1}})$, i.e., $L_{i_k,i_{k+1}}$, represents the first faulty link in the ordered sequence representation of P_i, then k is called the *screen* of P_i and denoted by scn (P_i). If the path P_i is not faulty, then scn $(P_i) = \infty$.

Definition 6: A new set of paths, $SP^*_{s,d}$, is defined as

$$SP^*_{s,d} \equiv SP_{s,d} - \{P_i | P_i \in SP_{s,d} \text{ and } P_i \text{ contains a ping-pong-type loop}\}.$$

Obviously, $SP^*_{s,d} \subseteq SP_{s,d}$.

[2] If the faulty link is not in the optimal path, both strategies will have the same performance.

Definition 7: The *delay function* d: $SP \to R^+$ is defined as sum of delays of all links in a path, where $SP \equiv \cup_{i \le i,j \le n} SP_{i,j}$ and R^+ is the set of positive real numbers.

Definition 8: The *hop function* h: $SP \to I^+$ is defined as the number of links in a path where I^+ is the set of positive integers.

Definition 9: The function A: $SP \to R^+$ represents the *average link delay* in a path, i.e., $A(P_i) = [d(P_i)]/[h(P_i)] \forall P_i \in SP$.

For example, consider a path $P_i = (N_1, N_2, N_1, N_2, N_1, N_3)$ in the example network of Fig. 1. According to the above definitions, $d(P_i) = 12$, $h(P_i) = 5$, $A(P_i) = 2.4$, and P_i contains a ping-pong-type loop occurring between N_1 and N_2. Hence, $P_i \in SP_{1,3}$, but $P_i \notin SP^*_{1,3}$. When the link $L_{1,3}$ is faulty, P_i is faulty and scn $(P_i) = 4$.

B. Analysis of the Two Routing Strategies

Under both strategies, each node gathers routing information and then updates its local delay table by exchanging routing messages with all its neighbors every fixed time interval. From (3) and (4), one can see that strategy B removes all the paths with ping-pong-type loops from network delay tables. In case of a link failure, this fact will enable a node under strategy B to determine a new nonfaulty optimal path faster than under strategy A and, thus, enhance the network's adaptability. More formally, this property is stated as a lemma without proof as follows.

Lemma 1: Strategy B is free of a ping-pong-type looping, i.e., $OP^B_{s,d}(m) \in SP^*_{s,d} \forall m \in I$ where I is the set of integers.

Since the information that the link $L_{f,f^*} = L_{i_k,i_{k+1}}$ of a path has become faulty is broadcast only one hop per unit time under both strategies, it will take the same number of time units as the screen value of that path for the source node to be informed of the link failure. Thus, the screen values of $OP^A_{s,d}(m)$ and $OP^B_{s,d}(m)$ must be greater than m. This can be formally stated as a lemma below.

Lemma 2: The following inequalities hold $\forall m \in I^+$:

$$\text{scn } (OP^A_{s,d}(m)) > m$$
$$\text{scn } (OP^B_{s,d}(m)) > m.$$

Generally speaking, the screen value of a path means the number of time intervals required to remove this path from the network delay table of N_s after the occurrence of a failure in the path. Thus, under both strategies, the path with the shortest delay to a destination node N_d chosen by a source node N_s is the shortest among all possible paths which have not been found faulty by N_s up to time m. More formally, we have the following theorem.

Theorem 1:

i) $DOP^A_{s,d}(m) = \min \{d(P_i) | P_i \in SP_{s,d}$ and scn $(P_i) > m\} \forall m \in I^+$,

ii) $DOP^B_{s,d}(m) = \min \{d(P_j) | P_j \in SP^*_{s,d}$ and scn $(P_j) > m\} \forall m \in I^+$.

Proof of i): Recall that m_A time units are required to find a new nonfaulty optimal path under strategy A in case of a link failure. Consider two cases: $m \ge m_A$ and $m < m_A$. When $m \ge m_A$, $OP^A_{s,d}(m)$ is the newly found optimal path and fault-

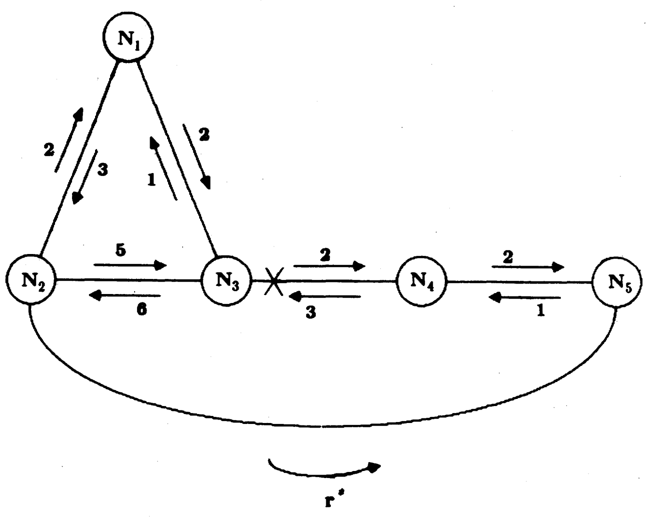


Fig. 1. An example computer network.

free. That is, scn $(OP^A_{s,d}(m)) = \infty$, resulting in

$$DOP^A_{s,d}(m) = \min \{d(P_i) | P_i \in SP_{s,d} \text{ and scn } (P_i) = \infty\}, \text{ thus satisfying i).}$$

