12 54

ADA087623

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered) UNCLASSIFIED

12

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO. AD-A087623

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
DISTRIBUTED NETWORK PROTOCOLS

5. TYPE OF REPORT & PERIOD COVERED
Paper-Technical rept.

6. PERFORMING ORG. REPORT NUMBER
LIDS-P-1014

7. AUTHOR(s)
Adrian Segall

8. CONTRACT OR GRANT NUMBER(s)
ARPA Order No. 3045/5-7-75
[redacted]/N00014-75-C-1183
ARPA Order-3045

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Massachusetts Institute of Technology
Laboratory for Information and Decision Systems
Cambridge, Massachusetts 02139

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Program Code No. 5T10
ONR Identifying No. 049-383

11. CONTROLLING OFFICE NAME AND ADDRESS
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209

12. REPORT DATE
Jul[redacted]80

13. NUMBER OF PAGES
52

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
Office of Naval Research
Information Systems Program
Code 437
Arlington, Virginia 22217

LEVEL

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

DTIC ELECTE AUG 7 1980 S C

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A unified approach to the formal description and validation of several distributed protocols is presented. After introducing two basic protocols, a series of known and new protocols for connectivity test, shortest path and path updating are described and validated. All protocols are extended to networks with changing topology.

80 8 4 246

DDC FILE COPY

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

410950

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

July 1980 LIDS-P-1014

DISTRIBUTED NETWORK PROTOCOLS

Adrian Segall
Department of Electrical Engineering
Technion, Israel Institute of Technology,
Haifa, Israel.

ABSTRACT

A unified approach to the formal description and validation of several distributed protocols is presented. After introducing two basic protocols, a series of known and new protocols for connectivity test, shortest path and path updating are described and validated. All protocols are extended for networks with changing topology.

This research was conducted on a consulting agreement with the Laboratory for Information and Decision Systems at MIT, Cambridge, Mass. with partial support provided by the Advanced Research Project Agency of the US Department of Defense (monitored by ONR) under contract No. N00014-75-C-1183.

This paper has been submitted to the IEEE Transactions on Information Theory.

1. Introduction

Consider the situation when a number of physically distinct computation units work on a common problem, while their operation is coordinated via communication channels connecting some of these units. Each computation unit has certain processing and memory capability and is preprogrammed to perform its part of the computation, as well as to receive and send control messages over the communication channels. The program residing in each node will be referred to as the node algorithm and the ensemble of all algorithms providing the solution to the common problem is named a distributed protocol.

For the purpose of this paper it is convenient to regard the computation units as nodes in a network whose links are the connecting communication channels. The specific protocols considered here will be collectively called Distributed Network Protocols (DNP) to indicate the fact that the common problem that has to be solved is connected with the network topology. Many of the "classical" graph algorithms have their distributed version, and, in addition, several new distributed network protocols appear from practical problems. The main application considered so far for DNP's is in data or voice communication networks. In such networks, geographically dispersed devices must transmit information to one another and must somehow coordinate this transmission. With the advances of mini and micro-computers, it is certainly feasible that nodes will have their own processing and memory unit and will serve as communication processors and/or as switches. In principle, the common goal of all these units is to efficiently transmit the required information to achieve certain performance goals, like minimum delay or maximum throughput. With this application in mind, several examples of problems for which DNP's have been proposed or are currently under investigation are routing of information, shortest path, minimum weight spanning tree, common channel random access coordination and others.

The main purpose of the present paper is to give a formal description and rigorous validation to a number of DNP's, some of which have been presented previously in an intuitive way and some of which are proposed here for the first time. We mainly consider DNP's for the purpose of connectivity tests, shortest path in terms of number of links and routing-path updating. In addition, we give a unifying approach to the validation of the protocols by presenting several basic simple DNP's that provide building blocks to the presented protocols.

The presented protocols have one additional important feature. Since nodes and links may fail and be added asynchronously to the network, the protocols must be able to work under arbitrarily changing network topology. Although we first consider DNP's for networks with fixed topology, in Sec. 7 we extend those protocols to incorporate cases of changing topology.

2. The General Model

In this section we give the general model and assumptions used in all presented DNP's. Consider a network (V,E) where V is a set of nodes and $E \subset V \times V$ is a set of links. For the first part of this paper, we assume that the network has fixed topology. We shall use the following assumptions:

a) Each link is bidirectional; the link connecting the node i with node j considered in the direction from i to j is denoted (i,j).

b) All messages referred to in this paper are control messages.

c) On each link in each direction there is a link protocol that insures that each message sent by node i say on link (i,j) will arrive *correctly within finite nonzero undetermined time and* all messages are received at node j in the same order as they were sent by i (observe that we do not preclude channel errors, provided that there exists a proper detection/retransmission or correction algorithm on each link).

d) All messages received at a node i are stamped with the identification of the link from which they came and then are transferred into a common queue; each node uses one processor for the purpose of the algorithm; the processor extracts the control message at the head of the queue, proceeds to process it and discards the message when processing is completed; no other operation related to the protocol is performed by the processor while a message is being processed; consequently we may assume that the processing of each message takes zero time.

e) Each node has an identification; before the protocol starts, each node knows the identity of all nodes that are potentially in the network; it knows nothing about the topology of the network and in particular about what nodes actually belong to the network. We denote by

$1,2,\ldots,|\bar{V}|$ the nodes that are potentially in the network and, when needed, by $1,2,\ldots,|V|$ the nodes actually belonging to the netowrk.

f) Each node knows its adjacent links, but not necessarily the identity of its neighbors, i.e. the nodes at the other end of the links; however, in our algorithms it will be convenient to use expressions like: "send messages to all neighbors", meaning "send messages over all adjacent links". The collection of all neighbors of node i will be denoted by G_i .

g) Unless otherwise stated, the protocol can be started by any node or by several nodes asynchronously; a node starts the algorithm by receiving a special message "START" from the outside world; a standing assumption is that, once a node has entered the algorithm, it cannot receive "START".

3. Basic Protocols

The two basic DNP's presented in this section provide a way for broadcasting information in the network.

3.1 Propagation of Information (PI)

Suppose that node j receives from the outside world a piece of information that has to be transmitted to all nodes in the network. The simplest procedure to accomplish this is for node i to transmit a message containing this information to all its neighbors and for each other node k in the network, when it receives the first such message, to send a similar message to its own neighbors. All other messages received at k are disregarded. We shall now formally present the algorithm for each node and validate the protocol.

PROTOCOL PI

Variables of the algorithm at node i

m_i shows if node i has already entered the algorithm (values 0,1).

Messages sent and received by the algorithm at node i

MSG - message sent by node i ;

MSG(ℓ) - message received from neighbor ℓ ;

START - message received from the outside world ;

It is assumed that each message carries the piece of information that has to be propagated.

Algorithm for node i

Assumption: just before entering algorithm, node i has $m_i = 0$.

1. For START[1] or MSG(ℓ)
2. <u>if</u> $m_i = 0$, then: $m_i \leftarrow 1$; send messages to all neighbors.

Properties of the protocol

Theorem PI-1

Suppose a node j receives START. Then:

a) All nodes i connected to j (i.e. that are in the connected network containing j) will set $m_i \leftarrow 1$ in finite time.

b) During the execution of the protocol, exactly one MSG is being sent on each link in each direction.

c) The propagation of information is the fastest possible in the following sense: for a node i, let p_i be the node from which node i receives the first MSG (see line <2> of the Algorithm[2]). For a link (i,ℓ) let the weight $w_{i\ell}$ of that link be the time it took for MSG to travel from i to ℓ, i.e. from the time i sends MSG on (i,ℓ) until the time the processor at ℓ starts operating on the MSG (this includes propagation and queueing time). Then the collection of links $\{(p_i,i)$, for all i in the network$\}$ forms the tree of shortest distances from j to all nodes.

Proof

The proof of all properties is straightforward and we give here only an outline. Property a) follows by induction on the distance (in terms of numbers of links) from node j. Suppose all nodes i that are at distance r from j perform $d_i \leftarrow 1$. Then a node k at distance $(r + 1)$ is a neighbor of a node at distance r and when receiving MSG from it, either this is the first message at k, in which case d_k becomes 1 or it is not, in which case d_k is 1 already. Property b) follows from the fact that for all nodes i, the parameter d_i becomes 1 exactly once, at which time node i sends MSG on all adjacent links. Property c) holds because if there was a shorter route from i to j, node i would have received MSG on that route before receiving MSG from p_i.

