A Routing Protocol for Finding Two Node–Disjoint Paths in Computer Networks

Kenji Ishida

Dept. of Information Science
Hiroshima Prefectural University
Shobara-shi, Hiroshima 727 Japan

Yoshiaki Kakuda, Tohru Kikuno

Dept. of Information and Computer Science
Osaka University
Toyonaka-shi, Osaka 560 Japan

Abstract

In this paper, we present a routing protocol for finding two node–disjoint paths between each pair of nodes in a computer network. In the proposed protocol, each node in the network has the same procedure, which is driven by local information with respect to the network topology such as an adjacent node on a spanning tree in the network. Thus, the execution of the protocol can continue after changes of the network topology and load.

The concept of spanning tree–based kernel construction is introduced to synchronize procedures under the distributed control of the protocol. The routing scheme based on the protocol possesses the enhanced capabilities of alternate routes and load splitting, which cope with failures and load variations in the network. Furthermore, even if topology changes occur which damage the obtained disjoint paths, the paths themselves can be updated efficiently.

1 Introduction

High speed computer networks such as ATM networks require end–to–end processing than hop–by–hop processing [2], [7]. It is because communication channel can deliver messages at a higher rate relative to individual node processing. However, if a failure occurs in the communication channel, then a lot of messages being delivered are lost and they must be retransmitted from the source node. In order to enhance reliability of the routing in high speed computer networks, reliable routing protocols for recovery are indispensable.

A number of distributed routing protocols have been developed [8], [9], and they are classified into the single path type and the multiple path type. The single path protocols include shortest weighted path protocols [3], [9], shortest unweighted path protocols [12], minimum spanning tree protocols [9], and selective broadcast protocols [4]. However, the single path protocols turn out not to be optimum over a short time interval even if the average network delay is to be minimized over long time intervals, and most of the protocols cannot recover quickly from failures.

In order to overcome these difficulties, multiple path protocols arise, in which messages could continue to be transmitted even when failures occur on the paths and load may be split over several paths. Some multiple path protocols are proposed to enhance reliability or throughput [1], [5], [8], [11]. However, such multiple–paths are shared with each other in some nodes or channels. Only node–disjoint path protocols enable the most reliable routing in high speed computer networks

In this paper, we propose a distributed routing protocol for finding two node–disjoint paths between each pair of nodes in a computer network. The routing scheme based on the protocol ensures the following three points: (1) Messages generated at any node can always reach their destination nodes. (2) Load can be effectively split into two node–disjoint paths. (3) The obtained disjoint paths can be easily updated in cases of failures by a distributed control.

The paper is organized as follows. In Section 2 the problem of finding two node–disjoint paths, called problem TDP, is formalized. In Section 3 a concept of spanning tree–based kernel construction for solving Problem TDP is introduced and in Section 4 a distributed protocol, based on the concept, for solving Problem TDP is presented. In Section 5 an example of Phase 2 in the distributed protocol is demonstrated. Section 6 shows concluding remarks.

2 System Model

Let an undirected graph $G = (V, E)$ represent a computer network, where a set of node V and a set of edges E correspond to switching computers and communication links, respectively. Let $r(v, v') = v_1, v_2, \cdots, v_k$ where $v = v_1$ and $v' = v_k$ denote a simple path between nodes v and v'. If $v = v'$ then it is called a simple circuit. If there are r simple paths between v and v', then these paths are distinguished by denoted as $r_s(v, v') = v_1^s, v_2^s, \cdots, v_k^s$ $(1 \leq s \leq r)$. We say that two simple paths $r_s(v, v')$ and $r_t(v, v')$ $(s \neq t)$ are node–disjoint if these paths do not have common nodes except node v and v'. We also say that an undirected graph $G = (V, E)$ is biconnected if there exists a node–disjoint simple paths between every pair of nodes in G. From now on, we may denote an undirected graph as a graph, a simple path as a path and node–disjoint as disjoint.

Now, we formalize Problem TDP as follows:

0-8186-7216-1/95 $04.00 © 1995 IEEE

[Problem TDP] Given a biconnected undirected graph $G = (V, E)$ and an arbitrary node $v_d \in V$ (we call it a destination node), find two simple paths $r_1(v, v_d)$ and $r_2(v, v_d)$ between any node $v (\in V, v \neq v_d)$ and v_d, which satisfy the following two conditions.

Condition D: $r_1(v, v_d)$ and $r_2(v, v_d)$ are node-disjoint.

Condition S: Let $r_1(v, v_d) = v_1^1, v_2^1, \cdots, v_k^1$ where $v_1^1 = v$ and $v_k^1 = v_d$. Then, given an arbitrary node v_j^1 $(1 < j < k)$ on $r_1(v, v_d)$ the terminal subpath $v_j^1, v_{j+1}^1, \cdots, v_k^1$ is $r_1(v_j^1, v_d)$. The similar condition holds for $r_2(v, v_d)$. □

We call $r_1(v, v_d) = v_1^1, v_2^1, \cdots, v_k^1$ a forward path of v and $r_2(v, v_d) = v_1^2, v_2^2, \cdots, v_k^2$ a backward path of v. Furthermore, for $r_1(v, v_d)$ and $r_2(v, v_d)$ we call v_2^1 a forward node of v and v_2^2 a backward node of v. If Problem TDP is solved for any $v_d \in V$, we have two node-disjoint paths between every pair of nodes in an undirected graph. In routing, a pair of the forward and backward nodes of v is stored in a routing table RT(v). Condition S is included mainly in consideration for the minimization of the size of routing table [6].