When $m < m_A$, $OP^A_{s,d}(m)$ still contains the faulty link L_{f,f^*}. We want to show that $OP^A_{s,d}(m)$ chosen at time m from the local delay table of N_s is indeed the real minimum delay path found by N_s. Let P_{i^*} be an arbitrary path containing the faulty link L_{f,f^*} at time m, $P_{i^*} \in SP_{s,d}$ and scn $(P_{i^*}) > m$. Then, P_{i^*} can be expressed as $(N_{i^*_0}, N_{i^*_1}, \cdots, N_{i^*_r}, N_{i^*_{r+1}}, \cdots, N_{i^*_u})$ where $N_{i^*_0} = N_s$, $N_{i^*_r} = N_f$, $N_{i^*_{r+1}} = N_{f^*}$, $N_{i^*_u} = N_d$, and scn $(P_{i^*}) = r > m$. From (1) and (2) we get

$$DOP^A_{i^*_0,i^*_u}(m) \le DL_{i^*_0,i^*_1}(m) + DOP^A_{i^*_1,i^*_u}(m-1)$$
$$DOP^A_{i^*_1,i^*_u}(m-1) \le DL_{i^*_1,i^*_2}(m-1) + DOP^A_{i^*_2,i^*_u}(m-2)$$
$$\vdots$$
$$DOP^A_{i^*_{r-1},i^*_u}(m-r+1) \le DL_{i^*_{r-1},i^*_r}(m-r+1) + DOP^A_{i^*_r,i^*_u}(m-r)$$
$$\Rightarrow$$
$$DOP^A_{i^*_0,i^*_u}(m) \le \sum_{k=0}^{1} DL_{i^*_k,i^*_{k+1}}(m-k) + DOP^A_{i^*_2,i^*_u}(m-2)$$
$$\vdots$$
$$\le \sum_{k=0}^{r-1} DL_{i^*_k,i^*_{k+1}}(m-k) + DOP^A_{i^*_r,i^*_u}(m-r).$$

Since $r > m$, m is a time before $L_{i^*_r,i^*_{r+1}}$ became faulty and we get

$$DOP^A_{i^*_r,i^*_u}(m-r) = \sum_{k=r}^{u-1} DL_{i^*_k,i^*_{k+1}}(m-k)$$

$$\Rightarrow DOP^A_{s,d}(m) = DOP^A_{i^*_0,i^*_u}(m) \le \sum_{k=0}^{u-1} DL_{i^*_k,i^*_{k+1}} = d(P_{i^*}).$$

Besides, from Lemma 2, scn $(OP^A_{s,d}(m)) > m$ and $OP^A_{s,d}(m) \in SP_{s,d}$, thereby leading to i).

Proof of ii): This part can be proved similarly to part i) by recursively applying (3) and (4) and retrospectively tracing the network delay tables of nodes in an arbitrary path. ■

The following two important results follow from Theorem 1.

Result 1: $DOP^A_{s,d}(m_A) = DOP^B_{s,d}(m_B)$. That is, the nonfaulty optimal paths from N_s to N_d obtained under both strategies have the same delay.

Result 2: $DOP^A_{s,d}(m) \le DOP^A_{s,d}(m+1)$ and $DOP^B_{s,d}(m) \le DOP^B_{s,d}(m+1) \forall m \in I$. In other words, the delay of the optimal path chosen under each strategy is a nondecreasing function of time.

Since at least one new nonfaulty path is assumed to exist in case of link failures, after the occurrence of a failure both $DOP^A_{s,d}(m)$ and $DOP^B_{s,d}(m)$ will converge to the delay of the new nonfaulty optimal path as m increases. Let $r < \infty$ denote the delay of the new nonfaulty optimal path. For a given network, m_A and m_B can be determined from the value of r by the following theorem.

Theorem 2:

i) $m_A = \max \{\text{scn } (P_i) | P_i \in SP_{s,d} \text{ and } d(P_i) < r\}$

ii) $m_B = \max \{\text{scn } (P_j) | P_j \in SP^*_{s,d} \text{ and } d(P_j) < r\}$.

Proof: Consider i) first. Suppose $m^*_A = \max \{\text{scn } (P_i) | P_i \in SP_{s,d} \text{ and } d(P_i) < r\}$ and $m^*_A \ne m_A$. We shall prove that both the assumptions of $m_A > m^*_A$ and $m_A < m^*_A$ lead to contradictions.

Case 1: $m_A > m^*_A$. Since m_A is the minimal number of time intervals required for N_s to obtain the new nonfaulty optimal path to N_d, $DOP^A_{s,d}(m_A - 1) < DOP^A_{s,d}(m_A) = r$. By Theorem 1, scn $(OP^A_{s,d}(m_A - 1)) = m^{**}_A > m_A - 1 \Rightarrow m^{**}_A \ge m_A$. Because $SP_{s,d}$ is the set of all paths from N_s to N_d,

$$m^*_A = \max \{\text{scn } (P_i) | P_i \in SP_{s,d} \text{ and } d(P_i) < r\} \ge m^{**}_A \ge m_A$$

leading to a contradiction to $m^*_A < m_A$.

Case 2: $m_A < m^*_A$. Since $m^*_A = \max \{\text{scn } (P_i) | P_i \in SP_{s,d} \text{ and } d(P_i) < r\}$, there exists a path P_j such that scn $(P_j) = m^*_A$, $P_j \in SP_{s,d}$ and $d(P_j) < r$. From Theorem 1, $DOP^A_{s,d}(m^*_A - 1) \le d(P_j) < r$. For $m_A, m^*_A \in I^+$, $m_A < m^*_A \Rightarrow m_A \le m^*_A - 1$. Thus, we get $DOP^A_{s,d}(m_A) \le DOP^A_{s,d}(m^*_A - 1) \le d(P_j) < r$. However, $r = DOP^A_{s,d}(m_A)$, leading to a contradiction. Since, by Lemma 1, $OP^B_{s,d}(k) \in SP^*_{s,d} \forall k \in I^+$, part ii) can be proved similarly to part i). ■

From Theorem 2, we can draw an important conclusion that while some link in the network becomes faulty, the convergence time required for a node to obtain a new nonfaulty optimal path under strategy B is always less than, or equal to, that required under strategy A. We formally state this as a corollary below.