Before proceeding to the second algorithm, we may note that the protocol will work correctly even if "START" is delivered to several nodes at arbitrary times, provided that each of these nodes has not entered the algorithm before receiving "START". Properties a) and b) still hold and the propagation is still the fastest possible.

3.2 Propagation of Information with Feedback (PIF)

Sometimes a node s that receives START and propagates information may want to be positively informed when the information has indeed reached all connected nodes. Here of course the assumption is that only one node can receive START. The following protocol can be used for this purpose. When receiving START, node s sends MSG^s to all neighbors[3]. When receiving any MSG^s, an arbitrary node i marks the link from which it was received. When receiving the first MSG^s from neighbor ℓ say, a node i denotes this neighbor with a special mark p_i^s, and sends MSG^s to all neighbors except to p_i^s. When it observes that it has received MSG from all neighbors, a node i other than s sends MSG^s to p_i^s. It is shown below that receipt of MSG^s from all neighbors at node s can be interpreted as the signal that the information has indeed reached all connected nodes. In this way, the propagation of MSG's occurs in two waves: (i) from node s into the network for purposes of propagating information, and (ii) from the network back to node s for the purpose of acknowledgment. The formal description of the protocol follows.

PROTOCOL PIF

The algorithm for node s that receives START is different from the algorithm for all other nodes. We shall first give the algorithm for an arbitrary node i other than s and then for node s.

Variables of the algorithm at node $i \neq s$

m_i^s shows if node i is currently participating in the protocol (values 0,1);

$N_i^s(\ell)$ marks receipt of MSG^s from neighbor ℓ (values 0,1), $\ell \epsilon G_i$;

p_i^s - neighbor from which MSG^s was received first.

<u>Messages sent and received by the algorithm at node $i \neq s$</u>

MSG^s and $MSG^s(\ell)$ with the same meaning as MSG in PI.

<u>Algorithm for node $i \neq s$</u>

Assumption: just before entering algorithm, node i has $m_i^s = 0$, $p_i^s = nil$, $N_i^s(\ell) = 0$ for all $\ell \in G_i$.

1. For $MSG^s(\ell)$
2. $N_i^s(i) \leftarrow 1$;
3. if $m_i^s = 0$, then: $m_i^s \leftarrow 1$; $p_i^s \leftarrow \ell$; send MSG^s to all neighbors except p_i^s.
4. if $\forall \ell' \leftarrow G_i$ holds $N_i^s(\ell') = 1$, then: send MSG^s to p_i^s; $m_i^s \leftarrow 0$; $\forall \ell' \in G_i$, set $N_i^s(\ell') \leftarrow 0$.

<u>Algorithm for node s</u>

For node s, the variables are m_s^s, $N_s^s(\ell')$ for all $\ell' \in G_s$, the messages are $MSG^s(\ell)$ and START and the algorithm is:

3. For START

3a. $m_s^s \leftarrow 1$; send MSG^s to all neighbors.

For $MSG^s(P)$

$N_s^s(\ell) \leftarrow 1$;

4. if $\forall \ell' \in G_s$, holds $N_s^s(\ell') = 1$, then: $m_s^s \leftarrow 0$; $\forall \ell' \in G_s$, set $N_s^s(\ell') \leftarrow 0$.

<u>Note</u> The lines in the algorithm for s have been numbered to denote similar operations as in the algorithm for an arbitrary node i.

In order to analyse the protocol, we shall need the following notations:

$\langle\cdot\rangle_i$ - the event of node i performing line <·> of its algorithm; whenever the corresponding line contains an <u>if</u> operation, the notation refers only to the cases when the condition indeed holds.

t(*) - time when event * happens.

<u>Theorem PIF-1</u>

Suppose node s receives START. Then

a) all connected nodes i will perform the event $\langle 3\rangle_i$ in finite time and exactly once; after this happens, the links

$$\{(i,\ p_i^s) \text{ for all connected } i\ \}$$

will form a directed tree rooted at j; in addition, for all i

$$t(\langle 3\rangle_i) > t(\langle 3\rangle_{p_i^s}) \qquad (3.1)$$

b) node j and all connected nodes i will perform <4> in finite time and exactly once; moreoever

$$t(\langle 3\rangle_i) \le (\langle 4\rangle_i) < t(\langle 4\rangle_{p_i^s}); \qquad (3.2)$$

also, when node s performs <4>, all connected nodes will have complet ed the algorithm, i.e. performed <4>.

c) exactly one MSG travels on each link in each direction.

<u>Proof</u>

a) and c) follow from Theorem PI-1. To prove b) let k be a leaf of the tree referred to in a), i.e. $\nexists \ell$ such that $p_\ell^s = k$. Then all neighbors m of k will send MSG to k whenever they perform $<3>_m$. Node k will receive all these messages and will be able to perform $<4>_k$. At that time it will send MSG to p_k^s. The same will be true for all leaves. Now nodes that are on the last-but-one level in the tree will be able to perform <4> and the procedure will continue downtree all the way to node s. This argument clearly proves (3.2) and completes the proof of the Theorem.

4. Connectivity Test Protocols

The purpose of this class of DNP's is to allow each node to learn what nodes are connected to it.

Protocol CT1

The idea here is to use protocol PI, first to inform all nodes that the protocol is in progress and then for each node to propagate its own identity. Every node (or several nodes) can start the protocol by receiving START. A node enters the protocol whenever it receives either START or the first control message from any of its neighbors. The first action taken by a node when entering the protocol is to send a control message containing its own identity to all its neighbors, thereby starting propagation of this identity. In addition, whenever a node i receives the first control message with the identity of some other node j, it marks j as connected and sends a message MSG^j with the identity of j to all neighbors. All further messages with the identity of j are discarded with no action taken.

Variables of the algorithm at node i

M_i - shows if i has already entered the algorithm (values NORMAL, WORK);

d_i^j - shows if i knows whether j is connected (values 0,1), for $j = 1,2,\ldots|\bar{V}|, j \neq i$.

Messages sent and received by the algorithm at node i

MSG^j - control messages with identity j sent by i ;

$MSG^j(\ell)$ - message with identity j received by i from ℓ ;

START - same meaning as in PI.

Algorithm for node i

Assumption: just before entering protocol, holds $d_i^j = 0$ for all j.

1. For START or $MSG^j(\ell)$
2. *if* M_i = NORMAL, then: $M_i \leftarrow$ WORK; send MSG^i to all neighbors.
3. *if* $d_i^j = 0$, then: $d_i^j \leftarrow 1$; send MSG^j to all neighbors.

Theorem CT1-1

If node j is connected to i and START is delivered to any node connected to j (or to j itself), then d_j^i will become 1 in finite time and if i and j belong to disconnected networks, then d_j^i will remain 0 forever.

Proof

The event $M_k \leftarrow$ WORK propagates as in PI and hence will happen in finite time at all nodes k connected to the node that received START. For a given i, after M_i becomes WORK, the event $d_k^i \leftarrow 1$ propagates again as in PI and hence will happen in finite time at node j. The second part of the Theorem is obvious.

Theorem CT1-2

With protocol CT1, there is no way for node j to know for sure what nodes are disconnected from it or in other words, there is no way for j to know when the algorithm is completed, except for the case when all nodes are connected.

Proof

Consider first the case of three nodes 1, 2, 3 with links (1,2) and (2,3). If 1 starts the protocol, it will receive the same sequence of messages whether (2,3) is working or not, except that if it does, it will later receive the identity of 3. Now, after receiving the identity of node 2 and before receiving the identity of 3, there is no way for node 1 to positively know whether it has already completed the protocol or not, i.e. whether new identities are supposed to still arrive. It is easy to see that similar situations may arise for any other topology.

Communication cost

The number of bits transmitted on each link in each direction is $|V| \log_2 |\overline{V}|$. This is because every identity travels exactly once on each link in each direction, there are $|V|$ identities and it takes $\log_2 |\overline{V}|$ bits to describe an identity. The total number of bits in the network is $2|E| \, |V| \log_2 |\overline{V}|$, where E is the number of bidirectional links.

The rest of this section is devoted to the presentation of several protocols that solve the problem raised in Theorem CT1-2, namely allow nodes to positively know that the protocol has indeed been completed. We shall say then that the protocol has the termination property. Protocol CT2 achieves the property by employing the basic protocol PIF, while the others use a different idea.