[Example 1] We will show an example of Problem TDP. We are given an undirected graph $G_1 = (V_1, E_1)$ and a node $v_1 \in V_1$ as v_d in Fig.1. The following forward and backward paths of any node v $(\in V_1, v \neq v_1)$ satisfy Condition D and Condition S of Problem TDP. The forward paths are $r_1(v_2, v_1) = v_2, v_5, v_4, v_1$, r_1 $(v_3, v_1) = v_3, v_4, v_1$, r_1 $(v_4, v_1) = v_4, v_1$, r_1 $(v_5, v_1) = v_5, v_4, v_1$, r_1 $(v_6, v_1) = v_6, v_7, v_4, v_1$, r_1 $(v_7, v_1) = v_7, v_4, v_1$, r_1 $(v_8, v_1) = v_8, v_4, v_1$, and r_1 $(v_9, v_1) = v_9, v_8, v_4, v_1$. The backward paths are r_2 $(v_2, v_1) = v_2, v_1$, r_2 $(v_3, v_1) = v_3, v_5, v_2, v_1$, r_2 $(v_4, v_1) = v_4, v_5, v_2, v_1$, r_2 $(v_5, v_1) = v_5, v_2, v_1$, r_2 $(v_6, v_1) = v_6, v_2, v_1$, r_2 $(v_7, v_1) = v_7, v_6, v_2, v_1$, r_2 $(v_8, v_1) = v_8, v_9, v_6, v_2, v_1$, and r_2 $(v_9, v_1) = v_9, v_6, v_2, v_1$. These paths are stored in a routing table RT(v_i) for each v_i. In Fig.1, F and B denote a forward and backward nodes, respectively. □

3 Kernel Construction

We introduce a recursive procedure called a kernel construction which provides a basis for a solution to Problem TDP under centralized control. The original concept of kernel construction is presented in [6]. Here, we extend this concept to a spanning tree-based kernel construction.

A spanning tree $T = (V_T, E_T)$ of an undirected graph $G = (V, E)$ is a connected subgraph of G, which includes all nodes $v \in V$ and does not have any circuit. A co-tree $\bar{E}_t$ is a set of edges which does not include edges in E_T, that is, $E - E_T$. Then, there always exists a circuit which consists of exactly one edge e_i in $\bar{E}_t$ and the other all edges in E_T. We call the circuit a fundamental circuit for e_i. Let it be denoted by c_i. Let the set of edges c_i be denoted by $C = \{c_i | 1 \leq i \leq |\bar{E}_t|\}$. For convenience of notations we assume $c_i = (V_{c_i}, E_{c_i})$ may denote a subgraph consisting of nodes and edges on a fundamental circuit c_i.

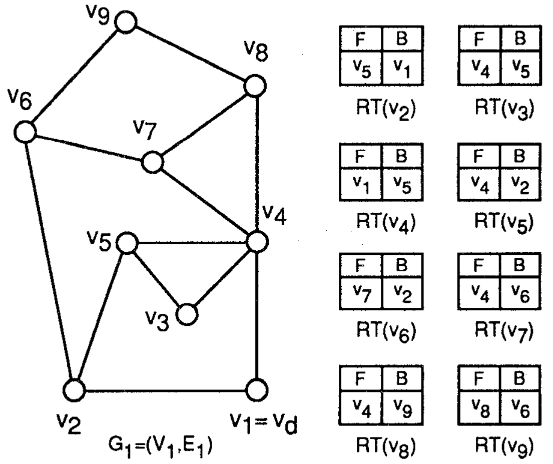


Figure 1: Forward and backward paths.

Now, we define the sequences of kernels denoted by $\tilde{G}_1 = (\tilde{V}_1, \tilde{E}_1)$, $\tilde{G}_2 = (\tilde{V}_2, \tilde{E}_2)$, $\cdots$, $\tilde{G}_m = (\tilde{V}_m, \tilde{E}_m)$. Given an undirected graph $G = (V, E)$, a node $v_d \in V$, and a spanning tree $T = (V_T, E_T)$ of G, the procedure is described recursively as follows.

[Spanning tree-based kernel construction]

Step 1: Select a fundamental circuit including v_d from C (let it be c_1) . Let $L_{v_d} = \{c_1\}$ and $c_1 = (V_{c_1}, E_{c_1})$. Then, construct the first kernel $\tilde{G}_1 = (\tilde{V}_1, \tilde{E}_1)$ where $\tilde{V}_1 = V_{c_1}$ and $\tilde{E}_1 = E_{c_1}$. Let L be $C - \{c_1\}$.

Step i $(2 \leq i \leq |E_T|)$: The following procedure is iterated unless $L = \phi$. We assume that the $(i-1)$-th kernel $\tilde{G}_{i-1} = (\tilde{V}_{i-1}, \tilde{E}_{i-1})$ has been constructed. If L contains a fundamental circuit $c_i = (V_{c_1}, E_{c_1})$ which satisfies the following Condition C, then construct the i-th kernel $\tilde{G}_i = (\tilde{V}_i, \tilde{E}_i)$ where $\tilde{V}_i = \tilde{V}_{i-1} \cup V_{c_i}$ and $\tilde{E}_i = \tilde{E}_{i-1} \cup E_{c_i}$. Let L be $L - \{c_i\}$. If such c_i does not exist, then the kernel construction is terminated.

[Condition C:] c_i is the construction of the following two simple paths $P_i(c_k)$ and $Q_i(c_k)$ (see Fig.2).

P_i $(c_k) = v_{p_1}, v_{p_2}, \cdots, v_{p_s}$ and Q_i $(c_k) = v_{q_1}, v_{q_2}, \cdots, v_{q_t}$, where $v_{p_1}, v_{p_s} \in \tilde{V}_{i-1}$; $v_{p_r} \notin \tilde{V}_{i-1}$ $(2 \leq r \leq s-1)$; $(v_{p_1}, v_{p_2}), (v_{p_2}, v_{p_3}), \cdots, (v_{p_{s-1}}, v_{p_s}) \notin \tilde{E}_{i-1}$; $v_{q_j} \in \tilde{V}_{i-1}$ $(1 \leq j \leq t, t \geq 2)$; $(v_{q_1}, v_{q_2}), (v_{q_2}, v_{q_3}), \cdots, (v_{q_{t-1}}, v_{q_t}) \in \tilde{E}_{i-1}$; $v_{p_1} = v_{q_1}$ and $v_{p_s} = v_{q_t}$.