Corollary 2.1: $m_B \le m_A$.

Proof: $\forall P_i \in SP^*_{s,d} \Rightarrow P_i \in SP_{s,d}$ since $SP^*_{s,d} \subseteq SP_{s,d}$, $\Rightarrow m_B \le m_A$ by Theorem 2. ■

By definition, we can describe the screen and delay functions of a path $P_i = (N_{i_0}, N_{i_1}, \cdots, N_{i_k}, N_{i_{k+1}}, \cdots, N_{i_u})$

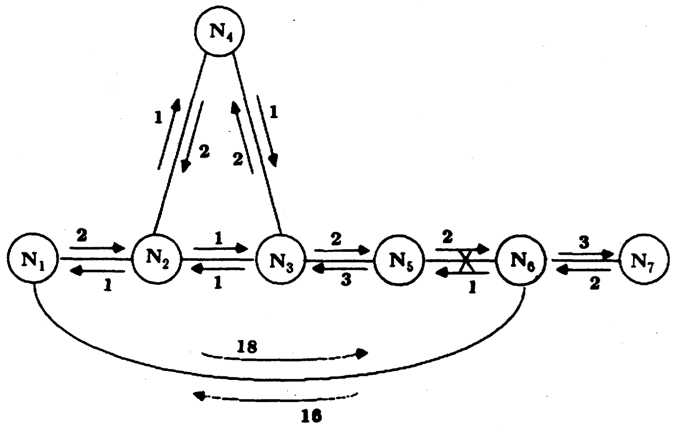


Fig. 2. An example network showing $r = 21$ when $L_{5,6}$ becomes faulty.

$\in SP_{s,d}$ where $N_{i_0} = N_s$, $N_{i_k} = N_f$, $N_{i_{k+1}} = N_{f^*}$, and $N_{i_u} = N_d$ as below.

$$\text{scn } (P_i) = h(P_i') + \sum_{j=1}^{k} n_{i_j} h(L_{i_j}) \tag{5}$$

$$d(P_i) = d(P_i') + \sum_{j=1}^{k} n_{i_j} d(L_{i_j}) + d(P_{i_r}) \tag{6}$$

where $P_i \supset P_{i_r} \in SP_{f,d}$, $P_i' \in SP_{s,f}$, L_{i_j} is a loop starting and ending at N_{i_j} and n_{i_j} the number of times the loop L_{i_j} appears in the path P_i.

Since we can include all redundant parts of P_i in the second term of (5) and the second and third terms of (6), P_i' can be regarded as a path without loops. For the example network of Fig. 2, consider a path $P_i = (N_1, N_2, N_3, N_4, N_2, N_3, N_5, N_3, N_5, N_3, N_5, N_6, N_7)$ in which the link $L_{5,6}$ is faulty. Then $P_i' = (N_1, N_2, N_3, N_5)$, $P_{i_r} = (N_5, N_6, N_7)$, $L_{i_1} = (N_2, N_3, N_4, N_2)$, $n_{i_1} = 1$ and $L_{i_2} = (N_3, N_5, N_3)$, $n_{i_2} = 2$. Using (5) and (6), we can express Theorem 2 as the following optimization problems.

$$m_A: \max_{P_i \in SP_{s,d}} h(P_i') + \sum_{j=1}^{k} n_{i_j} h(L_{i_j}) \tag{7}$$

$$\text{subject to } d(P_i') + \sum_{j=1}^{k} n_{i_j} d(L_{i_j}) + d(P_{i_r}) < r \text{ where } L_{i_j} \in SP_{i_j,i_j} \text{ and } L_{i_j} \subset P_i. \tag{7'}$$

$$m_B: \max_{P_i \in SP^*_{s,d}} h(P_i') + \sum_{j=1}^{k} n_{i_j} h(L_{i_j}) \tag{8}$$

$$\text{subject to } d(P_i') + \sum_{j=1}^{k} n_{i_j} d(L_{i_j}) + d(P_{i_r}) < r \text{ where } L_{i_j} \in SP^*_{i_j,i_j} \text{ and } L_{i_j} \subset P_i. \tag{8'}$$

It is interesting to observe that the n_{i_j}'s satisfying (7) and (7') (or (8) and (8')) can be determined by solving a linear knapsack problem. For example, consider the determination of m_A. Suppose $P_{i^*} \in SP_{s,d}$ is the path which maximizes the expression of (7) subject to the constraint (7'). The knowledge of the path P_{i^*} completely specifies $P_i'^*$, $P_{i_r^*}$, $L_{i_j^*}$, and $n_{i_j^*}$ for

$1 \le j \le k$. Since $SP_{s,d}$ is the set of all paths from N_s to N_d, from (7) and (7′) the vector $\boldsymbol{n} = [n_{i_1^*}, n_{i_2^*}, \cdots, n_{i_k^*}]$ is the solution to a linear integer knapsack programming problem P_0, which can be obtained by applying the algorithm A_0 given below.

$$P_0\colon\ S = \max_{n_i, 1 \le i \le k} \sum_{i=1}^{k} a_i n_i$$

$$\text{subject to } \sum_{i=1}^{k} b_i n_i < M \text{ and } n_i \in I^+$$

$$\text{for } i = 1, \cdots, k$$

where $S = m_A - h(P_i'^*)$, $h(L_{i_j^*}) = a_i$, $d(L_{i_j^*}) = b_i$ and $M = r - d(P_i'^*) - d(P_{i_r^*})$.

Algorithm A_0:

Step 1) Set $m := M$ and $n_i := 0$ for all $i \in \{1, \cdots, k\}$.