Protocol CT2

The protocol is started and entered by nodes in the same way as in CT1. Whenever a node i receives the first message MSG^j with the identity of j, from neighbor ℓ say, a node i denotes this neighbor (as in PIF) with a special mark p_i^j, and sends MSG^j to all neighbors, except to p_i^j. When it observes that it has received MSG^j (for $j \neq i$) from all neighbors, node i sends MSG^j to p_i^j. The termination property holds because it is shown below that receipt of MSG^i from all neighbors can be interpreted as the signal that node i positively knows the nodes that are connected to it and also the nodes that are disconnected.

Variables of the algorithm at node i

M_i = WORK while i is participating in the protocol and = NORMAL after completing the protocol;

d_i^j shows if i knows whether j is connected (values 0,1) for all j;

$N_i^j(\ell)$ shows if MSG^j has been received already from neighbor ℓ (values 0,1) for all j and $\ell \in G_i$;

p_i^j - neighbor from which MSG^j has been received first, for all j.

Messages received and sent by the algorithm at node i

Same as in CT1.

Algorithm for node i

Assumption: just before node i enters the algorithm, it has

$$d_i^j = 0,\ N_i^j(\ell) = 0 \text{ for all } j \text{ and } \ell \in G_i.$$

1. For START or $MSG^j(\ell)$, $j \neq i$

1a. <u>if</u> MSG, then: $N_i(\ell) \leftarrow 1$.

2. <u>if</u> $d_i^i = 0$, then: $M_i \leftarrow$ WORK; $d_i^i \leftarrow 1$; send MSG^i to all neighbors.

3. <u>if</u> MSG and $d_i^j = 0$, then: $d_i^j \leftarrow 1$; $p_i^j \leftarrow \ell$; send MSG^j to all neighbors, except p_i^j.

4. <u>if</u> $\forall \ell' \in G_i$ holds $N_i^j(\ell) = 1$, then: send MSG^j to p_i^j; $\forall \ell' \in G_i$, set $N_i^j(\ell') \leftarrow 0$.

5. For $MSG^i(\ell)$

5a. $N_i^i(\ell) \leftarrow 1$;

6. <u>if</u> $\forall \ell' \in G_i$, holds $N_i^i(\ell') = 1$, then: $M_i \leftarrow$ NORMAL; $\forall \ell' \in G_i$, set $N_i^i(\ell') \leftarrow 0$.

In order to analyse the protocol, we shall need the following notation (see also notations just before Theorem PIF-1):

$\langle\cdot\rangle_i^j$ - the event of node i performing line $\langle\cdot\rangle^j$ of its algorithm regarding node j (i.e. reacting to receipt of MSG^j).

The properties of the algorithm are given in the following:

Theorem CT2-1

Suppose START is delivered to any node connected to a given node j (or to j itself). Then :

a) same as Theorem CT1-1

b) node j will perform $<6>_j$ in finite time and exactly once, and when this happens, it will have $d_j^k = 1$ for all connected nodes k and $d_j^k = 0$ for all disconnected nodes k. In other words, it will positively know at that time what nodes are connected, resolving the problem raised in theorem CT1-2.

<u>Proof</u>

The event $M_k \leftarrow WORK$ propagates as in PI and hence will happen in finite time at all nodes k connected to the node that received START. For a given node i, after d_i^i becomes 1, the event $d_k^i \leftarrow 1$ (i.e. $<3>_k^i$ in the present protocol) propagates in the same way as $<3>_k$ in PIF and hence (cf. Thm. PIF-1) it will happen in finite time at node j, completing the proof of a). Similarly, for the given node i, $<4>_k^i$ propagates in the same way as $<4>_k$ in PIF and hence $<4>_j^i$, $<4>_i^j$ and $<6>_j$ will happen in finite time, each exactly once. It remains to show that $<6>_j$ is indeed the signal indicating that node j knows all connected nodes, namely to show that

$$t(<3>_j^k) < t(<6>_j) \tag{4.1}$$

for all nodes k connected to j. For given k and j, consider the nodes $k = i_0, i_1, \ldots, i_r = j$, where $i_{\ell+1} = p_{i_\ell}^j$ for $\ell = 0, 1, \ldots, r-1$. In words, this is the branch of the tree rooted at j referred to in Thm. PIF-1 on which node k sits. We wish to prove (4.1) by using induction on the mentioned series of nodes, namely we want to prove by induction that

$$t(<3>_i^k) < t(<4>_i^j) \quad \text{for } i = k, i_1, \ldots, i_{n-1}, j. \tag{4.2}$$

Observe that $<3>_k^k$ is not defined in the algorithm and is used here for convenience of notation to mean $<2>_k$ and similarly, $<4>_j^j$ means $<6>_j$.

Since $<2>_k \equiv <3>_k^k$ is the first operation at node i, expression (4.2) is clearly true for $i = k = i_0$. Now the induction will be complete if we prove that for any node i, the fact

$$t(\langle 3\rangle_i^k) < t(\langle 4\rangle_i^j) \tag{4.3}$$

implies

$$t(\langle 3\rangle^k_{p_i^j}) < t(\langle 4\rangle^j_{p_i^j}) \ . \tag{4.4}$$

We distinguish two cases. Suppose first that $p_i^k = p_i^j$. Then (3.1) applied to k implies

$$t(\langle 3\rangle_i^k) > t(\langle 3\rangle^k_{p_i^k}) \tag{4.5}$$

and (3.2) applied to j implies

$$t(\langle 4\rangle_i^j) < t(\langle 4\rangle^j_{p_i^j}) \ . \tag{4.6}$$

These, combined with (4.3) and the fact $p_i^k = p_i^j$ imply (4.4). Suppose next that $p_i^k \neq p_i^j$. Let us denote by $SEND_i^k(\ell)$ and $RCV_i^k(\ell)$ the event of node i sending/receiving MSG^k to/from neighbor ℓ respectively. Then <3> and Assumption d) in Sec. 2 imply that

$$t(\langle 3\rangle_i^k) = t(SEND_i^k(p_i^j)) \tag{4.7}$$

and <4> says that

$$t(\langle 4\rangle_i^j) = t(SEND_i^j(p_i^j)) \ . \tag{4.8}$$

Now (4.3) , Assumption c) in Sec. 2 and (4.7), (4.8) imply

$$t(RCV^k_{p_i^j}(i)) < t(RCV^j_{p_i^j}(i)) \ . \tag{4.9}$$

But

$$t(<3>^k_{p_i^j}) \leq t(RCV^k_{p_i^j}(i)) \qquad (4.10)$$

since $<3>^k$ is performed whenever the first MSG^k is received, and similarly,

$$t(<4>^j_{p_i^j}) \geq t(RCV^j_{p_i^j}(i)) \qquad (4.11)$$

since $<4>^j$ is performed after having received MSG^j from all neighbors. Now, (4.9) - (4.11) imply (4.4), and this completes the induction and the proof of the Theorem.

Communication cost

Observe that by Theorem CT2-1, the communication requirements of CT2 are the same as those of CT1, namely $|V| \log_2 |\bar{V}|$ bits per link in each direction. Observe however that the storage and processing requirements, as well as the required execution time are larger than in CT1.

Protocols CT3 - CT5 use a different idea for achieving the termination property. CT3 is quite wasteful in terms of communication requirements, but it is convenient in order to illustrate the idea and to be used as a basis for developing the more efficient versions CT4 and CT5. In addition, it can be used for different purposes, like learning the network topology.

Protocol CT3

Suppose we use protocol CT1, except that for each node we propagate not only the identity of the node, but also of its neighbors. In other words MSG^j of CT1 will now carry the identity of j as well as of all its neighbors, i.e. will have the format $MSG^j(K_j)$, where K_j contains the identities of all neighbors of j. The termination property is achieved using the fact that, if a node k receives $MSG^j(K_j)$, it will eventually receive $MSG^i(K_i)$ as well, for any $i \in K_j$, and the termination signal will occur when node k will have heard from all these nodes. Clearly, the algorithm at each node will have two stages, where in the first one it will learn the identity of its own neighbors and in the second will proceed with the protocol as described before. In the description of the protocol, we shall use a special notation WAKE for messages belonging to the first stage.