We call v_{p_1} $(= v_{q_1})$ and v_{p_s} $(= v_{q_t})$ S-node of $\tilde{G}_{i-1}$ and S'-node of $\tilde{G}_{i-1}$, which are denoted by $S(\tilde{G}_{i-1})$ and $S'(\tilde{G}_{i-1})$, respectively. □

[Theorem 1] Given a biconnected undirected graph $G = (V, E)$, a node $v_d \in V$ and a spanning tree $T = (V_T, E_T)$ of G, then the spanning tree-based kernel

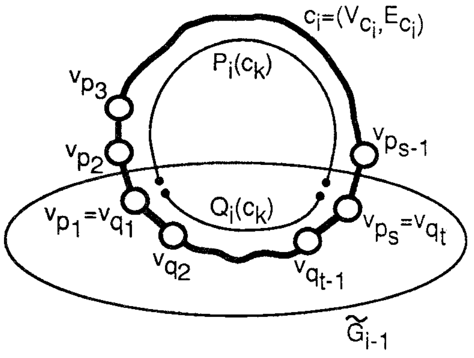


Figure 2: Fundamental circuit satisfying Condition C.

construction can be applied to the step m such that $\tilde{G}_m = G$. □

This theorem can be proved by induction on steps of the spanning tree–based kernel construction. The proof scheme is similar to that of the original kernel construction [6].

Next, based on the above spanning tree–based kernel construction we give a constructive procedure for choosing forward and backward paths of any node on a biconnected undirected graph.

[Path choice procedure]

Step 1: Let $c_1 = v_1(= v_d)$, v_2, $\cdots$, $v_{m_1}(= v_d)$. Then, for any node $v_j(= v)$ on c_i choose $r_1(v, v_d) = v_j, v_{j+1}, \cdots, v_{m_1}$ as a forward path of v.

Step i $(2 \le i \le |\tilde{E}_T|)$: For $\tilde{G}_{i-1} = (\tilde{V}_{i-1}, \tilde{E}_{i-1})$ and $c_i = (V_{c_1}$, if $E_{c_1})$ $V_{c_1} - \tilde{V}_{i-1} = \phi$, then proceed to Step $i+1$. Otherwise, a forward path and backward path of a node $v \in V_{c_1}$ - $\tilde{V}_{i-1}$ are chosen as follows.

Let $c_1 = v_1(= v)$, v_2, $\cdots$, $v_{m_i}(= v)$. We divide c_i into the following three subpaths (see Fig.3).

(1) $r_a(v, v_x) = v\ (= v_1)$, v_2, $\cdots, v_x$ where v_1, $v_2, \cdots, v_{x-1} \in V_{c_1} - \tilde{V}_{i-1}$, $v_x \in \tilde{V}_{i-1}$.

(2) $r_b(v_x, v_y) = v_x$, v_{x+1}, $\cdots, v_y$ where $v_x, v_{x+1}, \cdots$, $v_y \in \tilde{V}_{i-1}$.

(3) $r_c(v, v_y) = v(= v_{m_i})$, v_{m_i-1}, $\cdots$, v_{y+1}, v_y where v_{m_1}, v_{m_i-1}, $\cdots$, $v_{y+1} \in V_{c_1} - \tilde{V}_{i-1}$, $v_y \in \tilde{V}_{i-1}$. □

Suppose that forward paths and backward paths $r_s(v_x, v_d)$ and $r_t(v_y, v_d)$ $(s, t = 1, 2)$ have been chosen before Step $i-1$. If $s = 1$ and $t = 2$, then choose a concatenation of $r_a(v, v_x)$ and $r_s(v_x, v_d)$ as a forward path of v (let it be a-path) and a concatenation of $r_c(v, v_y)$ and $r_t(v_y, v_d)$ as a backward path of v (let it be b-path). If $s = 2$ and $t = 1$, then choose a-path as a backward path of v and b-path as a forward path of v.

[Theorem 2] Problem TDP can be solved by the spanning tree–based kernel construction and the path choice procedure described above (Note that this method is executed under centralized control.) □

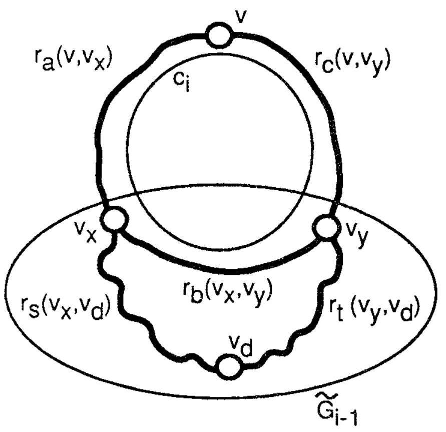


Figure 3: Path choice procedure.

This theorem can be proved by induction on steps of the path choice procedure. The proof scheme is also similar to that of the original kernel construction [6]. In Fig.3 it can be proved that $r_s(v_x, v_d)$ and $r_t(v_y, v_d)$ are node–disjoint.

4 Distributed Protocol

In this section we present a distributed protocol for solving Problem TDP. The protocol is composed of the following two phases.

Phase 1: Given a biconnected undirected graph $G = (V, E)$ and a node $v_d \in V$, construct a spanning tree $T = (V_T, E_T)$ and a set of fundamental circuits $C = \{c_k \mid 1 \le k \le |E - E_T|\}$.

Phase 2: Based on T and C obtained in Phase 1, choose a forward path and backward path of each node v $(\in V, v \neq v_d)$ by exploiting the spanning tree–based kernel construction.

4.1 Phase 1

A distributed protocol of Phase 1 is easily obtained from the distributed protocol proposed in [10] by adding a procedure with respect to the spanning tree and the fundamental circuit. Though details of the protocol are omitted here, we show a result obtained after the execution of the protocol. For example, as shown in Fig.4 a spanning tree T_1 of the graph $G_1 = (V_1, E_1)$ in Fig.1 depicted by bold lines and the associated five fundamental circuits c_1, c_2, c_3, c_4, and c_5 are obtained as local information of each node in G_1.