Step 2) Select an i^* such that $a_{i^*}/b_{i^*} = \max_{b_i < m} \{a_i/b_i\}$.

Step 3) $n_{i^*} := \lfloor m/b_{i^*} \rfloor$, $m := m - b_{i^*} n_{i^*}$.

Step 4) **If** there exists a $b_i < m$ **then** go to Step 2) **else** stop.

The determination of m_B can be treated similarly and the above result leads to the following lemma.

Lemma 3: $S = \lfloor M/b_{i^*} \rfloor a_{i^*} + R$ where $R < a_{i^*}$ and $a_{i^*}/b_{i^*} = \max_{1 \le i \le k} \{a_i/b_i\}$.

Proof: Let $\boldsymbol{n}^* = [n_1^*, n_2^*, \cdots, n_k^*]$ be a solution of P_0 obtained by applying algorithm A_0.

$$S = \sum_{i=1}^{k} a_i n_i^* = \left\lfloor \frac{M}{b_{i^*}} \right\rfloor a_{i^*} + \sum_{i \ne i^*}^{k} a_i n_i^*$$

$$\cdot \sum_{i=1}^{k} b_i n_i^* < M \Rightarrow \sum_{i \ne i^*}^{k} b_i n_i^* < M - \left\lfloor \frac{M}{b_{i^*}} \right\rfloor b_{i^*} < b_{i^*}$$

$$\Rightarrow \sum_{i \ne i^*}^{k} a_i n_i^* < a_{i^*} \quad \text{because } \frac{a_{i^*}}{b_{i^*}} = \max_{1 \le i \le k} \left\{ \frac{a_i}{b_i} \right\}.$$

Thus, Lemma 3 follows. ■

Define loops L_A and L_B such that $A(L_A) = \min\{A(L_i) | L_i \in SP_{i,i}, 1 \le i \le n\}$, and $A(L_B) = \min\{A(L_i) | L_i \in SP_{i,i}^*, 1 \le i \le n\}$. From (7) and Lemma 3, when the delay of the new nonfaulty optimal path r is large enough, $A(L_A)$ will become the dominating factor in determining m_A. Clearly, what L_A is to m_A is L_B to m_B. This fact leads to Theorem 3 below.

Theorem 3:

$$\lim_{r \to \infty} \frac{m_B}{m_A} = \frac{A(L_A)}{A(L_B)}.$$

Proof: Let L_{i_m} be the loop with the minimal average delay among all loops in the path P_i, i.e., $A(L_{i_m}) = \min \{A(L_j) | L_j \in SL \text{ and } L_j \subseteq P_i\}$. Applying Lemma 3 to (7) and (8), we get

$$m_A = \max_{P_i \in SP_{s,d}} \left\{ \left\lfloor \frac{r - d(P_i') - d(P_{i_r})}{d(L_{i_m})} \right\rfloor \cdot h(L_{i_m}) + R_i + h(P_i') \right\} \quad \text{where } R_i < h(L_{i_m}). \tag{9}$$

$$m_B = \max_{P_i \in SP_{s,d}^*} \left\{ \left\lfloor \frac{r - d(P_i') - d(P_{i_r})}{d(L_{i_m})} \right\rfloor \cdot h(L_{i_m}) + R_i + h(P_i') \right\} \quad \text{where } R_i < h(L_{i_m}). \tag{10}$$

Clearly, the second and third terms in these two equations become negligible as $r \to \infty$. Therefore

$$\lim_{r \to \infty} \frac{m_B}{m_A} = \frac{d(L_A) h(L_B)}{d(L_B) h(L_A)} = \frac{A(L_A)}{A(L_B)}. \quad ■$$

The first and second terms in (9) and (10) are related to the redundant parts of a path, while the third term can be viewed as the distance (measured in hops) to the faulty link. Theorem 3 shows that the improvement via strategy B strongly depends on network topology. This fact will be illustrated with the examples in Section V.

IV. Routing Strategies Without Multinode Loops

Although ping-pong-type loops can be removed by strategy B, loops with more than two nodes (i.e., multinode loops) may still exist under both strategies. Multinode loops, if they occur, usually cause much severer degradation in network performance. Consequently, it is important to develop a routing strategy which eliminates multinode loops.

Strategy B removes ping-pong-type loops by disallowing each node to send its neighbors the routing messages which have been found to be useless to receivers. However, under strategy B every node can determine only the delay and the second node of each path from its network delay table. Naturally, as the amount of local information is increased, every node is expected to form more useful routing messages leading to a greater reduction of looping effects. In fact, a routing strategy can be developed to eliminate multinode loops by keeping in network delay tables not only the delay of each path but also the set of nodes which follow the first node in the ordered sequence representation of the path. Thus, the format of routing messages is modified to include the corresponding set of nodes in each path. The augmented information in the network delay table of each node will be very useful in forming suitable routing messages for its neighbors.

The number of nodes included in the routing message will henceforth be referred to as the *order* of the routing strategy. As the order of the routing strategy gets higher, it will become free of higher order loops, i.e., loops with more number of nodes. Note that strategies A and B are actually the routing strategies whose orders are 0 and 1, respectively. (A simple example for the operations under the second order routing strategy is presented in the next section.)

Performance of the kth-order routing strategy can be analyzed using the same approach as in Section III. The following definition is required to facilitate our discussion.

Definition 10: A path P_i is said to *be free of kth-order loops* iff for any two occurrences of a node in its ordered sequence representation there are at least $k + 1$ nodes between them.

Let SP^k, $k \in I^+$, be the set of paths which are free of kth order loops and $SP^k_{i,j} \equiv SP^k \cap SP_{i,j}$ be the set of paths from N_i to N_j and free of kth-order loops. Obviously, $SP^{k+1} \subseteq SP^k$, $\forall k \in I^+$. Since the relationship of $SP^*_{s,d}$ to strategy B is the same as the relationship of $SP^k_{s,d}$ to the kth-order routing strategy, the node N_s under the kth-order routing strategy chooses its nonfaulty optimal paths only from the set $SP^k_{s,d}$. The entries of the network delay table during failure recovery can be clearly described by the following theorem, whose proof is similar to that of Theorem 1 and, thus, omitted.

