Minimal Order Loop-Free Routing Strategy

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

***Abstract*—Conventional distributed adaptive routing strategies usually work well for packet switching networks only in the absence of link/node failures. However, they cannot avoid looping messages for an extended period in case of link/node failures.**

In this paper, we develop a multiorder routing strategy which is loop-free even in the presence of link/node failures. Unlike most conventional methods in which the same routing strategy is applied indiscriminately to all nodes in the network, nodes under the proposed strategy may adopt different routing strategies in accordance with the network structure. We not only develop the formulas to determine the minimal order of routing strategy for each node to eliminate looping completely, but also propose a systematic procedure to strike a compromise between the operational overhead and network adaptability. Several illustrative examples are also presented.

***Index Terms*—Distributed adaptive routing strategies, distribution vector, looping effects, multiorder strategy, strategy compatibility.**

I. Introduction

FOR packet switching networks, routing is a key to their performance and reliability [1], [2]. Among the various routing algorithms proposed thus far [3]–[10], distributed adaptive routing algorithms have drawn considerable attention because of their high potential for reliability and adaptability. The ARPANET's previous routing strategy (APRS) [3] is a typical example of these. 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 the other nodes is passed to all of its adjacent nodes as a routing message at every fixed time interval (i.e., 128 ms in APRS). Note, however, that under APRS each node sends the *same* routing message to *all* its neighbors without making any distinction between receiving nodes. This forces some nodes to receive useless routing messages, thereby resulting in undesirable looping in case of link/node failures. The network recovery process after certain failures will thus be delayed [11]. An example of the network recovery process under APRS for the network in Fig. 1 is given in Table I. Notice that it requires 20, 19, 17, and 20 time intervals, respectively, for N_1, N_2, N_3, and N_4 to get their new optimal paths to N_5.

Manuscript received August 1, 1987; revised January 28, 1988. This work was supported in part by the Office of Naval Research under Contract N00014-85-K-0122. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of ONR.

K. G. Shin is with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109.

M.-S. Chen was with the Real-Time Computing Laboratory, the University of Michigan, Ann Arbor, MI 48109. He is now with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

IEEE Log Number 9035951.

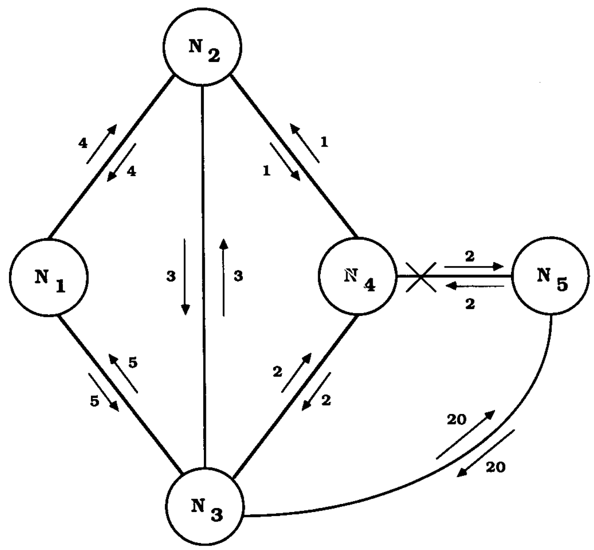

Fig. 1. An example network where $L_{4,5}$ is broken.

The routing algorithms proposed in [5]–[7] have the same major features as the one in APRS, except they employ more provisions to cope with network failures. However, they still cannot avoid some inherent drawbacks such as poor adaptability and inefficiency [7], [12]. The ARPANET's current routing strategy (ACRS) [8] uses a different approach for handling routing messages. In ACRS, every node in the network is required to keep and maintain information of the entire network. ACRS will always reach a correct routing decision as long as the global information at each node is accurate and consistent. However, this strategy requires every node to contain a large storage area for the global information and may make the entire network congested with messages for updating the global information.

The TIDAS network in [9] adopted a routing strategy which is similar to APRS except for the following modification. If the routing message is sent from node N_j to node N_i which is the second node in the optimal path from N_j to some other destination node N_d, the delay of the optimal path from N_j to N_d was replaced with the delay of its *second optimal path* in the routing message passed to N_i. An example of the network recovery process for the network in Fig. 1 under the above modification to APRS is given in Table II. It can be seen that the time intervals required for N_1, N_2, N_3, and N_4 to determine their new optimal paths to N_5 become 11,

0018-9340/90/0700-0870$01.00 © 1990 IEEE

TABLE I

NETWORK DELAY TABLES OF N_1, N_2, N_3, AND N_4 UNDER APRS WHERE N_5 IS THE DESTINATION NODE

entry	$t_0 \in (-\infty, 0)$	t=0	t=1	t=2	t=3	$t=k,\ 4 \le k \le 15$	t=16	t=17	t=18	t=19	$t \in [20, \infty)$
N_2	7	7	7	9	9	$\lfloor \frac{n}{2} \rfloor 2+7$	23	23	25	25	27
N_3	9	9	9	11	11	$\lfloor \frac{n}{2} \rfloor 2+9$	25	25	25	25	25*

(a) Network delay table of N_1.

entry	$t_0 \in (-\infty, 0)$	t=0	t=1	t=2	t=3	$t=k,\ 4 \le k \le 15$	t=16	t=17	t=18	t=19	$t \in [20, \infty)$
N_1	11	11	11	11	13	$\lceil \frac{n}{2} \rceil 2+9$	25	27	27	29	29
N_3	7	7	7	9	9	$\lfloor \frac{n}{2} \rfloor 2+7$	23	23	23	23	23
N_4	3	3	5	5	7	$\lfloor \frac{n}{2} \rfloor 2+3$	19	21	21	23*	23

(b) Network delay table of N_2.

entry	$t_0 \in (-\infty, 0)$	t=0	t=1	t=2	t=3	$t=k,\ 4 \le k \le 15$	t=16	t=17	t=18	t=19	$t \in [20, \infty)$
N_1	12	12	12	12	14	$\lceil \frac{n}{2} \rceil 2+10$	26	28	28	30	30
N_2	6	6	6	8	8	$\lfloor \frac{n}{2} \rfloor 2+6$	22	22	24	24	26
N_4	4	4	6	6	8	$\lceil \frac{n}{2} \rceil 2+4$	20	22	22	24	24
N_5	20	20	20	20	20	20	20	20*	20	20	20

(c) Network delay table of N_3.

entry	$t_0 \in (-\infty, 0)$	t=0	t=1	t=2	t=3	$t=k,\ 4 \le k \le 15$	t=16	t=17	t=18	t=19	$t \in [20, \infty)$
N_2	4	4	4	6	6	$\lfloor \frac{n}{2} \rfloor 2+4$	20	20	22	22	24
N_3	6	6	6	8	8	$\lfloor \frac{n}{2} \rfloor 2+6$	22	22	22	22	22*
N_5	2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

(d) Network delay table of N_4.

TABLE II

NETWORK DELAY TABLES OF N_1, N_2, N_3, AND N_4 UNDER THE TIDAS ROUTING STRATEGY WHERE N_5 IS THE DESTINATION NODE

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=11	$t \in [12, \infty)$
N_2	7	7	7	11	13	13	17	19	19	23	25	25	27	27
N_3	9	9	9	11	15	15	17	21	21	23	25	25	25*	25

(a) Network delay table of N_1.

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=11	$t \in [12, \infty)$
N_1	13	13	13	13	15	19	19	21	25	25	27	29	29	29
N_3	7	7	7	13	13	13	19	19	19	23	23	23	23*	23
N_4	3	3	9	9	9	15	15	15	21	21	21	23	23	23

(b) Network delay table of N_2.

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=11	$t \in [12, \infty)$
N_1	12	12	12	12	16	18	18	22	24	24	28	30	30	32
N_2	6	6	6	12	12	12	18	18	18	24	24	24	26	26
N_4	4	4	10	10	10	16	16	16	22	22	22	26	26	26
N_5	20	20	20	20	20	20	20	20	20	20*	20	20	20	20

(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=11	$t \in [12, \infty)$
N_2	8	8	8	8	14	14	14	20	20	20	24	24	24	24
N_3	8	8	8	8	14	14	14	20	20	20	22*	22	22	22
N_5	2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

(d) Network delay table of N_4.