The result is maintained by each node in an undirected graph as local information. A spanning tree is represented by a set of a father node N_f and a set of son nodes N_s of each node in a graph. Let edges connected to each node be denoted by e_1, e_2, $\cdots$. A fundamental circuit is represented by $E(e_r) = \{c_k \mid c_k$ *contains* $e_r\}$ of each node in a graph defined for all

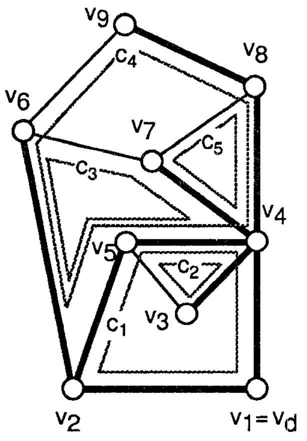


Figure 4: Phase 1.

e_r. In Fig.4 the local information of v_4 is given as follows: with respect to a spanning tree $N_f = \{v_1\}$ and $N_s = \{v_3, v_5, v_7, v_8\}$, and with respect to fundamental circuits $E(e_1) = \{c_1\}$, $E(e_2) = \{c_2\}$, $E(e_3) = \{c_1, c_2, c_3, c_4\}$, $E(e_4) = \{c_3, c_4, c_5\}$, $E(e_5) = \{c_4, c_5\}$, where $e_1 = (v_4, v_1)$, $e_2 = (v_4, v_3)$, $e_3 = (v_4, v_5)$, $e_4 = (v_4, v_7)$, $e_5 = (v_4, v_8)$.

4.2 Phase 2

4.2.1 Extended Kernel Construction

In this subsection we provide an extension of spanning tree–based kernel construction called " extended kernel construction". The extended kernel construction is suitable to describe a distributed protocol of Phase 2 with concurrency. Now, we define the sequences of extended kernels denoted by $\hat{H}_1 = (\hat{V}_1, \hat{E}_1)$, $\hat{H}_2 = (\hat{V}_2, \hat{E}_2)$, $\cdots$, $\hat{H}_n = (\hat{V}_n, \hat{E}_n)$. Given an extended kernel $\hat{H}_j = (\hat{V}_j, \hat{E}_j)$ a set of fundamental circuits for v denoted by $C_v (\subseteq C)$ is defined as follows. Here, let e_a, e_b, e_r denote edges connected to v.

$C_v = \{ c_k \mid c_k$ is a fundamental circuit including v and c_k satisfied the following conditions C1 and C2.

C1: For an edge e_r such that $e_r \notin \hat{E}_j$, $c_k \in E(e_r)$.

C2: For arbitrary edges e_a and e_b such that e_a, $e_b \notin \hat{E}_j$ and $e_a \neq e_b$, $c_k \notin E(e_a) \cap E(e_b)$.

Furthermore, let the set of fundamental circuits satisfying C1 and C2 be C'_v. For each e_r such that $e_r \notin \hat{E}_j$, $\mid C'_v \cap E(e_r)| = 1$. $\}$

Here, for convenience of notations we assume $C_v = (V_{C_v}, E_{C_v})$ may denote a subgraph consisting of nodes and edges on fundamental circuits in C_v. Given an undirected graph $G = (V, E)$, a node $v_d \in V$ and a set of fundamental circuits C, the extended kernel construction is described recursively as follows.

[Extended kernel construction]

Step 1: Choose an arbitrary fundamental circuits (let it be c_k) including v_d from C. Let $C_{v_d} = \{c_k\}$ and $c_k = (V_{c_k}, E_{c_k})$. Then, construct the first extended kernel $\hat{H}_1 = (\hat{V}_1, \hat{E}_1)$ where $\hat{V}_1 = V_{c_k}$ and $\hat{E}_1 = E_{c_k}$. Let L be $C - C_{v_d}$.

Step j ($j \geq 2$): The following procedure is iterated until $L = \phi$. We assume that the $(j-1)$–th extended kernel $\hat{H}_{j-1} = (\hat{V}_{j-1}, \hat{E}_{j-1})$ has been constructed. Then, choose a $C_v = V_{C_v}, E_{C_v})$ such that $C_v \neq \phi$ for a node $v \in \hat{V}_{j-1}$. Construct the j–th extended kernel $\hat{H}_j = (\hat{V}_j, \hat{E}_j)$ where $\hat{V}_j = \hat{V}_{j-1} \cup V_{C_v}$ and $\hat{E}_j = \hat{E}_{j-1} \cup E_{C_v}$. Let L be $L - C_{v_d}$.

[Example 2] We show an example of the extended kernel construction in Fig.4. After Phase 1 we get $C = \{ c_1, c_2, c_3, c_4, c_5 \}$. In Step 1 if $C_{V_1} = \{c_1\}$ and $c_1 = (V_{c_1}, E_{c_1})$ are chosen, then we construct $\hat{H}_1 = (\hat{V}_1, \hat{E}_1)$ where $\hat{V}_1 = V_{c_1}$ and $\hat{E}_1 = E_{c_1}$ and get $L = \{c_2, c_3, c_4, c_5\}$. In Step 2 if $C_{v_4} = \{c_2, c_3, c_4\}$ is chosen (let $C_{v_4} = (V_{C_{v_4}}, E_{C_{v_4}})$), then we construct $\hat{H}_2 = (\hat{V}_2, \hat{E}_2)$ where $\hat{V}_2 = \hat{V}_1 \cup V_{C_{v_4}}$ and $\hat{E}_2 = \hat{E}_1 \cup E_{C_{v_4}}$ and get $L = \{c_5\}$. In Step 3 if $C_{V_8} = \{c_5\}$ is chosen then we construct $\hat{H}_3 = (\hat{V}_3, \hat{E}_3)$, which is equal to G_1 and we get $L = \phi$. Hence the procedure is terminated. □

[Theorem 3] Given a sequence of extended kernels $\hat{H}_1$, $\hat{H}_2$, $\cdots$, $\hat{H}_j$, $\hat{H}_{j+1}$, $\cdots$, there exists a sequence of kernels $\tilde{G}_1$, $\tilde{G}_2, \cdots$, $\tilde{G}_i, \cdots$, $\tilde{G}_{i'}$, which satisfy the following Condition I.