Theorem 4:

$$DOP^k_{s,d}(m) = \min \{d(P_j) | P_j \in SP^k_{s,d} \text{ and scn } (P_j) > m\} \ \forall m \in I^+.$$

It can be verified by Theorem 4 that the delays of nonfaulty optimal paths from N_s to N_d obtained under all strategies are indeed the same, i.e, $DOP^i_{s,d}(m_i) = DOP^j_{s,d}(m_j) = r$, $\forall i, j \in I^+$ where m_i is the number of time intervals for N_s to determine a new nonfaulty optimal path to N_d under the ith-order routing strategy after the occurrence of a link failure. From Theorem 4, we have the following theorem to determine $m_k \ \forall k \in I^+$, which is an extended version of Theorem 2 and can be proven similarly.

Theorem 5:

$$m_k = \max \{\text{scn } (P_j) | P_j \in SP^k_{s,d} \text{ and } d(P_j) < r\}.$$

From this theorem, we get $m_{k+1} \leq m_k \forall k \in I^+$. This implies that the network's adaptability is improved monotonically by adding more information in routing messages. Moreover, with the same reasoning as in Section III we can obtain a generalized form of (7) and (7′) as follows:

$$m_k: \max_{P_i \in SP^k_{s,d}} h(P_i') + \sum_{j=1}^{k} n_{i_j} h(L_{i_j}) \tag{11}$$

$$\text{subject to } d(P_i') + \sum_{j=1}^{k} n_{i_j} d(L_{i_j}) + d(P_{i_r}) < r \text{ where } L_{i_j} \in SP^k_{i_j,i_j} \text{ and } L_{i_j} \subset P_i \tag{11'}$$

where P_i', P_{i_r}, n_{i_j} and L_{i_j} are defined as before.

Define L^k_* to be a loop such that $A(L^k_*) = \min \{A(L_i) | L_i \in SP^k_{i,1}, 1 \leq i \leq n\}$. When the delay of the new nonfaulty optimal path r is large enough, $A(L^k_*)$ will become the dominating factor in determining m_k and we have the following extended results for high-order routing strategies.

Theorem 6:

$$\lim_{r\to\infty} \frac{m_i}{m_j} = \frac{A(L^i_*)}{A(L^j_*)}, \ \forall i, j \in I^+.$$

Proof: Let L_{i_m} be the loop with the minimal average delay among all loops in the path P_i. Then, from (11) we have

$$m_k = \max_{P_i \in SP^k_{s,d}} \left\{ \left\lfloor \frac{r - d(P_i') - d(P_{i_r})}{d(L_{i_m})} \right\rfloor h(L_{i_m}) + R_i + h(P_i') \right\} \text{ where } R_i < h(L_{i_m}). \tag{12}$$

Following a proof similar to the proof of Theorem 3, the desired result is obtained. ■

The above results indicate that the network's adaptability is actually enhanced with the increased amount of local information. However, according to experimental results in [5], multinode loops occur very infrequently. Besides, the operational overheads required in higher order routing strategies cannot be ignored and the feasibility of implementing higher order routing strategies must be justified. To this end, Theorem 5 provides a good indication for the usefulness of a high-order routing strategy in a certain network and can also be used to determine the minimal order of routing strategy required in a given network to remove all possible looping.

V. Examples

In this section, two examples are presented to illustrate our analytic results obtained thus far.

Example 1: In a computer network shown in Fig. 1, assume that the link $L_{3,4}$ became faulty at time 0 and delays of the other links remain constant. From (9) and (10), the number of time intervals required for each node in this network to obtain its new nonfaulty optimal path is a function of $r^* = L_{2,5}$. Thus, network performance has been examined while this parameter is being varied.

Case 1: $r^* = 16$

Given below are the operations taken under strategies A and B for the source node N_1 to obtain a new nonfaulty optimal path to the destination node N_5 by exchanging routing messages with neighboring nodes every time interval.

Under strategy A, network delay tables become as shown in Table I. From Table I, one can obtain

$$OP^A_{1,5}(t_0) = (N_1, N_3, N_4, N_5) \quad \text{where } t_0 \in (-\infty, 0)$$

$$OP^A_{1,5}(0) = (N_1, N_3, N_4, N_5)$$

$$OP^A_{1,5}(1) = OP^A_{1,5}(2) = (N_1, N_3, N_1, N_3, N_4, N_5)$$

$$OP^A_{1,5}(3) = OP^A_{1,5}(4) = (N_1, N_3, N_1, N_3, N_1, N_3, N_4, N_5)$$

$$OP^A_{1,5}(5) = OP^A_{1,5}(6) = (N_1, N_3, N_1, N_3, N_1, N_3, N_1, N_3, N_4, N_5)$$

$$OP^A_{1,5}(7) = OP^A_{1,5}(8) = (N_1, N_3, N_1, N_3, N_1, N_3, N_1, N_3, N_1, N_3, N_4, N_5)$$

$$OP^A_{1,5}(9) = (N_1, N_2, N_5).$$

Thus, $m_A = 9$, while scn $(OP^A_{1,5}(8)) = 9$. This agrees with the result of Theorem 2. In addition, one can observe from the above tables that the numbers of time intervals for N_2 and N_3 to obtain the new nonfaulty optimal paths are 6 and 10, respectively (marked by *).