Variables of the algorithm at node i

M_i same meaning as in CT2 ;

d_i^i = 0 before entering algorithm,

1 while looking for identity of neighbors,

2 while looking for all connected nodes;

d_i^j = 0 when i knows nothing about j (for $j \neq i$),

1 while i knows j only as a neighbor of another node,

2 while i knows j directly (i.e. $MSG^j(K_j)$ has been received);

$N_i(\ell)$ shows if WAKE has been received from neighbor ℓ (values 0,1);

K_i is the list containing the identities of all neighbors of i.

Messages received and sent by the algorithm at node i

$MSG^i(K_i)$ - message containing identities of i and of its neighbors;

$WAKE^i$ - message asking the neighbors to wake up and to send their identity;

START - as before.

Similarly $MSG^j(K_j)$ and $WAKE^\ell$ for received messages.

<u>Algorithm for node i</u>

Assumption: just before node i enters algorithm, it has K_i = empty and $d_i^j = 0$, $N_i(\ell) = 0$ for all j and $\ell \in G_i$,

1. For START

1a. $d_i^i \leftarrow 1$; $M_i \leftarrow$ WORK; send WAKE^i to all neighbors.

2. For WAKE^ℓ

2a. $N_i(\ell) \leftarrow 1$; include ℓ in K_i ;

2b. <u>if</u> $d_i^i = 0$, then: same as <1a> ;

2c. $d_i^\ell \leftarrow \max\{d_i^\ell, 1\}$;

3. <u>if</u> $\forall \ell' \in G_i$, holds $N_i(\ell') = 1$, then

3a. $d_i^i \leftarrow 2$; $\forall \ell' \in G_i$, set $N_i(\ell') \leftarrow 0$, send $\text{MSG}^i(K_i)$ to all neighbors.

4. For $\text{MSG}^j(K_j)$ and M_i = WORK

5. <u>if</u> $d_i^j \neq 2$, then $d_i^j \leftarrow 2$; $k \in K_j$, set $d_i^k \leftarrow \max\{d_i^k, 1\}$, send $\text{MSG}^j(K_i)$ to all neighbors.

6. <u>if</u> $\forall j$ holds $d_i^j = 2$ or 0, then $M_i \leftarrow$ NORMAL.

The properties of the protocol are given in the following:

<u>Theorem CT3</u>

Suppose START is delivered to one or more nodes. Then

a) exactly one message WAKE traverses each link in each direction;

b) <3> happens at all connected nodes in finite time and exactly once;

c) when $\text{MSG}^i(K_i)$ is sent by node i (see <3a>), then K_i contains exactly the identities of all neighbors of i;

d) for each node j in the connected network, exactly one message $\text{MSG}^j(K_j)$ traverses each link in each direction;

e) every node i will perform <6> (i.e. $M_i \leftarrow$ NORMAL) in finite time and when this happens it will have $d_i^j = 2$ for all connected nodes j and $d_i^j = 0$ for all disconnected nodes j (i.e. this is the termination signal).

Proof

The propagation of WAKE happens as in protocol PI and hence a) and b). Now, c) is clear from condition <3>. For each j, propagation of $MSG^j(K_j)$ happens as in protocol PI except that it is triggered by <3> instead of by START and hence d). In order to prove e), consider the situation after all messages considered in d) have arrived. Then from <5>, a node i will have $d_i^j = 2$ for all nodes j connected to it. For all disconnected nodes j, it will have $d_i^j = 0$ and hence <6> will be performed. It remains to prove that there cannot be a situation where <6> holds while $d_i^j = 0$ for some connected node j. If this was the case, there must exist a set of nodes V containing i, where V is not the entire network and $d_i^j = 2$ for $j \in V$, while $d_i^k = 0$ for $k \notin V$. But then <5> shows that $d_i^k \geq 1$ for all k that are neighbors of any node in V, contradicting $d_i^k = 0$ for all $k \notin V$. This completes the proof of e) and of the theorem.

Communication cost

On each link in each direction we need $\log_2|\overline{V}|$ bits for the WAKE message and $|V|\ (D + 1)\ \log_2|\overline{V}|$ bits for the MSG messages, where D is the average degree of the nodes (average number of neighbors). Clearly $D = 2|E|/|V|$ and hence the communication cost is $(2|E|+|V|+ 1)\ \log_2|\overline{V}|$ bits per link in each direction.

As mentioned before, protocol CT3 employs too much communication and its performance can be considerably improved. One way is to use the position of a variable in a vector to indicate the identity of a node, instead of explicitly mentioning it. This idea was used in a protocol by Finn [3] and we present here an improved version of that protocol:

Protocol CT4

Variables of the algorithm are d_i^i , d_i^j , $N_i(\ell)$, M_i with the same meaning as in CT3.

<u>Messages sent and received by the algorithm at node i</u>

START;

$D_i = \{d_i^1, d_i^2, \ldots, d_i^{|\bar{V}|}\}$, message sent;

$D(\ell)$ message received from neighbor ℓ ; we denote its contents by

$\{d^1, d^2, \ldots, d^{|\bar{V}|}\}$.

<u>Algorithm for node i</u>

Assumption: just before node i enters algorithm, it has all $d_i^j = 0$ and $N_i(\ell) = 0$ for all j and $\ell \in G_i$.

1. For START

1a. $d_i^i \leftarrow 1$; $M_i \leftarrow$ WORK; send D_i to all neighbors.

2. For $D(\ell)$

2a. <u>if</u> $D(\ell) = \{0,0,\ldots,0,1,0,\ldots,0\}$, then

2b. $N_i(\ell) \leftarrow 1$;

2c. <u>if</u> $d_i^i = 0$, then: same as <1a>;

2d. $\forall k$, set $d_i^k \leftarrow \max\{d_i^k, d^k\}$;

3. <u>if</u> $\forall \ell' \in G_i$, holds $N_i(\ell') = 1$, then:

3a. $d_i^i \leftarrow 2$; $\forall \ell' \in G_i$, set $N_i(\ell') \leftarrow 0$; send D_i to all neighbors;

4. <u>if</u> $D(\ell) \neq \{0,0,\ldots,0,1,0,\ldots,0\}$ and $M_i = 1$, then:

4a. <u>if</u> $d_i^i \neq 2$, then $\forall k$ set $d_i^k \leftarrow \max\{d_i^k, d^k\}$

5. <u>else</u>, <u>if</u> $\exists j$ such that $d^j = 2 > d_i^j$, then:

5a. $\forall k$ set $d_i^k \leftarrow \max\{d_i^k, d^k\}$; send D_i to all neighbors.

6. <u>if</u> $\forall j$ holds $d_i^j = 2$ or 0, then $M_i \leftarrow$ NORMAL

Observe that the message {0,0,...0,1,0,...0} replaces WAKE of protocol CT3. Note also that Finn's [3] protocol requires a node to send messages every time its table is updated, while here messages are sent only when relevant new information is received (see <4a>, <5>). In this sense, the present version is more efficient than [3]. The properties of the protocol are summarized in

<u>Theorem CT4</u>

Suppose START is delivered to some node. Then

a) exactly one message {0,0,...0,1,0,...,0} traverses each link in each direction and this is the first message on each link;

b) <3> happens at all connected nodes exactly once and then $d_i^j \geq 1$ for all neighbors j of i;

c) no more than $|V|$ messages with format $\neq$ {0,0,...,1,0,...0} traverse each link in each direction;

d) same as e) in Theorem CT3.

<u>Proof</u>

Lines <1>, <2c> and the fact that the propagation is as in PI imply a) and b). From the algorithm it is clear that d_i^j can only increase and from <3a> and <5a> follows that a message $\neq$ {0,0,...,1,0,...0} can be sent by i only when some d_i^j is set from 0 or 1 to 2 and this can happen only once for each j. Hence c). Finally d) follows in the same way as e) in Theorem CT3.

Communication cost

Each message contains $2|\overline{V}|$ bits and hence at most $2|\overline{V}|\ (|V| + 1)$ bits will travel on each link in each direction.

Protocol CT3 can be improved in another way, resulting in a more efficient protocol CT5.