Condition I: For any $j (\geq 1)$ there exists a subsequence of kernels $\tilde{G}_i$, $\tilde{G}_{i+1}, \cdots$, $\tilde{G}_{i'}$, such that $\tilde{G}_i = \hat{H}_j$ and $\tilde{G}_{i'} = \hat{H}_{j+1}$. □

This theorem is proved by showing the following claim.

[Claim] When in Condition I, $\tilde{G}_{i+k+l}$ is obtained from $\tilde{G}_{i+k}$ for any k ($0 \leq k \leq i' - i - 1$), a fundamental circuit that satisfies Condition C in the spanning tree-based kernel construction, can be always chosen. □

By this theorem it can be also shown that Theorem 1 and 2 also hold in the extended kernel construction.

Here, we give some notations. Given $\hat{H}_{j-1} = (\hat{V}_{j-1}, \hat{E}_{j-1})$ and C_v such that $v \in \hat{V}_{j-1}$, let p_k denote a simple path which extracts all edges in $\hat{H}_{j-1}$ from $c_k \in C_v$. For example, in Fig.4 given $\hat{H}_1 = c_1$ if $c_3 = v_5, v_4, v_7, v_6, v_2, v_5$, then $p_3 = v_4, v_7, v_6, v_2$. Thus, a set of simple paths $P_v = \{ p_1, p_2, \cdots, p_n \}$ is obtained from a set of fundamental circuits $C_v = \{ c_1, c_2, \cdots, c_n \}$. One of two end points of simple paths in P_v is always v. We call v W-node of $\hat{H}_{j-1}$ and denote it by $W(\hat{H}_{j-1})$. On the other hand, we call a set of the other end points of simple paths in P_v except v a set of W'-nodes of $\hat{H}_{j-1}$ and denote it by $W'(\hat{H}_{j-1})$. For example, in Fig.4 given $\hat{H}_1 = c_1$ if $W(\hat{H}_1) = v_4$ then $W'(\hat{H}_1) = \{v_2, v_5\}$.

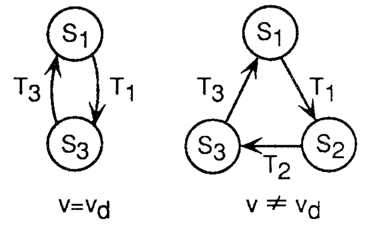


Figure 5: Finite state machine M.

4.2.2 Outline

Here, we explain the outline of distributed protocol of Phase 2. The protocol iterates the following three Procedures 1, 2, and 3 which use three control messages PR, EP, and TY, respectively. These control messages are different from messages. The control messages are used for this protocol while the messages are used for usual information transmission. In the following, the procedure to obtain $\hat{H}_j = (\hat{V}_j, \hat{E}_j)$ from $\hat{H}_{j-1} = (\hat{V}_{j-1}, \hat{E}_{j-1})$ is shown.

(1) Selection of W–node: Select W–node $W(\hat{H}_{j-1})$ by sending messages PR along a spanning tree.

(2) Construction of kernel $\hat{H}_j$: Suppose $W(\hat{H}_{j-1}) = w$. Send message EP from to all W'–nodes in $W'(\hat{H}_{j-1})$ along all simple paths $p_k \in P_w$.

(3) Decision of forward and backward paths: Send messages TY from all W'–nodes in $W'(\hat{H}_{j-1})$ to w along all simple paths $p_k \in P_w$. In this process, find forward and backward paths of all nodes $v \in \hat{V}_j - \hat{V}_{j-1}$.

4.2.3 Description

The proposed distributed protocol of Phase 2 is described by a combination of a finite state machine and an abstract program.

In this case the finite state machine has three states S_1, S_2, and S_3 in any node $v(\neq v_d)$ and two states S_1 and S_3 in a destination node v_d (see Fig.5). The abstract program shows a procedure triggered when each node receives a control message. To describe it Pascal–like program is adopted here.

We assume that before Phase 2 the states of all nodes are S_1. The transition T_1 from S_1 to S_2 is done when each node receives EP. The transition T_2 from S_2 to S_3 is done when each node receives TY. The transition T_3 from S_3 to S_1 is done when each node receives PR from all adjacent nodes on the spanning tree.

Before giving the procedure at a node v we explain control messages transmitted to v and variables stored in v. The information is used as local information of the proposed distributed protocol.

At first, control messages used in the protocol are PR, $EP(L, F, B, S)$ and $TY(R, D)$. L, F, B, R, and

```
A-1. begin  WN:=SEL;
A-2.   STATE : =$S_3$;
A-3.   FPATH : =v;
A-4.   BPATH : =v;
A-5.   L:=$c_i$ selected from $E(e_i)$ for an $e_i$;
A-6.   Send EP(L, v, v, nil) along $e_i$ such that $c_i \in E(e_i)$;
A-7.   $M(e_i) := E1$   end;
```

Figure 6: Procedure INITIALIZE.

D are information with which messages are transmitted. L represents a fundamental circuit $c_k \in C_w$ selected at a W–node w. F and B represent a forward path $r_1(w, v_d)$ and a backward path $r_2(w, v_d)$, respectively. S represents a subpath between w and v' of path $p_k \in P_w$ when v receives EP from v'. R represents a simple circuit which involves v_d, W–node w, W'–node w' and v, which is denoted by $R = v_1(= v_d)$, $v_2, \cdots, v_p$ (=W–node w), $\cdots$, $v_q(= v)$, $\cdots$, v_r (=W'–node w'), $\cdots$, v_{m-1}, v_m $(= v_d)$. Hence, two node–subsequences of R, that is $v_q, v_{q-1}, \cdots, v_p, \cdots, v_2, v_1$ (let it be a path P_a) and $v_q, v_{q+1}, \cdots, v_r, \cdots, v_m$ (let it be a path P_b) turn out to be a forward path of v or a backward path of v.

D represents which paths P_a and P_b are the forward path or the backward path. In the protocol it is determined so that if $D = FORWARD$ then P_a is the forward path and P_b is the backward path and if $D = BACKWARD$ then P_a is the backward path and P_b is the forward path (Refer to Fig.3).