On the other hand, we obtain the network delay tables under

TABLE I
NETWORK DELAY TABLES OF $N_1 N_2$ AND N_3 UNDER STRATEGY A WHERE N_5 IS THE DESTINATION NODE (a) NETWORK DELAY TABLE OF N_1 (b) NETWORK DELAY TABLE OF N_2 (c) NETWORK DELAY TABLE OF N_3

Entry	$t_0 \in (-\infty, 0)$	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$	$t \in [11, \infty)$
N_2	11	11	11	11	14	14	17	17	19	19	19*	19	19
N_3	6	6	9	9	12	12	15	15	18	18	21	21	22

(a)

Entry	$t_0 \in (-\infty, 0)$	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$	$t \in [11, \infty)$
N_1	8	8	8	11	11	14	14	17	17	20	20	21	21
N_3	9	9	12	12	15	15	18	18	21	21	24	24	25
N_5	16	16	16	16	16	16	16	16*	16	16	16	16	16

(b)

Entry	$t_0 \in (-\infty, 0)$	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t \in [10, \infty)$
N_1	7	7	7	10	10	13	13	16	16	19	19	20*
N_2	14	14	14	14	17	17	20	20	22	22	22	22
N_4	4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

(c)

TABLE II
NETWORK DELAY TABLES OF $N_1 N_2$ AND N_3 UNDER STRATEGY B WHERE N_5 IS THE DESTINATION NODE (a) NETWORK DELAY TABLE OF N_1 (b) NETWORK DELAY TABLE OF N_2 (c) NETWORK DELAY TABLE OF N_3

Entry	$t_0 \in (-\infty, 0)$	$t=0$	$t=1$	$t=2$	$t=3$	$t \in [4, \infty)$
N_2	12	12	12	19	19	19*
N_3	6	6	16	16	16	24

(a)

Entry	$t_0 \in (-\infty, 0)$	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	$t \in [5, \infty)$
N_1	8	8	8	18	18	18	26
N_3	9	9	18	18	18	25	25
N_5	16	16	16	16*	16	16	16

(b)

Entry	$t_0 \in (-\infty, 0)$	$t=0$	$t=1$	$t=2$	$t \in [3, \infty)$
N_1	13	13	13	13	20*
N_2	14	14	14	14	22
N_4	4	∞	∞	∞	∞

(c)

strategy B as shown in Table II. From this table we get

$$OP^B_{1,5}(t_0) = (N_1,\ N_3,\ N_4,\ N_5) \quad \text{where } t_0 \in (-\infty,\ 0)$$

$$OP^B_{1,5}(0) = (N_1,\ N_3,\ N_4,\ N_5)$$

$$OP^B_{1,5}(1) = (N_1,\ N_2,\ N_3,\ N_4,\ N_5)$$

$$OP^B_{1,5}(2) = OP^B_{1,5}(3) = (N_1,\ N_3,\ N_2,\ N_1,\ N_3,\ N_4,\ N_5)$$

$$OP^B_{1,5}(4) = (N_1,\ N_2,\ N_5)$$

$$m_B = 4,\ \text{scn}\ (OP^B_{1,5}(3)) = 4.$$

The numbers of time intervals required for N_2 and N_3 to obtain the new optimal paths are also reduced from six and ten to two and three, respectively.

Case 2: $r^* = 160$

Applying the same procedure as above, we can obtain the results for $r^* = 160$ with a simple calculation. The numbers of time intervals required for N_1, N_2, and N_3 to obtain their new optimal paths to N_5 are reduced from 105, 102, and 106 to 50, 49, and 51, respectively.

Case 3: $r^* = 1000$

In this case, the numbers of time intervals for N_1, N_2, and N_3 to obtain their optimal paths to N_5 are 665, 662, and 666 under strategy A and 332, 331, and 333 under strategy B. Clearly, the performance improvement is around 50 percent.

In Fig. 1, $L_A = (N_1, N_3, N_1)$ and $L_B = (N_1, N_2, N_3, N_1)$. Since $d(L_A) = 3$, $h(L_A) = 2$, $d(L_B) = 9$ and $h(L_B) = 3$, $A(L_A) = 1.5$ and $A(L_B) = 3$. The results of Cases 2 and 3 agree with Theorem 3.

Example 2: Consider the example network shown in Fig. 3. Applying the same procedure as in Example 1, the time intervals required for N_1, N_2, N_3, and N_4 to determine their new optimal paths to N_5 become 20, 19, 17, and 20, respectively, under strategy A (i.e., the 0th-order routing strategy), and 11, 10, 8, and 9, respectively, under strategy B (i.e., the first-order routing strategy). Table III illustrates the operations under the 2nd-order routing strategy. The subscript of each entry in network delay tables represents the set of the second and the third nodes of the corresponding path. The result shows that the required time intervals become 6, 5, 5, and 4, respectively. It is easy to see that this is a significant improvement over lower order (i.e., 0th and first) routing strategies.

With enough routing information, a node can find some of its neighbors unable to offer any loop-free path, and thus remove them from consideration. The entries in Table III marked by $\sim$'s represent such cases. While the network in Fig. 3 is still subject to looping under the 2nd-order routing strategy, it can be easily verified that the 2nd-order routing strategy will remove all possible looping in the network in Fig. 1 regardless of the value of r^*. This fact implies that the required order of the routing strategy to remove all looping depends on the network structure.

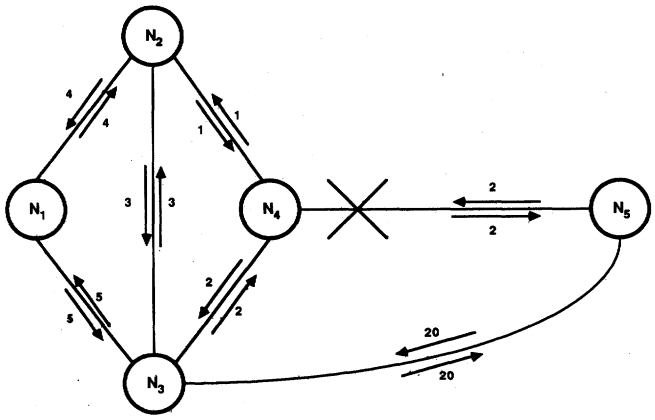


Fig. 3. An example network with link delays specified.