Protocol CT5

Consider protocol CT3 with the following variation:
Whenever receiving $MSG^j(K_j)$, a node i consults its table containing $\{d_i^k\}$. If $d_i^j = 2$, the MSG is discarded, since such a MSG has been previously received and forwarded to all neighbors; this part is the same as in CT3. If $d_i^j < 2$, then $d_i^j \leftarrow 2$ and the MSG is sent to all neighbors, but now, before sending $MSG^j(K_j)$, *the following pruning operation is performed.*:

For all $k \in K_j$, if $d_i^k \geq 1$, then k is deleted from K_j; otherwise k is not deleted from K_j and the variable d_i^k receives value 1. Then $MSG^j(K_j)$ is sent to all neighbors. Node k can indeed by deleted when $d_i^k \geq 1$ because in this case k has been sent before by i to neighbors, either as a neighbor of some node, in which case, $d_i^k = 1$ or in MSG^k, in which case $d_i^k = 2$. One way or the other, there is no need to send k again. All properties of CT3 hold here as well, but the pruning operation assures that the identity of each node k travels no more than twice on each link in each direction: once as a neighbor of some node and once in MSG^k. Hence the communication cost is bounded by $2|V|\log_2|\overline{V}|$ bits per link in each direction.

5. Minimum-hop-path protocols

The problem considered next is to obtain the paths with smallest number of links (hops) from each node to each other node. As before, at the beginning of the algorithm a node knows only its own identity and the adjacent links. When the algorithm is completed at a node i, we want the node to know its distance d_i^k in terms of number of links to all other nodes to which it is connected and a preferred neighbor p_i^k through which it has the minimum-hop path to k. Observe that we do not require nodes to know the entire minimum-hop path.

If the travel time of a control message were identical on all links, then we could have accomplished the minimum-hop-path by using protocol PI (see Theorem PI-1 c)). However, as stated before; such an assumption is not practical, and the problem is to design a DNP where nodes will receive the first message with a given identity from the neighbor providing the shortest path, even if link delays are arbitrary. Such a protocol has been proposed by Gallager [1] and here we give its formal description and validation.

Protocol MI

A node enters the algorithm in the same way as in the CT protocols, namely when receiving START or the first control message, and at that time is sends its own identity to all neighbors. After having received the identity of all neighbors, node i knows all nodes that are at distance 1 from it. Node i keeps this information, sends it to all neighbors and then waits to receive the lists of all nodes that are at distance 1 from each of its neighbors. The union of these lists minus the set of nodes already known to i, i.e. those that are at distance 0 or 1 from it, is exactly the set of nodes that is at distance 2 from i. This information is kept again at i and also distributed to neighbors, and the procedure is repeated. If at some level, the union of the lists received from all neighbors contains no nodes that are unknown to i, then node i has completed the algorithm. It sends to all neighbors a message saying that it has no new node identities to send and stops. Any further message it may receive is disregarded.

Variables of the algorithm at node i

d_i^k - distance from i to k; set initially to $|\bar{V}|$ for all k (values $0,1,\ldots|\bar{V}|$);

p_i^k - preferred neighbor from i to k, for all k;

Z_i - state of node i showing distance covered by the protocol up to now (values $-1,0,1,\ldots,|\bar{V}|-1$);

M_i shows if node i is currently participating in the protocol (values NORMAL, WORK);

$N_i(\ell)$ - level of last message received on link (i,ℓ) (values $-1,0,\ldots,|\bar{V}|-1$), for $\ell \in G_i$.

Messages sent and received at node i

MSG($LIST_i$) - message sent by node i

MSG (ℓ,LIST) = MSG(LIST) received on link (i,ℓ)

START

Algorithm for node i

Assumpti n: just before node i enters algorithm, it has p_i^k = nil,

$$d_i^k = |\bar{V}| \text{ for all } k,\ Z_i = N_i(m) = -1 \text{ for all } m \in G_i.$$

1. For START or MSG(ℓ,LIST)

2. <u>if</u> $Z_i = -1$, then: $d_i^i \leftarrow 0$; $M_i \leftarrow$ WORK; $Z_i \leftarrow 0$; $LIST_i = \{i\}$; send MSG($LIST_i$) to all $m \in G_i$.

3. <u>if</u> MSG and $M_i = 1$, then

4. $N_i(\ell) \leftarrow N_i(\ell) + 1$;

5. $\forall k \in$ LIST, then

5a. <u>if</u> $d_i^k > N_i(\ell) + 1$, then $d_i^k \leftarrow N_i(\ell) + 1$; $p_i^k \leftarrow \ell$.

6. <u>if</u> $Z_i \leq N_i(m)$, $\forall m \in G_i$, then

6a. $Z_i \leftarrow Z_i + 1$; $LIST_i = \{k | d_i^k = Z_i\}$; send MSG($LIST_i$) to all neighbors;

7. <u>if</u> $LIST_i = \phi$, then $M_i \leftarrow$ NORMAL.

Preliminary properties of the protocol are given in Lemma MH-1, while the main properties appear in Lemma MH-2 and in Theorem MH-1.

<u>Lemma MH-1</u>

Suppose START is delivered to a node (or several nodes). Then for any connected node i holds:

a) i will enter the protocol in finite time;

b) messages are sent by node i if and only if Z_i is incremented at the same time; if MSG is sent by i while $Z_i = Z$, receipt of the MSG at neighbor ℓ will cause $N_\ell(i) \leftarrow Z$;

c) Z_i and $N_i(m)$ for each $m \in G_i$ change only by increments of +1;

d) for each $m \in G_i$, holds $N_i(m) = Z_i$ or $Z_i \pm 1$ and there is at least one m for which $N_i(m) = Z_i - 1$ (note: this implies $Z_i = \min_m N_i(m) + 1$);

e) no message can arrive on links (i,m) for which $N_i(m) = Z_i + 1$;

f) if Z_i is incremented at time t, then for all $m \in G_i$ holds $N_i(m)(t+) = Z_i(t+)$ or $Z_i(t+) - 1$.

<u>Proof</u>

a) holds since propagation of <2> happens as in PI. Assertion b) holds since Z_i is incremented whenever MSG is sent (<2>,<6>), $N_i(\ell)$ is incremented whenever MSG is received from ℓ(<4>) and both are initialized to -1. In addition, c) follows from <2>, <4> and <6a>. Property d) is true immediately after the time node i enters the algorithm, at which time either $Z_i = 0$ and $\min_m N_i(m) = -1$, or $Z_i = 1$ and $\min_m N_i(m) = 0$, the latter if i has only one neighbor and enters the algorithm by receiving MSG from it. Suppose now that the property is true at node i up to time t- and we want to show that it will hold at time t+ as well. The variables $N_i(\cdot)$ or Z_i can change at time t only if a MSG is received, from neighbor ℓ say. Let[4] $Z_i(t-) = Z$. We have several cases:

i) $N_i(\ell)(t-) = Z - 1$ and $\exists m \neq \ell$ with $N_i(m)(t-) = Z - 1$; then $N_i(\ell)(t+) = Z_i(t+) = Z$ and all other $N_i(\cdot)$ do not change, hence d) continues to hold at time t+;

ii) $N_i(\ell)(t-) = Z - 1$ and $\nexists m \neq \ell$ with $N_i(m)(t-) = Z - 1$; then $N_i(\ell)(t+) = Z$ and $Z_i(t+) = Z + 1$, since <6> holds at t, and d) continued to hold at t+;

iii) $N_i(\ell)(t-) = Z$, in which case $N_i(\ell)(t+) = Z + 1$ and $Z_i(t+) = Z$, hence d) continues to hold at time t+;

iv) we claim that $N_i(\ell)(t-)$ cannot be $Z + 1$. Suppose $N_i(\ell)(t-) = Z + 1$. Then $N_i(\ell)(t+) = Z + 2$, and from b) follows that at time $t1 < t$, node ℓ has sent MSG($LIST_\ell$) while $Z_\ell = Z + 2$. From <6>, <6a> we have $Z_\ell(t1-) = Z + 1$ and $N_\ell(i)(t1+) \geq Z + 1$. This means that $\exists t2 < t1$ when i has sent MST(LIST) to ℓ, while $Z_i(t2+) = Z + 1$. But the latter and $Z_i(t-) = Z$ contradicts the monotonicity of Z_i (see c)).

This completes the proof of d). Observe now that e) is exactly case iv) in d). Finally, observe that scanning cases i) - iv) of d), we see that Z_i is incremented only in case ii) and f) clearly holds in this case, completing the proof of the Lemma.

Definition

The number of links on the minimum-hop path from i to k is called the hop-distance from i to k.