Next, variables stored in v except those explained in Subsection 4.1 are the following six things. Here, let e_i denote an edge connected to v. WN indicates that if v is now W–node then $WN = SEL$ and otherwise $WN = NSEL$. $M(e_i)$ indicates four states of e_i. That is, $E0$ is an initial state, $E1$ is a state that EP is transmitted along e_i or received along e_i, $E2$ is a state that TY is transmitted along e_i or received along e_i, and $E3$ is a state that PR is received along e_i. $FPATH$ and $BPATH$ are a forward path of v and a backward path of v, respectively. $BF(e_i)$ indicates EP that v received along e_i, which is processed afterward. $STATE$ indicates a state of the finite state machine at v.

Phase 2 is triggered when v_d initiates Procedure $INITIALIZE$ in Fig.6. Phase 2 is terminated when v_d receives PR from all son nodes N_s of v_d on the spanning tree. Procedure $DISJOINT\_PATH$ is a procedure called when v receives a control message MSG. Procedure $START$ and $DECISION$ are called by Procedure $DISJOINT\_PATH$ and they execute the selection of W–node and the decision of forward and backward paths of v, respectively.

5 Example of Phase 2

We given an example of Phase 2 executed in $G_1 = (V_1, E_1)$ in Fig.4. In this example we focus on the propagation of states. In the following explanation, [S] represents a statement number s in the corresponding procedure in Fig.7.

```
Procedure DISJOINT_PATH
B-1.  begin  case v receives MSG from $e_i$
B-2.  MSG=EP (L,F,B,S):  begin
B-3.  if STATE=$S_l$  then begin
B-4.   M(ei) :=El;
B-5.   STATE :=$S_2$;
B-6.   Send EP(L,F,B,S·v) along $e_i$ such that $L \in E(e_i)$;
B-7.   M(ei) :=E1  end;
B-8.   if STATE=$S_2$ then $BF(e_i)$ := EP (L,F,B,S);
B-9.   if STATE=$S_3$ then call DECISION  end;
B-10. MSG=TY (R,D):  begin
B-11. $M(e_i)$ :=E2;
B-12. if STATE=$S_2$  then  begin
B-13.  STATE:=$S_3$;
B-14.  if D=FORWARD  then  begin
B-15.   FPATH:=subsequence of R, that is, $v_q, ..., v_r, ..., v_{m-1}, v_m$;
B-16.   BPATH:=subsequence of $\bar{R}$, that is, $v_q, ..., v_p, ..., v_2, v_1$
       end;
B-17.  if D=BACKWARD  then  begin
B-18.   FPATH:=subsequence of $\bar{R}$, that is, $v_q, ..., v_p, ..., v_2, v_1$
B-19.   BPATH:=subsequence of R, that is, $v_q, ..., v_r, ..., v_{m-1}, v_m$
       end;
B-20.  Send TY(R,D) along $e_i$ such that $M(e_i)$=E1;
B-21   $M(e_i)$ :=E2 such that $M(e_i)$=E1;
B-22   for all $e_i$ such that $BF(e_i) \neq$ EMPTY  do
B-23.  call DECISION  end;
B-24. if STATE=$S_3$ and for all $e_i$, $M(e_i) \neq$ E1
B-25.  then call START  end;
B-26. MSG=PR:
B-27. begin  $M(e_i)$ :=E3;
B-28. call START  end;  end;

Procedure START
C-1.  begin  if for all $e_i$, $M(e_i) \neq$ E1  then begin
C-2.  WN:=NSEL;
C-3   if for all $e_i$ such that $e_i$=(v,v')
C-4.    where v' $\in N - f \cup N_s$,  $M(e_i)$=E3  then begin
C-5.   STATE :=$S_1$;
C-6.   Send PR to the father node of v in $N_f$  end;
C-7.  else Send PR to one of the son node of v in $N_s$
        which is connected to $e_i$ such that $E(e_i) \neq$ E3  end;
C-8   if for some $e_i$, $M(e_i)$=E1  then begin
C-9.  WN:=SEL;
C-10. for all $e_i$ such that $M(e_i)$=E1  do begin
C-11. L:=$ci$ selected from $E(e_i) - \cup (E(e_i) \cap E(e_j)$
        such that j($\neq i$ and $M(e_j)$ =E1)
C-12. Send EP(L,FPATH,BPATH,nil) to along $e_i$  end end end;

Procedure DECISION
D-1.  begin  $M(e_i)$ :=E1;
D-2.  if B is disjoint from FPATH  B is BPATH of W node
D-3.  then  begin D:=FORWARD;
D-4.  R := $\bar{B}$ · S · FPATH {$\bar{B}$ = $v_d, ..., w$ and FPATH= $w', ..., v_d$}
      end;
D-5.  if F is disjoint from BPATH  F is FPATH of W node
D-6.  then  begin D:=BACKWARD
D-7.   R := $\bar{F}$ · S · BPATH {$\bar{F}$ = $v_d, ..., w$ and FPATH= $w', ..., v_d$}
      end;
D-8.  Send TY(R,D) along $e_i$ such that $M(e_i)$=E1
D-9.  $M(e_i)$ :=E2 such that $M(e_i)$=E1 end;
```

* In the figure $\bar{R}$, $\bar{F}$ and $\bar{B}$ represent the reversed node sequence of R F and B respectively .

Figure 7: Procedure DISJOINT_PATH.