TABLE III
NETWORK DELAY TABLES OF $N_1N_2N_3$ AND N_4 UNDER THE SECOND-ORDER ROUTING STRATEGY WHERE N_5 IS THE DESTINATION NODE (a) NETWORK DELAY TABLE OF N_1 (b) NETWORK DELAY TABLE OF N_2 (c) NETWORK DELAY TABLE OF N_3 (d) NETWORK DELAY TABLE OF N_4.

entry	$t_0 \in (-\infty, 0)$	t=0	t=1	t=2	t=3	t=4	t=5	$t \in [6, \infty)$
N_2	$7_{\{2,4\}}$	$7_{\{2,4\}}$	$7_{\{2,4\}}$	$11_{\{2,3\}}$	$19_{\{2,4\}}$	$19_{\{2,4\}}$	$19_{\{2,4\}}$	$27_{\{2,4\}}$
N_3	$9_{\{3,4\}}$	$9_{\{3,4\}}$	$9_{\{3,4\}}$	$11_{\{3,2\}}$	$21_{\{3,4\}}$	$21_{\{3,4\}}$	$21_{\{3,4\}}$	$25_{\{3,5\}}$*

(a)

entry	$t_0 \in (-\infty, 0)$	t=0	t=1	t=2	t=3	t=4	t=5	t=6	$t \in [7, \infty)$
N_1	$13_{\{1,3\}}$	$13_{\{1,3\}}$	$13_{\{1,3\}}$	$13_{\{1,3\}}$	$\sim$	$25_{\{1,3\}}$	$25_{\{1,3\}}$	$25_{\{1,3\}}$	$29_{\{1,3\}}$
N_3	$7_{\{3,4\}}$	$7_{\{3,4\}}$	$7_{\{3,4\}}$	$23_{\{3,5\}}$	$23_{\{3,5\}}$	$23_{\{3,5\}}$	$23_{\{3,5\}}$*	$23_{\{3,5\}}$	$23_{\{3,5\}}$
N_4	$3_{\{4,5\}}$	$3_{\{4,5\}}$	$15_{\{4,3\}}$	$15_{\{4,3\}}$	$15_{\{4,3\}}$	$15_{\{4,3\}}$	$23_{\{4,3\}}$*	$23_{\{4,3\}}$	$23_{\{4,3\}}$

(b)

entry	$t_0 \in (-\infty, 0)$	t=0	t=1	t=2	t=3	t=4	t=5	t=6	$t \in [7, \infty)$
N_1	$12_{\{1,2\}}$	$12_{\{1,2\}}$	$12_{\{1,2\}}$	$12_{\{1,2\}}$	$\sim$	$24_{\{1,2\}}$	$24_{\{1,2\}}$	$24_{\{1,2\}}$	$32_{\{1,2\}}$
N_2	$6_{\{2,4\}}$	$6_{\{2,4\}}$	$6_{\{2,4\}}$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$	$\sim$
N_4	$4_{\{4,5\}}$	$4_{\{4,5\}}$	$16_{\{4,2\}}$	$16_{\{4,2\}}$	$16_{\{4,2\}}$	$16_{\{4,2\}}$	$\sim$	$\sim$	$\sim$
N_5	$20_{\{5\}}$	$20_{\{5\}}$	$20_{\{5\}}$	$20_{\{5\}}$	$20_{\{5\}}$	$20_{\{5\}}$	$20_{\{5\}}$*	$20_{\{5\}}$	$20_{\{5\}}$

(c)

entry	$t_0 \in (-\infty, 0)$	t=0	t=1	t=2	t=3	$t \in [4, \infty)$
N_2	$14_{\{2,1\}}$	$14_{\{2,1\}}$	$14_{\{2,1\}}$	$14_{\{2,1\}}$	$14_{\{2,1\}}$	$24_{\{2,3\}}$
N_3	$14_{\{3,1\}}$	$14_{\{3,1\}}$	$14_{\{3,1\}}$	$14_{\{3,1\}}$	$14_{\{3,1\}}$	$22_{\{3,5\}}$*
N_5	$2_{\{5\}}$	∞	∞	∞	∞	∞

(d)

VI. Conclusion

In this paper, we have rigorously analyzed the performance of two distributed adaptive routing strategies without resorting to simulation. We have also derived formulas for the number of time intervals required for a node to determine its new nonfaulty optimal path to any other destination node in the presence of link/node failures in the network. These results are then generalized to high-order routing strategies which are free of multinode loops. Moreover, for a given network we can determine the importance of each link/node by estimating the average number of time intervals required to recover from the failure of that link/node. That is, the link/node requiring a longer average convergence time for failure recovery possesses more importance.

Note that the order of a routing strategy determines the maximum number of nodes in an extant loop under that strategy. If we adopt a higher order routing strategy, the looping effects can be reduced further but it will induce higher operational overheads in processing the associated routing messages. It will require a complex procedure to optimally (in some sense) strike a compromise between these two mutually conflicting factors. This is a matter of our future research.