Lemma MH-2

Under the same conditions as in Lemma MH-1, holds:

a) if a node i has nodes at hop-distance r, then it sets $Z_i \leftarrow r$ in finite time and then sends MSG($LIST_i$), where $LIST_i$ contains exactly all nodes k that are at hop-distance r; for all those nodes holds $d_i^k = r$ and this d_i^k is final.

b) let S_i be the largest hop-distance from node i in the network, i.e. node i does have nodes at hop-distance S_i, but not at hop-distance $(S_i + 1)$; then node i will set $Z_i \leftarrow (S_i + 1)$ in finite time, at which time it sends MSG($LIST_i$) with $LIST_i = \phi$ and performs <7>; node i will not increase Z_i any further.

Proof

a) Setting of $Z_i \leftarrow 0$ while sending MSG($LIST_i$) with $LIST_i = \{i\}$ propagates as in PI and hence will happen at all nodes in finite time. How suppose a) holds for all nodes that have nodes at hop-distance (r-1). Consider a node i that has nodes at hop-distance r. Then itself and all its neighbors m have nodes at hop-distance (r-1) and by the induction hypothesis, they set $Z_m \leftarrow (r-1)$ and send MSG($LIST_m$). When such a message arrives at i, it sets $N_i(m) \leftarrow (r-1)$ and after all such messages arrive, <6> will hold with $Z_i = (r-1)$. This causes $Z_i \leftarrow r$. At this time we have from Lemma MH-1, $N_i(m) = r$ or $(r-1)$ for all m.

Now suppose k is at hop-distance r from i. Then there is a neighbor m of i such that k is at hop-distance (r-1) from m and there is no neighbor m of i such that k is at hop-distance strictly less than (r-1) from m. By the induction hypothesis, k was sent by m in MSG($LIST_m$) while $Z_m \leftarrow (r-1)$ and hence was received at i while $N_i(m) \leftarrow (r-1)$, but was sent by <u>no</u> neighbor m' while $Z_{m'} \leftarrow Z < (r-1)$. Hence at the time $Z_i \leftarrow r$ we have $d_i^k = r$, and therefore k is sent in MSG($LIST_i$). From <5a> it is clear that this d_i^k is final. A similar argument shows that nodes at hop-distance $>r$ or $<r$ from i cannot be included in the $LIST_i$ considered above.

b) First consider a node i s.t. $S_i = \min\{S_j\}$ where the min is over all nodes in the network. All its neighbors m have nodes at distance S_i and by a) they send MSG($LIST_m$) while $Z_m \leftarrow S_i$. When all these messages arrive to i, Z_i will become $S_i + 1$, but since i has no nodes at hop-distance $S_i + 1$, holds $LIST_i = \phi$ and hence i performs <7>. Now suppose by induction that b) holds for all nodes i for which $S_i \leq S - 1$. Consider a node j with $S_j = S$. Node j has a node k at hop-distance S and k is included in $LIST_j$ when j sends MSG($LIST_j$) while $Z_j \leftarrow S$. For an arbitrary neighbor m of j, node k is at hop-distance (S-1), S or (S+1) from m and hence $S_m \geq S-1$. If $S_m \geq S$, then a) implies that Z_m will become S in finite time. If $S_m = S-1$, then Z_m will become S in finite time from the induction hypothesis. Hence from Lemma MH-1 b), $N_j(m)$ will become S in finite time for all neighbors m of j and hence Z_j will become (S+1). Since j has no nodes at hop-distance (S+1), <7> will hold and this completes the proof.

From the previous Lemmas, we obtain the following:

<u>Theorem MH-1</u>

If START is delivered to a node (or to several nodes), then all connected nodes will enter the protocol in finite time. All nodes i will complete the protocol in finite time with correct d_i^k and p_i^k for all connected nodes k and with $d_i^k = |\overline{V}|$, p_i^k = nil for all disconnected nodes.

Proof

The only unproven part is the setting of p_i^k , which however follows immediately from the proof of Lemma MH-2 a).

Communication cost

Since the identity of any node travels exactly once on each link, we need $|V| \log_2 |\bar{V}|$ bits on each link in each direction.

6. Path-updating protocols

In the protocol of [2], [4] each node maintains a path to each other node in the network and updating "cycles" allow these paths to be changed so that they are improved in each cycle and, in addition, the collection of paths to any given node form at any given time a loop-free pattern (i.e. a tree). Here we present first the fixed-topology part of the path-updating protocol and then show that protocol CT2 can be used to initialize it. The validation of both is based on the PIF basic protocol.

Protocol PU

The protocol updates paths from all nodes in the network to a given node s and can be repeated independently to update paths to each of the other nodes. Therefore, *we can present only the protocol for a given* "destination" node s. The protocol works very similar to the PIF protocol. Here however it is assumed that just before START is delivered to s, all connected nodes i already have preferred neighbors p_i^s such that the collection of the links (i, p_i^s) form a directed tree rooted at s. We also assume that at that time, all $m_i^s = 0$ and in addition the variables d_i^s as defined below are such that $d_i^s > d_{p_i^s}^s$, or in words d_i^s is strictly decreasing while moving downtree. Finally, it is assumed that $N_i^s(\ell) = 0$ for all $\ell \in G_i$. The validation of the protocol (theorem PU-1) will show that these properties continue to hold when the protocol is completed, so that a new update cycle can then be started.

Protocol PU

Variables of the algorithm at node i ≠ s

$N_i^s(\ell)$, same as in protocol PIF ;

$d_{i\ell}$ - distance from node i to neighbor ℓ as measured at the time it is needed by the algorithm; can be time-varying (values: any strictly positive real number), $\ell \in G_i$;

d_i^s - estimated distance from i to s on the preferred path;

p_i^s - "preferred" neighbor of i for s;

$D_i^s(\ell)$ - storage for $d_\ell^s + d_{i\ell}$, for $\ell \in G_i$;

m_i^s = 1 after performing <3> and before performing <4> ; = 0 otherwise.

Messages received and sent by the algorithm at i

$MSG^s(d_i^s)$ - message sent;

$MSG^s(\ell, d^s)$ - message received.

Algorithm for node i ≠ s

1. For $MSG^s(\ell, d^s)$

2. $N_i^s(\ell) \leftarrow 1$; $D_i^s(\ell) \leftarrow d^s + d_{i\ell}$.

3. if $\ell = p_i^s$, then: $d_i^s \leftarrow \min D_i^s(\ell')$ over ℓ' s.t. $N_i^s(\ell') = 1$; $m_i^s \leftarrow 1$; send $MSG^s(d_i^s)$ to all neighbors, except p_i^s.

4. if $\forall \ell' \in G_i$ holds $N_i^s(\ell') = 1$, then: send $MSG^s(d_i^s)$ to p_i^s ; $p_i^s \leftarrow k^*$ that achieves $\min D_i^s(\ell')$ over $\ell' \in G_i$; $m_i^s \leftarrow 0$; $\forall \ell' \in G_i$, set $N_i^s(\ell') \leftarrow 0$.

The algorithm for s is the same as in PIF except that all messages sent by s have format $MSG^s(0)$.

Theorem PU-1

Suppose the assumptions given just before the presentation of the protocol hold. Then:

a) theorem PIF-1, where p_i^s refers (only in this part) to the initial preferred neighbors.

b) the collection of links $\{(i, p_i^s)\}$ forms at all times a tree rooted at s with the following properties:

(i) $m_i^s \leq m_{p_i^s}^s$

(ii) if $m_i^s = m_{p_i^s}^s = 0$, then $d_i^s > d_{p_i^s}^s$.

c) for each link (i,ℓ) the "distance" $d_{i\ell}$ is measured exactly once by node i; at the end of the protocol, all nodes will have paths to s that are no longer than before the protocol starts, where the length of a path is the sum of the weights of the links; if *initially the tree defined by* $\{p_i^s\}$ *is not* identical to the minimum-weight-tree in terms of the measured $\{d_{i\ell}\}$, then there is a nonempty set of nodes that will strictly improve their paths.