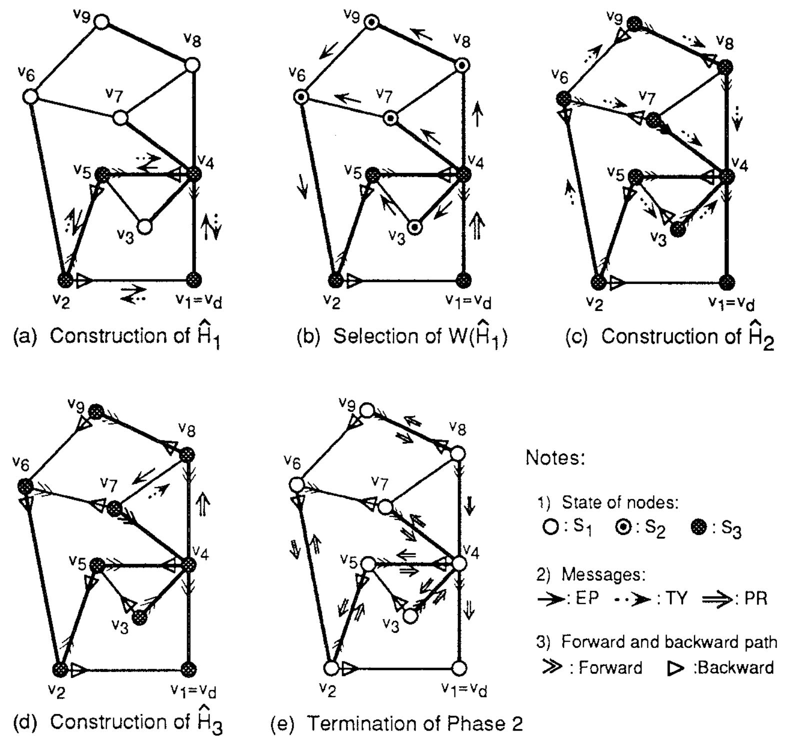
We assume that after Phase 1 the local information about a spanning tree of G_1 and fundamental circuits c_1, c_2, c_3, c_4, and c_5 are obtained as shown in Fig.4 and that the state of each node $v \in V_1$ is S_1. Figure 8 (a), (b), (c), (d), and (e) depict the five configurations of state of all nodes in G_1 in a successive time sequence. In the figure ○, ⊙, and ● represent three states of a node, that is, S_1, S_2, and S_3, respectively. $\rightarrow\!\!>$, $-\!>$ and $\Longrightarrow$ represent three control messages EP, TY and PR transmitted along an edge, respectively.

When v_d initiates Phase 2, v_d changes its state to S_3. Then, EP is transmitted along nodes v_1, v_4, v_5, v_2 and v_1, that is, a fundamental circuit c_1. In the process the states of nodes v_4, v_5 and v_2 are changed to S_2 [B–5]. After v_1 receives EP from v_2, TY is transmitted along nodes v_1, v_2, v_5, v_4 and v_1, that is, a fundamental circuit c_1. In the process the states of nodes v_2, v_5 and v_4 are changed to S_3 [B–13], as shown in Fig.8 (a), and the forward and backward paths of these nodes are determined. For example, according to $\gg$ and $\triangleright$ in the figure it is shown that a forward path of v_5 is v_5, v_4, v_1 and backward path of v_5 is v_5, v_2, v_1 [B–14 B–19]. Hence, the first kernel $\hat{H}_1$ is constructed.

Next, we try to select a W–node of $\hat{H}_1$, $W(\hat{H}_1)$. In this case after v_1 sends PR to v_4 [C–7], v_4 is selected as the W–node of $\hat{H}_1$ [C–8] as shown in Fig.8 (b). Then, v_4 computes $C_{v_4} = \{c_2, c_3, c_4\}$ [C–11], and EP is transmitted from v_4 to a W'–node of $\hat{H}_1$, that is, v_5 along a fundamental circuit c_2 while EP is also transmitted from v_4 to another W'–node of $\hat{H}_2$, that is, v_2 along two fundamental circuits c_3 and c_4. Here, we assume that EP reached at v_6 along c_3 earlier than along c_4. Then, the EP continues to be transmitted to v_2 along c_3. However along c_4, another EP transmitted later on along c_4 is not further transmitted to v_2, and it waits to be processed at v_6 afterward [B–8]. In the above process, the states of v_3 on c_2, v_7 and v_6 on c_3, and v_8 and v_9 on c_4 are changed to S_2 as shown in Fig.8 (b). W'–node v_2 and v_5 computes circuits, that is, v_1, v_4, v_9, v_6, v_2, v_1 (let it be C) and v_1, v_4, v_3, v_5, v_2, v_1, respectively (see Procedure DECISION). Then, TY with above information as R is transmitted from v_2 and v_5 to v_4 along c_2 and c_3, respectively. In the process, the states of v_3 on c_2 and v_7 and v_6 on c_3 are changed to S_3 while the forward and backward paths of their nodes are determined. For example, by using R obtained when v_6 receives TY, that is, the circuit C, the forward and backward paths of v_6 are determined as v_6, v_7, v_4, v_1 and v_6, v_2, v_1, respectively, as shown in Fig.8 (c). After the forward and backward paths of v_6 are determined, EP waited at v_6 is processed in a similar way [B–22, B–23] and then TY is transmitted from v_6 to v_4 along c_4.

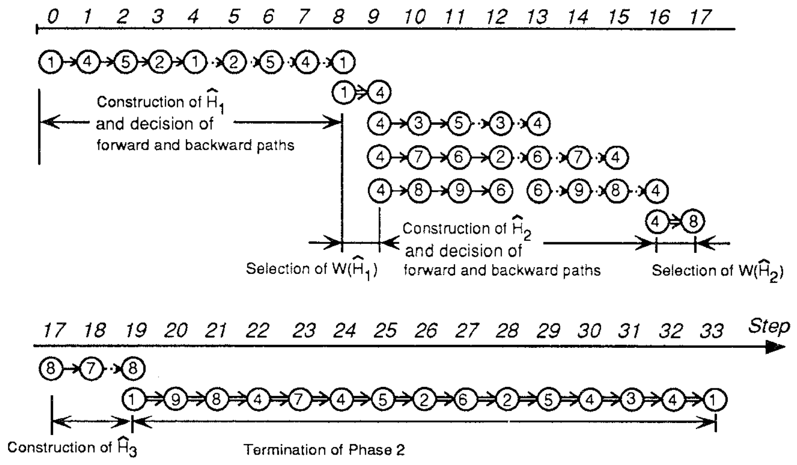
In the process, the states of v_9 and v_8 are changed to S_3 and the forward and backward paths are determined. Hence, the second kernel $\hat{H}_2$ is constructed.