10, 8, and 9, respectively. Although this modification leads to a significant improvement over APRS in reducing the looping effects, it does not eliminate them completely. In [4], we have rigorously analyzed the performance of a routing strategy using the above modification. We proved that, although ping-pong type loops (i.e., loops with two nodes) can be removed by the above modification, multinode loops (i.e., loops with more than two nodes) may still exist. More importantly, we extended our analytical results to routing strategies which are free of multinode loops. We showed that a routing strategy can eliminate multinode loops by keeping in network delay tables not only the delay of each minimal path but also a set of first few nodes in the path. The number of nodes included in the routing message is referred to as the *order* of the corresponding routing strategy. The number of nodes in a loop that can be present under a routing strategy increases with the order of the routing strategy [4].

To eliminate looping completely, one may consider the following straightforward approach. All nodes in each path are included in routing messages and sent to neighboring nodes. However, this naive approach is very inefficient due to its excessive overhead. Consequently, it is very important to determine the *minimal* order of routing strategy required for each node to make the network completely loop-free. As we shall prove later, depending on the network structure, we can determine the portion of a path that each node should keep and send to its neighboring nodes in order to eliminate looping completely. Unlike the other distributed routing strategies where the same strategy is applied indiscriminately to every node in a network, the order of a node's routing strategy depends on the network topology and varies from one node to another. It will be interesting to see that our proposed strategy will require most nodes to keep only a fairly small portion of each path and can still remove looping completely. Notice that we remove looping effects by augmented minimal delay vectors, whereas the method described in [10] is based on the use of extensive protocols.

This paper is organized as follows. In Section II, we present necessary definitions and notation, and then introduce the multiorder routing strategy. In Section III, we develop formulas to determine the order of the routing strategy required for each node to eliminate looping effects completely. We take into consideration the operational overhead in handling routing messages in Section IV and optimize the tradeoff between the network adaptability and the operational overhead. Complexity of the optimization algorithm is also analyzed. This paper concludes with Section V.

II. Description of the Routing Strategy

A. Definitions and Notation

For a computer network N, let $V(N)$ and $E(N)$ denote, respectively, the set of computer nodes and the set of computer links with $|V(N)| = p$ and $|E(N)| = q$, where $|S|$ represents the cardinality of the set S. Let DL_{ij} be the delay of a direct link L_{ij} from N_i to N_j. The set of nodes adjacent to N_i is denoted by A_i. There are usually many paths from N_i to N_j, which are represented by the set SP_{ij}, and let $\mathrm{SP} = \cup_{N_i, N_j \in V(N)} \mathrm{SP}_{ij}$. Let P_{ij} denote the path with the shortest delay (i.e., the optimal path) in SP_{ij} and P_{ij-uv} be the shortest delay path in the set $\mathrm{SP}_{ij} - \{L_{uv}\}$. Clearly, P_{ij-uv} is the new optimal path from N_i to N_j if the link L_{uv} becomes faulty. Note that $P_{ij-uv} = P_{ij}$ only when L_{uv} is not a part of P_{ij}.

A path in N is expressed by an *ordered sequence representation* of nodes. For example, a path $P_i \in \mathrm{SP}$ can be represented by $(N_{i_1}, N_{i_2}, \cdots, N_{i_m})$. Let $H_k(P_i)$ be the set of the first k nodes of a path $P_i \in \mathrm{SP}$. For a path $P_i = (N_{i_1}, N_{i_2}, \cdots, N_{i_m})$, $H_k(P_i) = \{N_{i_1}, N_{i_2}, \cdots, N_{i_k}\}$ if $m \geq k$, and $H_k(P_i) = \{N_{i_1}, N_{i_2}, \cdots, N_{i_m}\}$ otherwise. In addition, a function $h\colon \mathrm{SP} \to \boldsymbol{I}^+$ is the *hop function* of a path, where $h(P_i)$ denotes the number of links in a path $P_i \in \mathrm{SP}$ and $\boldsymbol{I}^+$ the set of positive integers, and a function $d\colon \mathrm{SP} \to \boldsymbol{R}^+$ is the *delay function* of a path, where $d(P_i)$ is the summation of all link delays in a path $P_i \in \mathrm{SP}$ and $\boldsymbol{R}^+$ the set of positive real numbers. A *loop* is a path with the minimal number of nodes which starts and ends at the same node, and the set of loops starting and ending at N_j is denoted by SL_{jj}. Also, a loop L_i is called a *kth order loop* if the number of hops in L_i is $k+1$, i.e., $h(L_i) = k+1$.

For example, while $(N_2, N_3, N_2, N_3, N_2)$ is not a loop, (N_1, N_2, N_3, N_1) is a second-order loop. Besides, to illustrate the network recovery process after link/node failures, we assume that the network N is connected throughout our discussion.

B. Description of Multiorder Routing Strategy

The main schemes used in all kth order routing strategies are basically the same, except that different values of k indicate different amounts of information to be recorded in the network delay table. Let $\mathrm{NT}^k_{i\backslash jd}$ denote the information kept in the network delay table of N_i about the shortest delay path from N_i via $N_j \in A_i$ to N_d under the kth order routing strategy. Also, let $P_{i\backslash jd}$ be the path specified by $\mathrm{NT}^k_{i\backslash jd}$. Then, $\mathrm{NT}^k_{i\backslash jd}$ is a record containing two fields: $\mathrm{NT}^k_{i\backslash jd}.\mathrm{dly}$ and $\mathrm{NT}^k_{i\backslash jd}.\mathrm{set}$, where $\mathrm{NT}^k_{i\backslash jd}.\mathrm{dly}$ denotes the delay of $P_{i\backslash jd}$ and $\mathrm{NT}^k_{i\backslash jd}.\mathrm{set}$ is an ordered set of the first $k+1$ nodes in $P_{i\backslash jd}$. That is, $\mathrm{NT}^k_{i\backslash jd}.\mathrm{dly} = d(P_{i\backslash jd})$ and $\mathrm{NT}^k_{i\backslash jd}.\mathrm{set} = H_{k+1}(P_{i\backslash jd})$. Let $\mathrm{RM}^k_{i\leftarrow jd}$ denote the routing message sent from $N_j \in A_i$ to N_i about the optimal from N_j to N_d under the kth order routing strategy. $\mathrm{RM}^k_{i\leftarrow jd}$ is again composed of two fields, $\mathrm{RM}^k_{i\leftarrow jd}.\mathrm{dly}$ and $\mathrm{RM}^k_{i\leftarrow jd}.\mathrm{set}$, which can be determined from the network delay table of N_j as follows.

$$\mathrm{RM}^k_{i\leftarrow jd}.\mathrm{dly} = \min_{\substack{N_q \in A_j \text{ and} \\ N_i \notin \mathrm{NT}^k_{j\backslash qd}.\mathrm{set}}} \mathrm{NT}^k_{j\backslash qd}.\mathrm{dly}, \tag{1}$$

$$\mathrm{RM}^k_{i\leftarrow jd}.\mathrm{set} = H_k(P_{i^*}) \qquad \text{where } P_{i^*} \text{ is the path with the delay } \mathrm{RM}^k_{i\leftarrow jd}.\mathrm{dly}. \tag{2}$$

When the routing message $\mathrm{RM}^k_{i\leftarrow jd}$ is received by N_i, N_i uses this message to update its network delay table as follows, where $\odot$ means prefixing a node to an ordered set.

$$\mathrm{NT}^k_{i\backslash jd}.\mathrm{dly} = \mathrm{RM}^k_{i\leftarrow jd}.\mathrm{dly} + \mathrm{DL}_{ij}, \tag{3}$$

$$\mathrm{NT}^k_{i\backslash jd}.\mathrm{set} = \{N_i\} \odot \mathrm{RM}^k_{i\leftarrow jd}.\mathrm{set}. \tag{4}$$

TABLE III
NETWORK DELAY TABLES OF N_1, N_2, N_3, AND N_4 UNDER THE SECOND-ORDER ROUTING STRATEGY WHERE N_5 IS THE DESTINATION NODE

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) Network delay table of N_1.

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	$13_{\{1,3\}}$	$13_{\{1,3\}}$	$13_{\{1,3\}}$	$13_{\{1,3\}}$	~	$25_{\{1,3\}}$	$25_{\{1,3\}}$	$25_{\{1,3\}}$	$29_{\{1,3\}}$
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) Network delay table of N_2.