Proof

Observe that the present protocol is identical to PIF, except that <3> is performed by a node i only when MSG is received from p_i^s (and not as soon as the first MSG is received as in PIF), we introduce the quantities d_i^s, $D_i^s(\ell)$, $d_{i\ell}$ and the preferred neighbor p_i^s is changed in <4>. Now, if we denote by $P1^s$ the initial tree, <3> and <4> propagate here exactly as in PIF, provided that in that protocol a MSG traverses any link in $P1^s$ much faster than any other link. Since Theorem PIF-1 holds for arbitrary link travel times, assertion a) follows. In order to prove b), suppose the assertions hold up to time t- and we want to show that if <3> or <4> happens at time t at some node i, the assertion continues to hold.

Observe that if <3> happens at node i at time t, then p_i^s is not changed and hence the tree property continues to hold. Also, b) ii) is not affected by <3> and hence we only have to check that b) i) continues to hold.

Since $m_i^s(t-) = 0$, we have by the induction hypothesis $m_j^s(t) = 0$ for any j for which $p_j^s(t) = i$ and hence b) i) continues to hold for j and i after time t. On the other hand, when performing <3>, node i receives MSG^s from p_i^s, so that p_i^s must have performed <3> before t, implying that $m^s_{p_i^s}(t) = 1$ and, since $m_i^s(t+) = 1$, assertion b) i) continues to hold after t for i and p_i^s as well.

Now suppose <4> happens at some node i at time t. Observe that at that time, i has already received MSG from all neighbors and it performs $m_i^s \leftarrow 0$. Consider first any node j such that $p_j^s(t) = i$. If $p_j^s(t_o) = i$, where t_o is the time the protocol started, then receipt of MSG at i from j means that j has performed <4> before time t. If $p_j^s(t_o) \neq i$, then j has changed p_j^s before time t and again this shows that it has performed <4> before time t. Consequently $m_j^s(t) = 0$ and hence b) i) continues to hold after time t for j and i. Also, from the way d_j^s is calculated and p_j^s is chosen follows that

$$d_j^s \geq D_j^s(i) = d_i^s + d_{ji} > d_i^s , \qquad (6.1)$$

where the last inequality follows from the assumption $d_{ji} > 0$ (see definition of $d_{i\ell}$). Consequently b) ii) continues to hold after time t. Now, consider the pair i and $k^* = p_i^s(t+)$. Assertion b) i) holds trivially after t for i and k* since $m_i^s \leftarrow 0$, while assertion b) ii) holds by the same argument as in (6.1). Now (i,k*) cannot close a loop since by b) i), all nodes ℓ in such a loop must have $m_\ell^s = 0$, and going around the loop this would imply by b) ii) that $d_i^s > d_i^s$. The proof of c) is quite simple and will be deleted here. The reader is referred to similar proofs that appear in [4, Sec.4] and [6, Appendix, Lemma 1].

<u>Communication cost</u>

Clearly there is exactly one message on each link in each direction. Its size in bits depends on the number of bits assigned to d_i^s.

Protocol PUI (path-updating initialization)

In order to allow proper evolution of the PU protocol, it is necessary to initialize it in the sense of building the initial trees $\{(i, p_i^j)\}$ for all "destinations" j in the network. This can be done by using protocol CT2 and we shall give here the additions that allow initialization of protocol PU.

Variables used by the algorithm at node i

Same as in CT2 except that d_i^j has the meaning as in PU and in addition m_i^j, $d_{i\ell}$, d_i^j, $D_i^j(\ell)$, $\forall j$ and $\ell \in G_i$, with the same meaning as in PU.

Messages sent and received by the algorithm at node i

Same as in PU and in addition, START.

Algorithm for node i

Assumption: just before node i enters the algorithm, it has

$$N_i^j(\ell) = m_i^j = 0 \text{ for all } j \text{ and } \ell \in G_i.$$

1. For START or $MSG^j(\ell, d^j)$, $j \neq i$.

1a. if MSG, then: $N_i^j(\ell) \leftarrow 1$; $D_i^j(\ell) \leftarrow d^j + d_{i\ell}$.

2. if $m_i^i = 0$, then: $M_i \leftarrow$ WORK; $m_i^i \leftarrow 1$; send $MSG^i(0)$ to all neighbors.

3. if MSG and $m_i^j = 0$, then: $p_i^j \leftarrow \ell$; $d_i^j \leftarrow D_i^j(\ell)$; send $MSG^j(d_i^j)$ to all neighbors except p_i^j.

4. if $\nexists \ell' \in G_i$, holds $N_i^j(\ell') = 1$, then: send $MSG^j(d_i^j)$ to p_i^j; $p_i^j \leftarrow k^*$ that achieves min D_k^j over $k \in G_i$; $m_i^j \leftarrow 0$; $\forall \ell' \in G_i$, set $N_i^j(\ell') \leftarrow 0$.

5. For $MSG^i(\ell, d^i)$

5a. $N_i^i(\ell) \leftarrow 1$;

6. <u>if</u> $\ell' \in G_i$, holds $N_i^i(\ell') = 1$, then: $M_i \leftarrow$ NORMAL; $m_i^i \leftarrow 0$; $\ell' \in G_i$, set $N_i^j(\ell') \leftarrow 0$.

<u>Theorem PUI-1</u>

Suppose START is delivered to any node. Then any given node j will perform <6> in finite time and at that time the links $\{(i, p_i^j)\}$ will form a directed tree rooted at j, with the property $d_i^j > d^j_{p_i^j}$ for all i. In addition, at that time, all $m_i^j = 0$, and all $N_i^j(\ell) = 0$.

<u>Proof</u>

The protocol here evolves as CT2 and hence all properties of CT2 hold here. Also, for a given j, action $<3>^j$ evolves as in PI, so that Theorem PI-1 c) holds. Consequently, $\{(i, p_i^j)\}$ as considered after all nodes perform $<3>^j$ form a tree rooted at j. Also, by <1a> and <3> and the fact $d_{i\ell} > 0$, the quantities d_i^j are strictly decreasing going downtree. After $<3>^j$ is performed at all nodes, the protocol for j behaves as in PU, so that all properties continue to hold until j performs <6> .

7. Topological changes

The protocols presented so far assume fixed topology of the network. As such, the CT and MI protocols may be performed only once and similarly, the path-updating protocol may be initialized only once (the PU protocol itself should be repeated periodically to account for load variations). In this section, we present extensions to the above protocols that take into consideration failures and additions of links and nodes, the main idea being that whenever a topological change is sensed at some node, a new "cycle" of the protocol is triggered to inform the network of the new situation. Since we are working with a distributed network, we can make no a priori assumption regarding the number, sequence or timing of topological changes, and as such the extended protocols must work for all circumstances.

With topological changes occurring in the network, the assumptions of Sec. 2 should be changed accordingly. In particular, assumptions a) and c) of Sec. 2 will be changed now to:

a') link (i,j) fails/recovers at the same time that link (j,i) fails/ recovers, so that (i,j) belongs to the network iff (j,i) belongs to the network;

c')i) each message sent by node i on link (i,j) arrives correctly in finite nonzero undertermined time or the link fails in finite time;

ii) whenever a link fails or recovers, both ends are notified in finite time, but not necessarily at the same time;

iii) failure or recovery of a node is considered as failure/recovery of all adjacent links;

To make ii) above more precise, let $F_i(\ell)$ denote a flag indicating the status of link (i,ℓ) as seen from node i, taking values DOWN or UP if i considers link (i,ℓ) as down or up respectively. Then we assume:

- if $F_i(\ell) = F_\ell(i)$ and $F_i(\ell) \leftarrow$ DOWN, then $F_\ell(i)$ becomes DOWN in finite time and before $F_i(\ell) \leftarrow$ UP;

- if $F_i(\ell) = F_\ell(i) =$ DOWN and $F_i(\ell) \leftarrow$ UP, then in finite time holds either $F_\ell(i) \leftarrow$ UP or $F_i(\ell) = F_\ell(i) =$ DOWN.