Next, by sending PR from v_4 to v_8, v_8 is selected as $W(\hat{H}_2)$ as shown in Fig.8 (d). Then, v_8 computes C_{v_8} = $\{c_5\}$, and it sends EP to v_7 along c_5. Conversely, v_7 sends TY to v_8 along c_5. However, in this case the



(a) Construction of $\hat{H}_1$
(b) Selection of $W(\hat{H}_1)$
(c) Construction of $\hat{H}_2$
(d) Construction of $\hat{H}_3$
(e) Termination of Phase 2
Notes:
1) State of nodes: ○ : S_1 ⊙ : S_2 ● : S_3
2) Messages: → : EP ⇢ : TY ⇒ : PR
3) Forward and backward path ≫ : Forward ▷ : Backward

Figure 8: Example of Phase 2.



Construction of $\hat{H}_1$ and decision of forward and backward paths
Selection of $W(\hat{H}_1)$
Construction of $\hat{H}_2$ and decision of forward and backward paths
Selection of $W(\hat{H}_2)$
Construction of $\hat{H}_3$
Termination of Phase 2
Step

Notes: 1) Messages: → : EP ⇢ : TY ⇒ : PR
2) Each circle number represents the coresponding node with samnumber.

Figure 9: Timing sequence of message transmission.

forward and backward paths for newly added nodes to $\hat{H}_3$ are not determined because there do not exist such nodes. Hence, the third kernel $\hat{H}_3$ is constructed.

We try to select a new W–node by transmitting PR along v_8, v_9, v_8, v_4, v_7, v_4, v_5, v_2, v_6, v_2, v_5, v_4, v_3, v_4, v_1. However, the new W–node is not selected. In the process the states of all nodes in G_1 are changed to S_1. Hence, when v_1 receives PR from v_4, Phase 2 is terminated.

For easier understanding of behaviors of the proposed distributed protocol (Phase 2), Fig.9 explains time sequence of message transmission in the example described above. In the figure we assume a hypothetical synchronization of the protocol in which every node executes a "step" of the protocol simultaneously at the fixed point in time. At each step a node may receive and process one message from each adjacent node and send messages to appropriate adjacent nodes. In Fig.9 the Termination of Phase 2 represents a procedure which tries to select a new W–node in vain. Before this procedure all nodes in G_1 obtain the forward and backward paths.

6 Concluding Remarks

We have presented a distributed protocol for finding two node–disjoint paths between each pair of nodes in a computer network. The routing derived from the protocol is reliable and adaptive to changes in network topology and load.

The proposed distributed protocol has the following advantageous features (1), (2), and (3). (1) Even if a node or an edge on a forward path of every node in a network, at which a message is originated, fails, the message can be transmitted by changing its path to another backward path of an intermediate node (and vice versa), because the intermediate node in the network has two node–disjoint forward and backward paths to the destination node v_d. (2) Load due to messages originated and transmitted at every node (let it be v) in a network can be independently split at v into forward and backward paths between v and the destination node v_d. The split load will not be interfered each other to the destination node v_d. (3) Even when changes of network topology occur, two node–disjoint paths between every pair of nodes can be updated by using the proposed protocol as long as the network is biconnected.

Now, we consider the complexities of the proposed distributed protocols. There exits three complexity measures to be accounted for. The first measure is communication complexity, which is evaluated by [the size of control messages] $\times$ [the number of transmitting control messages] (bits). The second is time complexity, which is defined by a time interval between the initiation and termination of protocol. The time is evaluated under the assumption that the processing time within each node is neglected and that the transmission time delay along any edge is the unit time. The third measure is space complexity, which is evaluated by the size of storage needed by each node (bits).

Let n and e denote the number of nodes and edges of a graph, respectively. The complexities of the proposed distributed protocol (Phase 1 and Phase 2) are $O(e^2 \log n)$, $O(e)$ and $O(e \log n)$ with respect to communication complexity, time complexity and space complexity, respectively. We can further improve the time complexity of the protocol by transmitting PR in parallel. However, the improved protocol requires more communications and spaces. Furthermore detailed considerations about improving the three complexities of the protocol should be done at the level of practical use.

References

[1] Attar, R.: "A distributed adaptive multi-path routing–consistent and conflicting decision making," *Proc. of 5th Berkeley Workshop on Distributed Data management & Computer Networks*, pp.217-236, 1981.

[2] Babson, M., Buster, D., Deval, G. and Xavier, J. S.: "ATM switching and CPE adaptation in the north carolina information highway," *IEEE NETWORK*, Vol.8, No.6, pp.40-46, 1994.

[3] Chandy, K. M. and Misra, J.: "Distributed computation on graphs: shortest path algorithms," *Commun. of ACM*, Vol.25, No.11, pp.833-837, 1982.

[4] Jaffe, J. M.: "Distributed multi-destination routing: The constraints of local information," *IBM Research Report* RC9247, 1982.

[5] Kakuda, Y.: *"Fault-tolerant distributed control in packet switching networks,"* Ph.D. Dissertation, Hiroshima Univ., 1983.

[6] Moss, F. H. and Merlin, P. M.: "Some theoretical results in multiple path destination in networks," *Networks*, Vol.11, pp.401-411, 1981.

[7] Pattavina, A.: "Nonblocking architectures for ATM switching," *IEEE Communications Magazine*, Vol.31, No.2, pp.38-48, 1993.

[8] Rai, S. and Agrawal, D. P.: *"Distributed computing network reliability,"* IEEE Computer Society Press, 1990.

[9] Schwartz, M.: *"Telecommunication networks: Protocols, modeling and analysis,"* Addison–Weslay, 1987.

[10] Segall, A.: "Distributed network protocols," *IEEE Trans. Inform. Theory*, Vol.IT-29, No.1, pp.23-35, 1983.

[11] Tanenbaum, A. S.: *"Computer networks (second edition),"* Prentice–Hall, 1988.

[12] Toueg, S.: "An minimum hop path failsafe and loop-free distributed algorithm," *IBM Research Report* RC8530, 1980.