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	$12_{\{1,2\}}$	$12_{\{1,2\}}$	$12_{\{1,2\}}$	$12_{\{1,2\}}$	~	$24_{\{1,2\}}$	$24_{\{1,2\}}$	$24_{\{1,2\}}$	$32_{\{1,2\}}$
N_2	$6_{\{2,4\}}$	$6_{\{2,4\}}$	$6_{\{2,4\}}$	~	~	~	~	~	~
N_4	$4_{\{4,5\}}$	$4_{\{4,5\}}$	$16_{\{4,2\}}$	$16_{\{4,2\}}$	$16_{\{4,2\}}$	$16_{\{4,2\}}$	~	~	~
N_5	$20_{\{5\}}$	$20_{\{5\}}$	$20_{\{5\}}$	$20_{\{5\}}$	$20_{\{5\}}$	$20_{\{5\}}$	$20_{\{5\}}$*	$20_{\{5\}}$	$20_{\{5\}}$

(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	$14_{\{2,1\}}$	$14_{\{2,1\}}$	$14_{\{2,1\}}$	$14_{\{2,1\}}$	$14_{\{2,1\}}$	$24_{\{2,3\}}$
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) Network delay table of N_4.

Notice that APRS and the routing strategy in TIDAS are actually special cases of the above strategy when $k = 0$ and $k = 1$, respectively. For the network in Fig. 1, the network operations under the second-order routing strategy are described in Table III, where the subscript of each entry in the network delay tables represents the set of the second and third nodes of the corresponding path. If enough routing information is recorded, a node can determine that the use of some of its neighbors will not lead to loop-free paths; such neighbors will be removed from the construction of loop-free paths. The entries in Table III marked by $\sim$ represent such cases. It can be verified by Tables I, II, and III that the kth order routing strategy is free of jth-order loops $\forall\, 1 \le j \le k$. It is interesting to see that the second-order routing strategy eliminates the first-order loop (N_2, N_4, N_2) and the second-order loop (N_2, N_4, N_3, N_2), which had caused the slowdown of the recovery processes in Tables I and II, respectively. As a result, the required time intervals for N_1, N_2, N_3, and N_4 to get their new optimal paths to N_5 are reduced, respectively, to 6, 5, 5 and 4. It can be seen that increasing the order of routing strategy speeds up each node's adaptation to failures of links/nodes in the network.

III. MINIMAL ORDER LOOP-FREE ROUTING STRATEGY

Although a higher order routing strategy is necessary for some nodes to avoid potential looping, it usually contributes nothing but higher operational overheads to other nodes. Thus, it is very important to determine the minimal order routing strategy required for *each* node to avoid all potential looping. Consider the case when L_{ij} becomes faulty. Let $R_{i \leftarrow k,j}$ denote the required order of routing message sent from $N_k \in A_i$ to N_i such that the routing message will not result in any path containing loops in the network delay table of N_i. To facilitate our presentation, we define a set of loops $S_{i \leftarrow k,j}$ as follows.

$$S_{i \leftarrow k,j} \equiv \{L_{i^*} \mid L_{i^*} \in \mathrm{SL}_{ii},\ \mathrm{2nd}(L_{i^*}) = N_k \text{ and } d(L_{i^*}) < d(P_{ij-ij}) - \mathrm{DL}_{ij}\}$$

where $N_i \in V(N)$, $N_kN_j \in A_i$, and $\mathrm{2nd}(L_{i^*})$ is the second node in the loop L_{i^*}. Then, the quantity $R_{i \leftarrow k,j}$ can be determined by the following theorem.

Theorem 1:

$$R_{i \leftarrow k,j} = \begin{cases} \max\limits_{L_{i^*} \in S_{i \leftarrow k,j}} \{h(L_{i^*})\}, & \text{if } S_{i \leftarrow k,j} \ne \varnothing, \\ 0, & \text{if } S_{i \leftarrow k,j} = \varnothing. \end{cases}$$

Proof: If the required order of the routing message about the path P_{kj} from $N_k \in A_i$ to N_i, denoted by $r_{i \leftarrow k,j}$, is less than $R_{i \leftarrow k,j}$, there is a loop $L_{i^*} \in \mathrm{SL}_{ii}$ such that $\mathrm{2nd}(L_{i^*}) = N_k$ and $d(L_{i^*}) + \mathrm{DL}_{ij} < d(P_{ij-ij})$. Thus, the path from N_k via L_{i^*} to N_i and then via L_{ij} to N_j contains the delay $d(L_{i^*}) - \mathrm{DL}_{ik} + \mathrm{DL}_{ij}$. The delay of the new optimal path

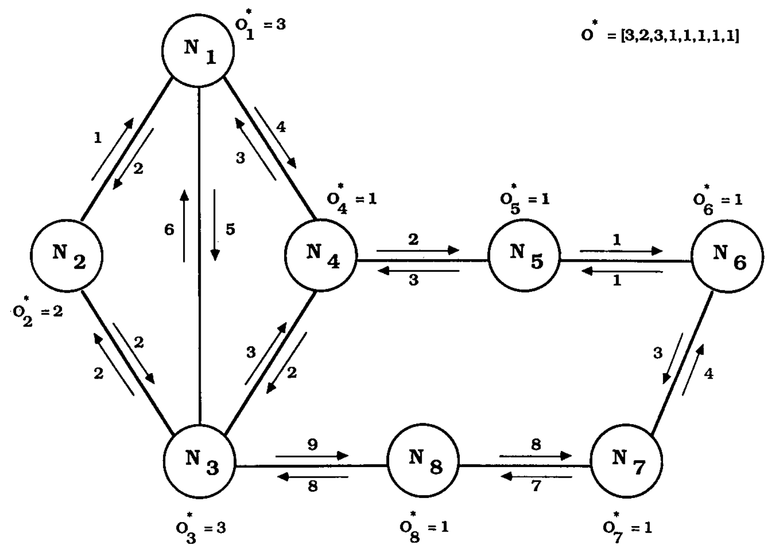

Fig. 2. A network for Example 1.

from N_k to N_j must be greater than $d(L_{i*}) - \mathrm{DL}_{ik} + \mathrm{DL}_{ij}$, since $d(P_{ij-ij})$ will otherwise be less than $d(L_{i*}) + \mathrm{DL}_{ij}$, leading to a contradiction. Thus, if $r_{i\leftarrow k,j} < h(L_{i*})$ before N_k finds its new optimal path, the delay $d(L_{i*}) - \mathrm{DL}_{ik} + \mathrm{DL}_{ij}$ will be sent to N_i, thereby resulting in a path with a loop in the network delay table of N_i. Therefore, $R_{i\leftarrow k,j} \leq r_{i\leftarrow k,j}$.

Next, we want to prove that a routing message of the order $R_{i\leftarrow k,j}$ sent from N_k to N_i will not result in a path with loops for N_i. Suppose there is a resulting loop L_{i*}. Then, $d(L_{i*}) + \mathrm{DL}_{ij} < d(P_{ij-ij})$ and $2\mathrm{nd}(L_{i*}) = N_k$. By (2) and (4), we get $h(L_{i*}) > R_{i\leftarrow k,j}$, leading to a contradiction. This means $R_{i\leftarrow k,j} \geq r_{i\leftarrow k,j}$, and $R_{i\leftarrow k,j} = r_{i\leftarrow k,j}$ thus follows. Q.E.D.

Note that the minimal order routing strategy for N_i must be determined by routing messages from all of its neighboring nodes. Let $R_{i\leftarrow k}$ represent the order of routing message sent from $N_k \in A_i$ to N_i to avoid all potential looping, i.e.,

$$R_{i\leftarrow k} = \max_{\substack{N_j \in A_i \text{ and} \\ j \neq k}} \{R_{i\leftarrow k,j}\}. \tag{5}$$

The minimal order of routing strategy required for N_i to avoid all potential looping, denoted by O_i^*, can be determined by the following corollary.

Corollary 1.1: There is no looping in the network if and only if

$$O_i^* = \max_{N_j \in A_i} \{R_{j\leftarrow i}\}, \qquad \forall N_i \in V(N).$$

Proof: If $O_i^* = \max_{N_j \in A_i} \{R_{j\leftarrow i}\}$, then it immediately follows from Theorem 1 that there is no looping when N_i adopts the O_i^*th order routing strategy. Next, we want to prove that O_i^* is the minimal order of routing strategy for N_i to avoid all potential looping. Suppose that the order of routing strategy adopted by N_i, denoted by O_i, is less than O_i^*. Then there exists an $R_{j\leftarrow i}$ such that $R_{j\leftarrow i} > O_i$. From the equations in Theorem 1, there is a node N_k such that $R_{j\leftarrow i,k} > O_i$. Thus, when the link L_{jk} becomes faulty, the routing message sent from N_i to N_j will result in a path with an $R_{j\leftarrow i}$th order loop in the network delay table of N_j, where $O_i < R_{j\leftarrow i} \leq O_i^*$. This is contradictory to the hypothesis of no looping. Q.E.D.