Appendix

List of Symbols

SP	Set of all paths in the network.
$SP_{s,d}$	Set of all paths from N_s to N_d.
$SP^*_{s,d}$	Set of all paths from N_s to N_d, which are free of ping-pong-type loops.
$SP^k_{s,d}$	Set of all paths from N_s to N_d, which are free of kth-order loops.
A_i	Set of all nodes adjacent to N_i.
$DL_{i,j}(m)$	The delay associated with $L_{i,j}$ at time m.
$D^A_{i\setminus j,d}(m)$	The delay from N_i via N_j to N_d in the network delay table of N_i under strategy A during the time interval $[m, m+1)$.
$OP^A_{s,d}(m)$	Path with the shortest delay from N_s to N_d in the network delay table of N_s under strategy A during the time interval $[m, m+1)$.
$DOP^A_{s,d}(m)$	The delay of $OP^A_{s,d}(m)$.
$D^B_{i\setminus j,d}(m)$	The delay from N_i via N_j to N_d in the network delay table of N_i under strategy B during the time interval $[m, m+1)$.
$OP^B_{s,d}(m)$	Path with the shortest delay from N_s to N_d in the network delay table of N_s under strategy B during the time interval $[m, m+1)$.
$DOP^B_{s,d}(m)$	The delay of $OP^B_{s,d}(m)$.
$SOP^B_{s,d}(m)$	Path with the second shortest delay from N_s to N_d in the network delay table of N_s under strategy B during the time interval $[m, m+1)$.
$DSOP^B_{s,d}(m)$	The delay of $SOP^B_{s,d}(m)$.
$OP^k_{s,d}(m)$	Path with the shortest delay from N_s to N_d in the network delay table of N_s under the kth-order routing strategy during the time interval $[m, m+1)$.
$DOP^k_{s,d}(m)$	The delay of $OP^k_{s,d}(m)$.
m_A	The number of time units required for N_s to obtain its new nonfaulty optimal path to N_d in case of a link failure under strategy A.
m_B	The number of time units required for N_s to obtain its new nonfaulty optimal path to N_d in case of a link failure under strategy B.
m_k	The number of time units required for N_s to obtain its new nonfaulty optimal path to N_d in case of a link failure under the kth-order routing strategy.

Acknowledgment

The authors are grateful to anonymous referees and to M. Woodbury of the University of Michigan for comments on the initial draft of this paper.

References

[1] L. Tymes, "Routing and flow control in TYMNET," *IEEE Trans. Commun.*, vol. COM-29, pp. 287-293, Apr. 1981.
[2] J. M. McQuillan, "Routing algorithms for computer networks—a survey," in *Proc. 1977 Nat. Telecommun. Conf.*, Dec. 1977, p. 28.

[3] C. W. Brown and M. Schwartz, "Adaptive routing in centralized computer communication networks," in *Proc. IEEE Int. Conf. Commun.*, June 1975, pp. 47-12 to 47-16.

[4] B. W. Boehm and R. L. Mobley, "Adaptive routing techniques for distributed communication systems," *IEEE Trans. Commun. Tech.*, vol. COM-17, no. 3, pp. 340–349, June 1969.

[5] W. E. Naylor, "A loop-free adaptive routing algorithm for packet switched networks," in *Proc. 4th Data Commun. Symp.*, Quebec, P.Q., Canada, Oct. 1975, pp. 7-9 to 7-14.

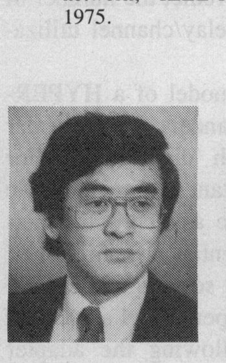
[6] J. M. McQuillan, "Adaptive routing algorithms for distributed computer networks," BBN Rep. 2831, May 1974.

[7] J. M. McQuillan and D. C. Walden, "The ARPA network design decisions," *Comput. Networks,* vol. 1, pp. 243–289, Aug. 1977.

[8] J. M. McQuillan, I. Richer, and E. C. Rosen, "An overview of the new routing algorithm for the ARPANET," in *Proc. 6th Data Commun. Symp.*, Nov. 1979, pp. 63–68.

[9] L. Kleinrock and H. Opderbeck, "Throughput in the ARPANET-protocols and measurement," *IEEE Trans. Commun.*, vol. COM-25, pp. 367–376, Jan. 1977.

[10] T. Cegrell, "A routing procedure for the TIDAS message-switching network," *IEEE Trans. Commun.*, vol. COM-23, pp. 575–585, June 1975.



Kang G. Shin (S'75–M'78–SM'83) received the B.S. degree in electronics engineering from Seoul National University, Seoul, South Korea, in 1970, and both the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, in 1976 and 1978, respectively.

From 1970 to 1972 he served in the Korean Army as an ROTC officer and from 1972 to 1974 he was on the Research Staff of the Korea Institute of Science and Technology, Seoul, working on the design of VHF/UHF communication systems. From 1974 to 1978 he was a Teaching/Research Assistant and then an Instructor in the School of Electrical Engineering, Cornell University. From 1978 to 1982 he was an Assistant Professor at Rensselaer Polytechnic Institute, Troy, NY. He was also a Visiting Scientist at the U.S. Air Force Flight Dynamics Laboratory in Summer 1979 and at Bell Laboratories, Holmdel, NJ, in Summer 1980 where his work was concerned with distributed airborne computing and cache memory architecture, respectively. He also taught short courses for the IBM Computer Science Series in the area of computer architecture. Since September 1982, he has been with the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor, MI, where he is currently an Associate Professor. His current teaching and research interests are in the areas of distributed and fault-tolerant computing, computer architecture, and robotics and automation.

Dr. Shin is a member of the Association for Computing Machinery, Sigma Xi, and Phi Kappa Phi.

Ming-Syan Chen was born in Taiwan, Republic of China, on November 17, 1959. He received the B.S. degree in electrical engineering from National Taiwan University in 1982 and the M.S. degree in computer, information and control engineering from The University of Michigan, Ann Arbor, in 1985.

He is now a Research Assistant and working towards the Ph.D. degree in the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. His research interests include computer networks, distributed processing systems, computer architecture, and graph theory.