Although the above formulas can determine the minimal order routing strategy for each node, one can find from the operation of routing strategy that the difference between the orders of routing strategies of two adjacent nodes *cannot* be greater than one. (We term this fact the "strategy compatibility.") Otherwise, a node with the lower order routing strategy would not be able to generate the routing messages required for all of its neighboring nodes. Thus, we may have to increase the orders of routing strategies of some nodes to hold the strategy compatibility. We present a simple example to demonstrate the ideas presented thus far.

Example 1: Consider the example network in Fig. 2. For this network we will determine the minimal order of loop-free routing strategy for each node.

a) The required order of loop-free routing strategy, O_i^*, $1 \leq i \leq 8$, can be determined by the following two steps.

Step 1: Using Theorem 1 and (5), determine $R_{i\leftarrow j}$, $N_j \in A_i$, $1 \leq i \leq 8$. For N_1, we get

$$\begin{cases} R_{1\leftarrow 4,2} = 0 \\ R_{1\leftarrow 4,3} = 0 \end{cases} \Rightarrow R_{1\leftarrow 4} = 0$$

$$\begin{cases} R_{1\leftarrow 3,2} = 0 \\ R_{1\leftarrow 3,4} = 0 \end{cases} \Rightarrow R_{1\leftarrow 3} = 0$$

$$\begin{cases} R_{1\leftarrow 2,3} = 0 \\ R_{1\leftarrow 2,4} = 1 \end{cases} \Rightarrow R_{1\leftarrow 2} = 1.$$

For N_2, we have $R_{2\leftarrow 1,3} = R_{2\leftarrow 1} = 1$ and $R_{2\leftarrow 3,1} = R_{2\leftarrow 3} = 1$.

In case of N_3, we obtain

$$\begin{cases} R_{3\leftarrow 1,2} = 0 \\ R_{3\leftarrow 1,4} = 0 \\ R_{3\leftarrow 1,8} = 0 \end{cases} \Rightarrow R_{3\leftarrow 1} = 0$$

$$\begin{cases} R_{3\leftarrow 2,1} = 0 \\ R_{3\leftarrow 2,4} = 1 \\ R_{3\leftarrow 2,8} = 2 \end{cases} \Rightarrow R_{3\leftarrow 2} = 2$$

and then $R_{3\leftarrow 4,1} = 0$, $R_{3\leftarrow 4,2} = 1$, $R_{3\leftarrow 4,8} = 1 \Rightarrow R_{3\leftarrow 4} = 2$, and $R_{3\leftarrow 8,1} = 0$, $R_{3\leftarrow 8,2} = 0$, $R_{3\leftarrow 8,4} = 0 \Rightarrow R_{3\leftarrow 8} = 0$. Following the same procedure, we get $R_{4\leftarrow 1} = 3$, $R_{4\leftarrow 3} = 3$, and $R_{4\leftarrow 5} = 0$ for N_4, $R_{5\leftarrow 6} = 1$ and $R_{5\leftarrow 4} = 1$ for N_5, $R_{6\leftarrow 5} = 1$ and $R_{6\leftarrow 7} = 1$ for N_6, $R_{7\leftarrow 6} = 1$ and $R_{7\leftarrow 8} = 1$ for N_7, and $R_{8\leftarrow 7} = 0$ and $R_{8\leftarrow 3} = 0$ for N_8.

Step 2: Using Corollary 1.1, determine O_i^*, $1 \le i \le 8$.

$$\begin{cases} R_{2\leftarrow 1} = 1 \\ R_{3\leftarrow 1} = 0 \\ R_{4\leftarrow 1} = 3 \end{cases} \Rightarrow O_1^* = 3$$

$$\begin{cases} R_{1\leftarrow 2} = 1 \\ R_{3\leftarrow 2} = 2 \end{cases} \Rightarrow O_2^* = 2$$

$$\begin{cases} R_{2\leftarrow 3} = 1 \\ R_{1\leftarrow 3} = 0 \\ R_{4\leftarrow 3} = 3 \\ R_{8\leftarrow 3} = 0 \end{cases} \Rightarrow O_3^* = 3$$

$$\begin{cases} R_{1\leftarrow 4} = 0 \\ R_{3\leftarrow 4} = 1 \\ R_{5\leftarrow 4} = 1 \end{cases} \Rightarrow O_4^* = 1$$

and then, $R_{4\leftarrow 5} = 0$, $R_{6\leftarrow 5} = 1 \Rightarrow O_5^* = 1$; $R_{5\leftarrow 6} = 1$, $R_{7\leftarrow 6} = 1 \Rightarrow O_6^* = 1$; $R_{6\leftarrow 7} = 1$, $R_{8\leftarrow 7} = 0 \Rightarrow O_7^* = 1$ and $R_{3\leftarrow 8} = 0$, $R_{7\leftarrow 8} = 1 \Rightarrow O_8^* = 1$. For this example network we get the minimum order vector, $O^* = [3, 2, 3, 1, 1, 1, 1, 1]$, and then [3, 2, 3, 2, 1, 1, 1, 2] after considering the strategy compatibility.

IV. Operational Overhead and Looping Delay Tradeoff

As mentioned earlier, the multiorder routing strategy in a node usually causes its neighboring nodes to increase their orders of routing strategies to satisfy the strategy compatibility. If we consider the operational overhead in handling routing messages, it may not be worth introducing a considerable amount of overhead for infrequent failures or for some failures whose recovery costs are not high. This implies the need of striking a compromise between looping effects and the operational overhead, and determining the optimal order of routing strategy for each node.

A. Optimization of Tradeoff

Although various procedures are conceivable to determine the operational overhead in (1) and (2), the main idea can be described as follows. The cardinality of $\mathrm{RM}^k_{i\leftarrow jd}$.set increases linearly with the order of routing strategy, meaning that the memory requirement for the routing strategy is linearly dependent on the value of k. The computational overhead for (3) and (4) is straightforward and has little dependence on k. However, from (1) it is easy to see that for a given network structure the number of comparisons required is linearly proportional to k. The computational cost is therefore linearly proportional to k.

Let c_c and m_c denote the incremental cost of computation and memory, respectively, when the order of routing strategy is incremented by one. Let $R_c(k)$ be the cost required per second for a node adopting the kth order strategy to generate and process a routing message. Note that the exact expression of $R_c(k)$ has a close dependence on the hardware and software used for each node computer. Following the above reasoning, $R_c(k)$ can, however, be approximately expressed as $[(m_c + c_c)k + \mathit{offset}]$, where *offset* is the sum of contributions from the factors unrelated to k.

Define a *strategy vector* as a p-tuple whose ith element is the order of the routing strategy adopted by N_i. (Recall that p is the number of nodes in N.) A network together with its adopted strategy vector is termed a *configuration*. Let O_i^k denote the order of the routing strategy adopted by N_i when the configuration is C_k. The operational overhead per second induced with the configuration C_k can then be determined by the formula

$$\mathrm{RC}(C_k) = \sum_{i=1}^{p} R_c(O_i^k). \tag{6}$$

Assume that the traffic density between every pair of nodes in the network is uniform. The expected number of time intervals required for an arbitrary node to find a new nonfaulty optimal path to any other node when L_{ij} becomes faulty can be expressed as

$$R_t(L_{ij}; C_k) = \frac{1}{p(p-1)} \sum_{N_u \in V(N)} \sum_{\substack{N_v \in V(N) \\ \text{and } u \neq v}} m_{uv-ij}(C_k) \tag{7}$$

where $m_{uv-ij}(C_k)$ denotes the number of time intervals for N_u to obtain a new nonfaulty optimal path to N_v when the configuration is C_k and L_{ij} becomes faulty. The expected number of time intervals to recover from an arbitrary link failure (i.e., switch from a broken path to a new nonfaulty path) in the configuration C_k can then be determined by

$$\mathrm{RT}(C_k) = \frac{1}{q} \sum_{L_{ij} \in E(N)} R_t(L_{ij}; C_k). \tag{8}$$

Note that $\mathrm{RT}(C_k)$ can be viewed as a *measure of adaptability* of C_k. The smaller $\mathrm{RT}(C_k)$, the better adaptability C_k possesses. To compute (7), we must show how to determine $m_{uv-ij}(C_k)\, \forall u, v, i, j$, and k. Consider the case when in a configuration C_k, N_i does not adopt a routing strategy of an order sufficient enough to remove looping completely. In such a case, by Corollary 1.1, certain link failures will induce looping. From (5) and Theorem 1, we can represent the set of loops (SPL) induced by the insufficient order of routing strategy as follows.

$$\mathrm{SPL}(N_i; C_k) = \bigcup_{N_j \in A_i} \bigcup_{\substack{N_1 \in A_j \\ \text{and } q \neq i}} \{L_{j^*} | L_{j^*} \in S_{j \leftarrow i, q} \text{ and } h(L_{j^*}) > O_i^k\}. \quad (9)$$

The set of all potential loops in the network with the configuration C_k can be expressed by

$$\mathrm{SPL}(C_k) = \bigcup_{N_i \in V(N)} \mathrm{SPL}(N_i; C_k). \quad (10)$$

Let $L(P_{i^*})$ denote the set of loops in the path P_{i^*}. P_{i^*} is said to be a *possible path* in the configuration C_k if $L(P_{i^*}) \subseteq \mathrm{SPL}(C_k)$, i.e., every loop contained in P_{i^*} belongs to $\mathrm{SPL}(C_k)$. Denote the set of all possible paths in the configuration C_k by $\mathrm{LP}(C_k)$. Then, $m_{uv-ij}(C_k)$ can be expressed by

$$m_{uv-ij}(C_k) = \max_{\substack{P_{u^*} \in \mathrm{LP}(C_k) \text{ and} \\ P_{u^*} \in \mathrm{SP}_{u,i}}} \{h(P_{u^*}) | d(P_{u^*}) < d(P_{uv-ij}) - d(P_{iv})\}. \quad (11)$$

Note that P_{u^*} in (11) does not have to contain all loops in $\mathrm{SPL}(C_k)$. The loops contained in P_{u^*} could be any subset of $\mathrm{SPL}(C_k)$ though. Let Sb denote a subset of $\mathrm{SPL}(C_k)$ and $V(\mathrm{Sb})$ be the set of the starting nodes of all the loops in Sb. Also, let $\mathrm{SP}_{u\backslash V(\mathrm{Sub}),i}$ be the set of paths from N_u to N_i which go through each node in $V(\mathrm{Sb})$ at least once. To obtain $m_{uv-ij}(C_k)$ systematically, the maximal function in (11) can be decomposed into two maximal functions as in (12) and computed by the algorithm A_1 given below.

$$m_{uv-ij}(C_k) = \max_{\mathrm{Sb} \subseteq \mathrm{SPL}(C_k)} \max_{\substack{L(P_{u^*}) \subseteq \mathrm{Sb} \text{ and} \\ P_{u^*} \in \mathrm{SP}_{u\backslash V(\mathrm{Sub}),i}}} \{h(P_{u^*}) | d(P_{u^*}) < d(P_{uv-ij}) - d(P_{iv})\}. \quad (12)$$

Algorithm A_1:
begin
max := 0;
For all $\mathrm{Sb} \in 2^{\mathrm{SPL}(C_k)}$ **do**
begin
S0. Denote the loops in Sb by $L_{d_1}, L_{d_2}, \cdots, L_{d_n}$, where $n = |\mathrm{Sb}|$.
S1. Sort L_{d_i}, $1 \le i \le n$, with the key $d(L_{d_i})/h(L_{d_i})$ in ascending order and check if the values of hop functions of loops in the resulting sequence are in descending order, i.e., $\forall\, L_{d_i}$, $L_{d_j} \in \mathrm{Sb}$, $d(L_{d_i})/h(L_{d_i}) \le d(L_{d_j})/h(L_{d_j})$ iff $h(L_{d_i}) \ge h(L_{d_j})$.
S2. **If** the test result of $S1$ is true **then**
for all $P_{u^*} \in \mathrm{SP}_{u/V(\mathrm{Sub}),i}$ such that $d(P_{u^*}) < d(P_{uv-ij}) - d(P_{iv}) - \sum_{i=1}^{n} d(L_{d_i})$ **do**
begin
Find an n-tuple $[n_1, n_2, \cdots, n_n]$ which maximizes $\sum_{i=1}^{n} n_i h(L_{d_i})$ subject to $\sum_{i=1}^{n} n_i d(L_{d_i}) \le d(P_{uv-ij}) - d(P_{iv}) - d(P_{u^*})$.
If max $< \sum_{i=1}^{n} n_i h(L_{d_i}) + h(P_{u^*})$ **then**
max := $\sum_{i=1}^{n} n_i h(L_{d_i}) + h(P_{u^*})$;
end /* inner maximal function of (12) */
end /* outer maximal function of (12) */
$m_{uv-ij}(C_k)$:= max;
end

Note that the number of subsets of $\mathrm{SPL}(C_k)$ is 2^m, where $m = |\mathrm{SPL}(C_k)|$, and different subsets (Sb's) will be associated with different $\mathrm{SP}_{u\backslash V(\mathrm{Sub}),i}$'s. We have to determine the inner maximal function in (12) for each Sb before applying the outer maximal function. S1 in A_1 shows that some subsets of $\mathrm{SPL}(C_k)$ that definitely do not lead to the solution can be skipped. Since the number of subsets with cardinality n is C_n^m and the number of possible permutations of loops in such a subset is $n!$, the average probability for an arbitrary Sb with $|\mathrm{Sb}| = n$ to pass the test in S1 is $1/n!$. Thus, the expected number of times S2 is to be executed is $\sum_{n=0}^{m} C_n^m / n!$. This is significantly less than $\sum_{n=0}^{m} C_n^m = 2^m$, a brute-force enumeration.

Once the network is given, using the above algorithm we can obtain $m_{uv-ij}(C_k)$ for all N_u, $N_v \in V(N)$ and $L_{ij} \in E(N)$, and then $\mathrm{RT}(C_k)$ from (7) and (8). $\mathrm{RC}(C_k)$ is determined by (6). Since the required order of routing strategy for each node can be obtained by Corollary 1.1, the number of possible configurations under the constraint of strategy compatibility can thus be determined. Once a design objective function $F(C_k) = f(\mathrm{RC}(C_k), \mathrm{RT}(C_k))$ is decided, the optimal configuration can be determined by evaluating each possible configuration.

Note that instead of exhaustively evaluating all possible strategy vectors, we can skip the evaluation of the configurations in either of the following two cases: 1) there is a node assigned a routing strategy of an order higher than it requires, i.e., $OV[i] > O_i^*$ and $OV[i] \ge OV[j]\ \forall N_j \in A_i$, where $OV[i]$ denotes the order of routing strategy for $N_i\ \forall N_i \in V(N)$, and 2) the difference in the order of strategy between any two adjacent nodes is greater than one. Clearly, the knowledge of the minimal order for loop-free routing and the strategy compatibility reduces the number of configurations to be evaluated significantly. Configurations of the example network in Fig. 2 are evaluated in the following sequence.

[3, 2, 3, 2, 1, 1, 1, 2] (evaluated)

[3, 2, 3, 2, 1, 1, 1, 1] ($OV[3] - OV[8] > 1 \Rightarrow$ skipped)

[3, 2, 3, 2, 1, 1, 0, 2] ($OV[8] - OV[7] > 1 \Rightarrow$ skipped)

[3, 2, 3, 2, 1, 1, 0, 1] ($OV[3] - OV[8] > 1 \Rightarrow$ skipped)

[3, 2, 3, 2, 1, 0, 1, 2] (evaluated)

[3, 2, 3, 2, 1, 0, 1, 1] ($OV[3] - OV[8] > 1 \Rightarrow$ skipped)

$\vdots$

[3, 2, 2, 2, 1, 1, 1, 2] ($OV[8] > O_8^*$ and

($OV[8] \geq OV[3], OV[7] \Rightarrow$ skipped)

[3, 2, 2, 2, 1, 1, 1, 1] (evaluated)

$\vdots$

[0, 0, 0, 0, 0, 0, 0, 0] (evaluated)

B. Complexity of the Optimization Algorithm

For each configuration C_k, the number of $m_{uv-ij}(C_k)$'s needed to obtain $F(C_k)$ is $p(p-1)q$, where $p = |V(N)|$ and $q = |E(N)|$. That is, A_1 has to be executed $p(p-1)q$ times for each configuration. Therefore, the number of configurations to be evaluated is a dominating factor in the execution time of the whole procedure.

To estimate the number of configurations to be evaluated, consider the following interesting combinatorial problem first. Given a labeled graph, if we want to assign each node with an integer chosen from $I_m \equiv \{0, 1, 2, \cdots, m\}$ in such a way that the difference between the numbers assigned to any two adjacent nodes must be less than or equal to one, how many assignments are there? Notice that if the labeled graph is $G = (V(N), E(N))$, then the answer to the above problem is exactly the number of possible configurations in the case of $O_i^* = m \, \forall N_i \in V(N)$.

Define a *distribution vector* (*D-vector*) D_i for each node N_i, the kth component of which, denoted by $D_i(k)$, represents the number of times N_i is assigned the value $k \in I_m$. Fig. 3 is an illustration of this idea with $m = 3$.

Now, consider the case when one more node N_g is to be attached to a node N_d in a given graph. The D-vectors of N_g and N_d in the resulting graph are denoted by D_g and D_d, respectively, while the D-vector of N_d in the original graph is represented by D_d'. Then, the relationship between these quantities can be determined by the following lemma.

Note that Lemma 1 is a special case of Lemma 2 in [13].

Lemma 1:

$$\text{a)} \quad \begin{cases} D_g(0) = D_d'(0) + D_d'(1). \\ D_g(i) = \sum_{j=i-1}^{j=i+1} D_d'(j), \qquad 1 \leq i \leq m-1 \\ D_g(m) = D_d'(m-1) + D_d'(m). \end{cases}$$

$$\text{b)} \quad \begin{cases} D_d(0) = D_d'(1)^*2. \\ D_d(i) = D_d'(i)^*3, \qquad 1 \leq i \leq m-1 \\ D_d(m) = D_d'(m)^*2. \end{cases}$$

Proof: a) Suppose the node N_g is assigned 0. Then, it can be attached to N_d only when N_d was originally assigned 0 or 1. Thus, $D_g(0) = D_d'(0) + D_d'(1)$. Similarly, we can get the other two equations.

b) When N_d is assigned 0, possible numbers assigned to N_g are 0 and 1, each of which corresponds to a different assignment in the resulting graph. Thus, $D_d(0) = D_d'(0)^*2$. The other two equations in b) can be obtained similarly. Q.E.D.

To demonstrate how Lemma 1 can be used, consider the three cases shown in Fig. 4, where $m = 3$. The D-vectors of attaching and attached nodes can be easily obtained as follows.

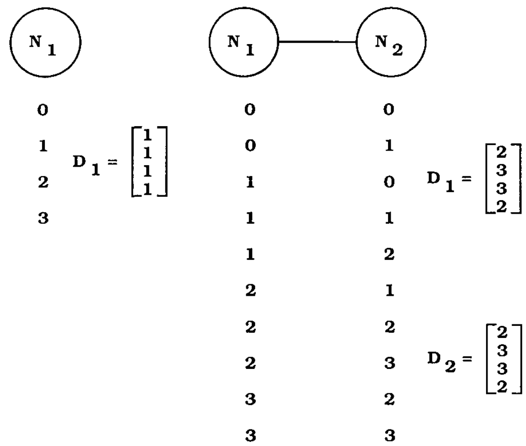

Fig. 3. Illustration of D-vectors.

$$\text{i)} \quad D_d' = \begin{bmatrix} 2 \\ 3 \\ 3 \\ 2 \end{bmatrix} \text{ from Fig. 3} \Rightarrow D_d = \begin{bmatrix} 4 \\ 9 \\ 9 \\ 4 \end{bmatrix}$$

$$\text{and } D_g = \begin{bmatrix} 5 \\ 8 \\ 8 \\ 5 \end{bmatrix}$$

$$\text{ii)} \quad D_d' = \begin{bmatrix} 5 \\ 8 \\ 8 \\ 5 \end{bmatrix} \text{ from } D_g \text{ of i)} \Rightarrow D_d = \begin{bmatrix} 10 \\ 24 \\ 24 \\ 10 \end{bmatrix}$$

$$\text{and } D_g = \begin{bmatrix} 13 \\ 21 \\ 21 \\ 13 \end{bmatrix}$$

$$\text{iii)} \quad D_d' = \begin{bmatrix} 4 \\ 9 \\ 9 \\ 4 \end{bmatrix} \text{ from } D_d \text{ of i)} \Rightarrow D_d = \begin{bmatrix} 8 \\ 27 \\ 27 \\ 8 \end{bmatrix}$$

$$\text{and } D_g = \begin{bmatrix} 12 \\ 22 \\ 22 \\ 12 \end{bmatrix}.$$

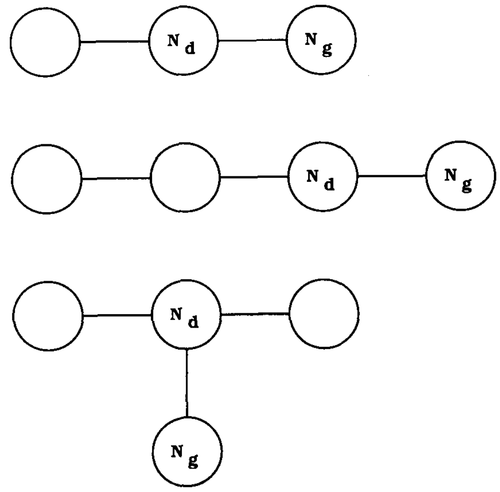

Fig. 4. Three example cases of adding a node N_g to a node N_d.

Note that for any node in the graph the sum of entries in its D-vector represents the number of assignments. It is also easy to see that $\sum_{i=0}^{m} D_d(i) = \sum_{i=0}^{m} D_g(i)$. Using D-vectors, we can determine the bounds of the number of assignments by the theorem below.

Theorem 2: The number of assignments from the integer set I_m to any connected graph with p nodes, subject to the constraint that the difference between the numbers assigned to two adjacent nodes must be less than or equal to one, lies within the interval $[m(2^p - 1) + 1, 2^p + 3^{p-1}(m-1)]$, where $m \geq 1$.

Proof: Obviously, the number of acceptable assignments for any connected graph is always less than or equal to that of its spanning tree. That is, the upper bound is attained by a tree structure. Now, we want to prove that the maximum is attained when the tree is a star structure, and then the upper bound follows from Lemma 1.

Since $\sum_{k=0}^{m} D_i(k)$ is the same for every N_i in a tree T, let $N(T, m) \equiv \sum_{k=0}^{m} D_i(k)$. For convenience, a tree T with p nodes is said to satisfy the *C-property*, if $N(T, m) \leq 2^p + 3^{p-1}(m-1)$. Clearly, the C-property is satisfied by every tree with p nodes when $p \leq 3$. Consider the case when one more node N_g is attached to a tree T with the C-property. Let D_d and D'_d denote, respectively, the D-vectors of N_d in the resulting and the original trees. Note that from Lemma 1 we have $D_d(i) \leq 3^{p-1}$ $\forall N_i \in V(N)$, and thus $\sum_{i=1}^{m-1} D_d(i) \leq 3^{p-1}(m-1)$. Since T satisfies the C-property, we get $\sum_{i=1}^{m-1} D_d(i) + D_d(0) + D_d(m) \leq 3^{p-1}(m-1) + 2^p$ which, by b) of Lemma 1, leads to $\sum_{i=0}^{m} D'_d(i) = (\sum_{i=1}^{m-1} D_d(i))^*3 + (D_d(0) + D_d(m))^*2 \leq 3^p(m-1) + 2^{p+1}$. This means that the resulting tree also satisfies the C-property, and the upper bound thus follows by induction.

Consider the lower bound. Since the complete graph K_p with p nodes possesses the maximal number of edges among all the graphs with p nodes, K_p attains the minimal number of assignments. Note that there are at most two distinct numbers which may occur in each assignment to K_p, and their difference must be less than or equal to one. There are 2^p ways to assign the numbers in the pair $\{j, j-1\}$ to p nodes, where $1 \leq j \leq m$. Assignment of the same number to every node, say j, occurs both in the case of $\{j+1, j\}$ and $\{j, j-1\}$, where $1 \leq j \leq m-1$. Thus, the total number of assignments is obtained by adding up the number of assignments from each pair $\{j, j-1\}$, where $1 \leq j \leq m$, to p nodes and subtracting double counts. Then, we get $2^p m - (m-1) = (2^p - 1)m + 1$ for the lower bound. Q.E.D.

By Theorem 2, for a given network with p nodes the number of configurations to be evaluated must be within the interval $[n(2^p - 1) + 1, 2^p + 3^{p-1}(m-1)]$, where $n = \min_{1 \leq i \leq p} \{O_i^*\}$ and $m = \max_{1 \leq i \leq p} \{O_i^*\}$. Note that due to the special nature of our problem, for a given topology a network with a higher average order of loop-free strategy does not always possess more configurations to be evaluated than the one with a lower average order of loop-free strategy. An example is shown in Fig. 5, where the network A has a higher average order of loop-free strategy than the network B, but B has more configurations to be evaluated. This is the very reason why $\max_{1 \leq i \leq p} \{O_i^*\}$ and $\min_{1 \leq i \leq p} \{O_i^*\}$ have to be used for upper and lower bounds, respectively.

Example 2: Consider the example network in Fig. 6. Following the same procedure shown in the part a) of Example 1, we can obtain [1, 1, 1, 1] as the minimal order vector of loop-free routing strategies. Clearly, there are $2^4 = 16$ possible configurations in this network.

As discussed in Section IV-A, the operational overhead required per second for the nth order routing strategy $R_c(n)$ can be assumed to have the form of $an + b$, where the values of a and b depend on the node computer at hand. For the sake of numerical demonstration, let $a = 2.1$, $b = 1.2$, and C_α, C_β, C_γ be the configurations with strategy vectors [1, 0, 0, 0], [1, 1, 0, 0], and [1, 1, 0, 1], respectively.

a) $\mathrm{RC}(C_\alpha)$, $\mathrm{RC}(C_\beta)$, and $\mathrm{RC}(C_\gamma)$ are obtained from (6) as follows.

$$\mathrm{RC}(C_\alpha) = R_c(1) + 3R_c(0) = 6.9$$

$$\mathrm{RC}(C_\beta) = 2R_c(1) + 2R_c(0) = 9$$

$$\mathrm{RC}(C_\gamma) = 3R_c(1) + R_c(0) = 11.1.$$

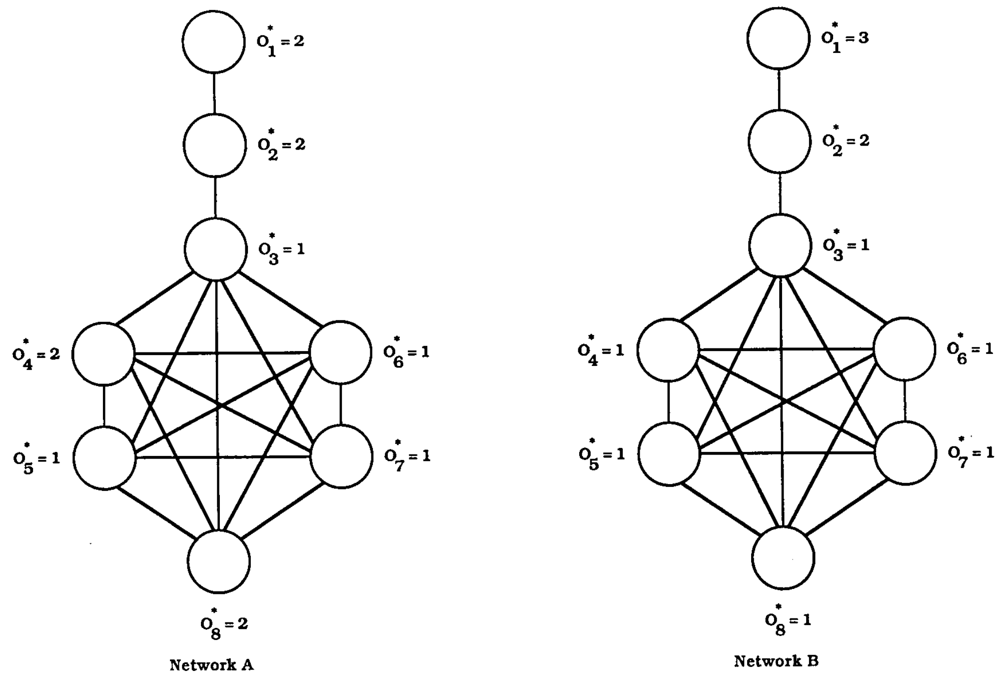

Fig. 5. Two comparative example networks where A has a higher average order of looping-free strategy, whereas B has more configurations to be evaluated.

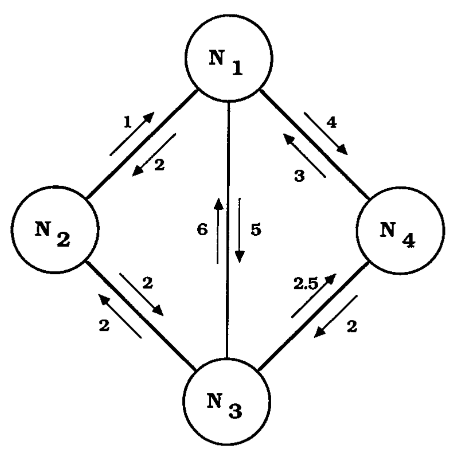

Fig. 6. A network for Example 2.

b) $\mathrm{RT}(C_\alpha)$, $\mathrm{RT}(C_\beta)$, and $\mathrm{RT}(C_\gamma)$ can be determined as follows.

i) With configuration C_α whose strategy vector is [1, 0, 0, 0], we find

$$\mathrm{SPL}(C_\alpha) = \{(N_2, N_3, N_2), (N_3, N_4, N_3), (N_3, N_2, N_3), (N_1, N_2, N_1)\}.$$

Using A_1, we can obtain $m_{uv-ij}(C_\alpha)$ as follows: $m_{14-14}(C_\alpha) = 2$, $m_{21-21}(C_\alpha) = 2$, $m_{31-21}(C_\alpha) = 1$, $m_{13-23}(C_\alpha) = 1$, $m_{32-32}(C_\alpha) = 2$, $m_{42-32}(C_\alpha) = 1$, $m_{24-34}(C_\alpha) = 1$, $m_{34-34}(C_\alpha) = 2$, and $m_{uv-ij}(C_\alpha) = 0$ elsewhere.

Then, from (7) we get

$$\left.\begin{cases} \mathrm{Rt}(L_{14};\, C_\alpha) = \dfrac{1}{6} \\ \mathrm{Rt}(L_{21};\, C_\alpha) = \dfrac{1}{4} \\ \mathrm{Rt}(L_{23};\, C_\alpha) = \dfrac{1}{12} \\ \mathrm{Rt}(L_{32};\, C_\alpha) = \dfrac{1}{4} \\ \mathrm{Rt}(L_{34};\, C_\alpha) = \dfrac{1}{4} \end{cases}\right. \Rightarrow \mathrm{RT}(C_\alpha) = \left(\frac{\frac{1}{6} + \frac{1}{4} + \frac{1}{12} + \frac{1}{4} + \frac{1}{4}}{10}\right) = \frac{1}{10}.$$

ii) With the configuration C_β whose strategy vector is [1, 1, 0, 0], we get

$$\mathrm{SPL}(C_\beta) = \{(N_2, N_3, N_2), (N_3, N_4, N_3)\}.$$

From A_1, we can obtain $m_{uv-ij}(C_\beta)$ as follows: $m_{21-21}(C_\beta) = 2$, $m_{31-21}(C_\beta) = 1$, $m_{13-23}(C_\beta) = 1$, $m_{32-32}(C_\beta) = 2$, $m_{42-32}(C_\beta) = 1$, $m_{24-34}(C_\beta) = 1$, and $m_{uv-ij}(C_\beta) = 0$ elsewhere.

Then, by (7), we get $\mathrm{Rt}(L_{21};\, C_\beta) = 1/4$, $\mathrm{Rt}(L_{23};\, C_\beta) = 1/12$, $\mathrm{Rt}(L_{32};\, C_\beta) = 1/4$, $\mathrm{Rt}(L_{34};\, C_\beta) = 1/12$ and $\mathrm{RT}(C_\beta) = 1/15$.

iii) With configuration C_γ, whose strategy vector is [1, 1, 0, 1], following the same procedure, we obtain $\mathrm{SPL}(C_\gamma) = \{(N_2, N_3, N_2)\}$, and then $\mathrm{RT}(C_\gamma) = 1/20$.

c) Suppose our design objective is to minimize the function $g(C_k) = \mathrm{RC}(C_k) + \lambda \mathrm{RT}(C_k)$, where λ is the weighting factor between the network adaptability and operational overhead in handling routing messages. Note that $F(C_k) = 1/g(C_k)$ in this case. If we choose λ to be 120, then we get[1]

$$g(C_\alpha) = 6.9 + \frac{120}{10} = 18.9,$$

$$g(C_\beta) = 9 + \frac{120}{15} = 17,$$

$$g(C_\gamma) = 11.1 + \frac{120}{20} = 17.1.$$

When the above objective function is used, the configuration with a strategy vector [1, 1, 0, 0] is better than those with strategy vectors [1, 0, 0, 0] and [1, 1, 0, 1]. Therefore, from this procedure we can determine the optimal configuration of this network.

C. Remarks

Using the procedure discussed thus far, one can determine the optimal configuration from a given network topology and its link delays. The minimal order routing strategy for each node can be used to indicate how to construct a routing message for the node in order to avoid looping. It is worth mentioning that the order of loop-free routing strategy for each node is determined from the number of links on a certain loop around that node, and may vary if link delays in the vicinity of the node change drastically within a short time period. A sudden, drastic change in link delays may force some nodes to alter their optimal paths. In such a case, new minimal order routing strategies for these nodes must be derived. This usually introduces significant overheads, thus making it practically unacceptable.[2]

However, in light of the derivation of Theorem 1, it can be verified that a higher order loop is less likely to occur, since the delay of the higher order loop is unlikely to be less than that of a second optimal path. Moreover, as we formulated in [4] and illustrated in Tables I, II, and III, recovery from a link/node failure is sped up significantly when the order of routing strategy is increased; this is true even if the order of routing strategy is increased not so high as that derived from Corollary 1.1. Considering the above observations, one can determine the minimal order of routing strategy off-line, incorporate it into each node's routing strategy, and ignore small on-line changes in link delays. This will remove the necessity of on-line recalculation of minimal order routing strategies while allowing for acceptably fast recovery from node/link failures.

V. Conclusion

In this paper, we have developed a minimal order loop-free routing strategy. Unlike most conventional methods in which the same routing strategy is applied indiscriminately to all nodes in the network, *each node* under the proposed strategy adopts its optimal routing strategy. We have not only developed the formulas to determine the minimal order of the routing strategy for each node to eliminate looping completely, but also proposed a systematic procedure to strike a compromise between the operational overhead and network adaptability. The number of configurations to be evaluated is rigorously analyzed with a combinatorial approach.

Note that the order of the optimal routing strategy for each node can be determined off-line from a given network and incorporated into each node before the network executes certain missions if the propagation delay is the main factor of link delay. The network is thus made to attain the maximal adaptability in case of link/node failures during such missions. However, in the case when reducing the operational overhead is the essential issue and infrequent looping is tolerable, the use and implementation of a high-order routing strategy may have to be justified. This can be accomplished by the selection of an appropriate design objective function addressed in Section IV-A. In our discussion, we assumed 1) a uniform traffic density between every pair of nodes in the network and 2) an equal probability of failure in every link/node. Both assumptions can be relaxed by changing the corresponding formulas to include appropriate stochastic aspects. This will make the problem more realistic and complicated.

Appendix

List of Symbols

$V(N)$:	Set of computer nodes in a network N.
$E(N)$:	Set of computer links in a network N.
DL_{ij}:	Delay of a direct link L_{ij} from node N_i to N_j.
A_i:	Set of nodes adjacent to N_i, i.e., $N_j \in A_i$ if a direct link L_{ij} exists.
SP_{ij}:	Set of all paths from N_i to N_j.
SL_{ii}:	Set of loops starting and ending at N_i.
SP:	$\mathrm{SP} = \cup_{N_i, N_j \in V(N)} \mathrm{SP}_{ij}$.
$d(P_i)$:	Summation of all link delays in a path P_i.
$h(P_i)$:	The number of links in a path P_i.
P_{ij}:	The path with the shortest delay (i.e., the *optimal path*) in SP_{ij}.
P_{ij-uv}:	The shrotest delay path in the set $\mathrm{SP}_{ij} - \{L_{uv}\}$.
$H_k(P_i)$:	The set of the first k nodes in the ordered sequence representation of a path $P_i \in \mathrm{SP}$.
$\mathrm{NT}^k_{i\backslash jd}$:	The information kept in the network delay table of N_i about the shortest path from N_i via $N_j \in A_i$ to N_d under the kth order routing strategy.
$\mathrm{RM}^k_{i\leftarrow jd}$:	The routing message sent from N_j to N_i about the routing from N_j to N_d under the kth order routing strategy.
$\mathrm{R}_{i\leftarrow k,j}$:	The required order of routing message sent from $N_k \in A_i$ to N_i to avoid all potential looping when L_{ij} became faulty.
$R_{i\leftarrow k}$:	The required order of routing message sent

[1] This choice is arbitrary and does not affect our method but yields interesting solutions.

[2] This fact was pointed out by one anonymous referee.

from $N_k \in A_i$ to N_i to avoid all potential looping.

O_i^*: The minimal order of routing strategy required for N_i to avoid all potential looping.

O_i^k: The order of the routing strategy adopted by N_i when the configuration is C_k.

$R_c(k)$: The cost required per second for a node adopting the kth order strategy to generate and process a routing message.

$\mathrm{RC}(C_k)$: The operational overhead per second induced under configuration C_k.

$R_t(L_{ij}; C_k)$: The expected number of time intervals required for an arbitrary node to obtain a new nonfaulty optimal path to any other node when L_{ij} became faulty.

$\mathrm{RT}(C_k)$: The expected number of time intervals for a path to recover from an arbitrary link failure under the configuration C_k.

$m_{uv-ij}(C_k)$: The number of time intervals required under the configuration C_k for N_u to obtain a new nonfaulty optimal path to N_v when L_{ij} became faulty.

$\mathrm{SPL}(N_i; C_k)$: Set of loops induced by the insufficient order of routing strategy of N_i in the configuration C_k.

$\mathrm{SPL}(C_k)$: Set of all potential loops under the configuration C_k.

$L(P_{i^*})$: Set of loops in the path P_{i^*}.

D_i: The distribution vector (D-vector) of node N_i.

I_m: Set of integers $\{0, 1, 2, \cdots, m\}$.

References

[1] J. M. McQuillan, "Routing algorithms for computer networks—A survery," in *Proc.* 1977 *Nat. Telecommun. Conf.*, Dec. 1977, p. 28.

[2] M. Schwartz and T. E. Stern, "Routing technique used in computer communication networks," *IEEE Trans. Commun.*, vol. COM-28, no. 4, pp. 539–552, Apr. 1980.

[3] J. M. McQuillan and D. C. Walden, "The ARPA network design decisions," *Comput. Networks*, vol. 1, no. 5, pp. 243–289, Aug. 1977.

[4] K. G. Shin and M.-S. Chen, "Performance analysis of distributed routing strategies free of ping-pong-type looping," *IEEE Trans. Comput.*, vol. C-36, no. 2, pp. 129–137, Feb. 1987.

[5] P. M. Merlin and A. Segall, "A failsafe distributed routing protocol," *IEEE Trans. Commun.*, vol. COM-27, no. 9, pp. 1280–1287, Sept. 1979.

[6] W. D. Tajibnapis, "A correctness proof of a topology information maintenance protocol for distributed computer networks," *Commun. ACM*, vol. 20, pp. 477–485, 1977.

[7] J. M. Jaffe and F. H. Moss, "A responsive distributed routing algorithm for computer networks," *IEEE Trans. Commun.*, vol. COM-30, no. 7, pp. 1758–1762, July 1982.

[8] J. M. McQuillan, I. Richer, and E. C. Rosen, "The new routing algorithm for the ARPANET," *IEEE Trans. Commun.*, vol. COM-28, no. 5, pp. 711–719, May 1980.

[9] T. Cegrell, "A routing procedure for the TIDAS message-switching network," *IEEE Trans. Commun.*, vol. COM-23, no. 6, pp. 575–585, June 1975.

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

[11] L. Kleinrock and H. Opderbeck, "Throughput in the ARPANET-protocols and measurement," *IEEE Trans. Commun.*, vol. COM-25, no. 1, pp. 367–376, Jan. 1977.

[12] M. J. Johnson, "Updating routing tables after resource failure in a distributed computer network," *Networks*, vol. 14, pp. 379–391, 1984.

[13] K. G. Shin and M.-S. Chen, "On the number of acceptable task assignments in distributed computing systems," *IEEE Trans. Comput.*, vol. 39, no. 1, pp. 99–110, Jan. 1990.

Kang G. Shin (S'75–M'78–SM'83), for a photograph and biography, see the January 1990 issue of this Transactions, p. 18.

Ming-Syan Chen (S'87–M'88), for a photograph and biography, see the January 1990 issue of this Transactions, p. 18.