Now the idea for extending DNP's described in the previous sections to account for topological changes is the following: the cycles of the protocol will be labelled with increasing numbers, every node remembers the highest cycle number known to it so far and each of the cycles corresponds now to the original (nonextended) protocol. When a node wants to trigger a new cycle as a result of detecting a topological change in an adjacent link, it resets its variables, increments the cycle number and acts as if it has received START for a new cycle with this number. Here "reseting variables" means to adjust the appropriate variables to their required initial value as stated in the corresponding assumption in each of the algorithms (e.g. in MH, $p_i^k \leftarrow$ nil, $d_i^k \leftarrow |\overline{V}|$ for all k and $Z_i \leftarrow -1$, $N_i(m) \leftarrow -1$ for all adjacent m). The number of the new cycle will be carried by all messages belonging to this cycle and now, any node receiving a message with cycle numbers lower than the one known to it so far discards this message. A node receiving a message with higher cycle number than the highest known to it, resets its own variables, increases its remembered cycle number accordingly and acts as if it now enters the algorithm (i.e. the corresponding cycle of the extended protocol). In this way the cycle with higher number will "cover" the lower number ones, in the sense that when a higher cycle reaches any node, the node will forget the previous knowledge and will participate only in the most "recent" cycle. Observe that several nodes may start the same new cycle independently, but the protocol allows this situation to happen, considering it in the same way as if several nodes receive START in the nonextended protocol.

There is a question, whether it is indeed necessary for all nodes to forget their entire previous knowledge, or rather it is possible to design protocols where only the information affected by the topological change is discarded. For the PU protocol, such a protocol appears in [2], [4], [7] but for the others this is still an open question.

As an example, we shall write exactly the extended MH protocol.

Protocol EMH (extended MH) - Version A)

Variables used by the algorithm at node i

Same as in MH, and in addition:

R_i - highest cycle number known to i (values: 0,1,...)

$F_i(\ell)$ - status of link (i,ℓ) as known by i (DOWN, UP)

Messages sent and received at node i

$MSG(R_i, LIST_i)$ - sent.

$MSG(\ell,R, LIST) = MSG(R, LIST)$ received on link (i,ℓ).

Algorithm for node i

Definition: "reset variables" means $p_i^k \leftarrow nil$, $d_i^k \leftarrow |\bar{V}|$ for all k, $Z_i \leftarrow -1$, $N_i(\ell) \leftarrow -1$ for all ℓ for which $F_i(\ell) = UP$.

1. Node i becomes operational (Note: Node i becoming operational forces all operating links (i,ℓ') with ℓ' operating, to become operational for ℓ')

1a. $F_i(\ell') \leftarrow UP$ for all operating adjacent links (i,ℓ') with ℓ' operating;

1b. $F_i(\ell') \leftarrow DOWN$ for all nonoperating adjacent links (i,ℓ') or those links (i,ℓ') with ℓ' nonoperating;

1c. reset variables; $R_i \leftarrow 0$; $M_i \leftarrow NORMAL$.

2. Adjacent link (i,ℓ) becomes operational or fails

2a. $R_i \leftarrow R_i + 1$;

2b. $F_i(\ell) \leftarrow$ DOWN or UP according to new status;

2c. reset variables; $M_i \leftarrow$ WORK; $d_i^i \leftarrow 0$; $Z_i \leftarrow 0$; $LIST_i = \{i\}$; send $MSG(R_i, LIST_i)$ to all m for which $F_i(m) = UP$.

3. For $MSG(\ell,R,LIST)$

3a. if $R \geq R_i$, then

3b. if $R > R_i$, then: $R_i \leftarrow R$; same as <2c>;

3c. if M_i = WORK, then:

4.-7. same as <4>-<7> in MH, except that MSG has format $MSG(R_i, LIST_i)$.

Note that <2> and <3b> here correspond to <2> in MH, while <3c> corresponds to <3> in MH. Clearly, similar extended protocols can be given for the CT and also for the PUI protocols. Their properties are similar to the ones of EMH-Version A as summarized in:

Theorem EMH-A-1

Consider an arbitrary finite sequence of topological changes with arbitrary timing and location. Within finite time after the sequence is completed, all nodes i in the final connected network will have M_i = NORMAL with the same cycle number R_i, with correct d_i^k and p_i^k for all connected nodes k and with $d_i^k = |\overline{V}|$, p_i^k = nil for all disconnected nodes k.

Proof

From <2a> and <1>, each topological change increments the cycle counter R_i at nodes i adjacent to the change. Every change in a link status affects two nodes, every change in a node status affects a finite number of nodes. Let $\{i_n\}$ be the collection of nodes that register change of status of an adjacent link, including those due to status changes of the node at the other end of a link, and let $\{t_n\}$ be the corresponding collection of times when the status change is registered. Since there is a finite number of topological changes, the collections $\{i_n\}$, $\{t_n\}$ are finite. Let $R = \max \{R_{i_n}(t_n+)\}$ over all n. Then R is the highest cycle number ever known in the network and the cycle with number R is started by (one or more) nodes $i \varepsilon \{i_n\}$ that increment their R_i to R as a result of sensing a topological change. These nodes can be considered as if they receive START in the MH protocol and, indeed, the network covered by the cycle with number R registers no more topological changes, since no counter number R_i is ever increased to (R+1). Consequently, the evolution of this cycle is the same as in protocol MH and therefore Lemmas MH-1, MH-2 and Theorem MH-1 hold here, completing the proof.

In version A of MHE as presented above, as well as all other similar extended protocols, there is the problem that the cycle counter numbers $\{R_i\}$ increases without bound and hence the question of how many bits are enough to represent this number. In [1], [3] the authors propose a modified version of the extended protocols that insure bounded counter numbers. We present this protocol here formally (version C) and give a new proof for its validation. The procedure is illustrated as before on Protocol MH, but similarly can be implemented on CT and PUI. In order to provide a framework for understanding Version C, it is convenient to first present and validate a non-distributed Version B that will introduce the important features of the procedure and then to explain the equivalence between Versions B and C.

Protocol MHE - Version B

Messages and variables: same as in version A.

Algorithm for node i

Same as in Version A, except for the following changes:

Lines <2> and <2a> become:

2. Adjacent link (i,ℓ) fails

2a. $R_i \leftarrow R_i + 1$ and proceed to <2b>.

2'. Adjacent link (i,ℓ) becomes operational

2a'. wait until $M_i = M_\ell =$ NORMAL and then

2a". if $R_i < R_\ell$, then: $R_r \leftarrow R_r + (R_\ell - R_i)$ for all nodes r that are connected to i and furthermore, in all messages that have been sent by such node r and not received yet, increase R by $(R_\ell - R_i)$

2a'". if $R_\ell < R_i$, similar to <2a"> for nodes connected to ℓ where the increment is $(R_i - R_\ell)$.

2a'V. $R_i \leftarrow R_i + 1$ and proceed to <2b>.

After <1c> insert

1d. for all ℓ' for which $F_i(\ell') =$ UP, proceed as in <2a'>.

The main property of Version B is given in Theorem MHE-B-1, but first we need several definitions:

Definitions

A link (i,ℓ) is said to be operating if $F_i(\ell) = F_\ell(i) =$ UP. Two nodes are said to be connected if there is a set of operating nodes and links connecting them. A set of nodes is said to be connected if every pair of nodes in the set is connected. A set of nodes S is said to be at level R if $\min R_i = R$ for $i \in S$. A set of nodes S (connected or not) is said to be synchronized if either a) or b) below holds:

a) all nodes $i \in S$ have M_i = WORK.

b) there is at least one node $i \in S$ with M_i = NORMAL and then holds:

i) $\forall j \in S$ with M_j = NORMAL holds $R_j = R_i$ and

ii) $\forall j \in S$ with M_j = WORK holds $R_j \geq R_i$.

Theorem MHE-B-1

In MHE - Version B, if at any time t a set of nodes S is connected, then it is also synchronized. Furthermore, if the set is at level R at time t and if any node j will be connected at any future time $t' > t$ to any node $i \in S$, then it will have $R_j(t') > R$. (Note: the first property is the important one; the second is only helpful in the proof).

Proof

We proceed by induction on events happening in the network. Suppose both properties above hold up to time t-. Explicitly, every set of nodes S' that was connected at any time $\tau < t$ was also synchronized at that time and every node j that was connected to any node in S' at any time between τ and t- had $R_j \geq$ level of S' at time τ. The events that can happen at time t and affect the properties of the Theorem are: (i) a node becomes operational, (ii) a link fails, (iii) a link is brought up and (iv) M_i and/or R_i is changed. We proceed to check each of the possibilities. First, a node that becomes operational will not connect to the rest of the network until <ld> holds, so that this case reduces to (iii). Second, if the set was synchronized at t- and a link fails, it will remain synchronized just after the failure, except that one has to take into consideration that the failure causes changes of M_i and R_i at the adjacent nodes. However these changes are treated in (iv). Observe next that

(iii) can happen only if $M_i(t-) = M_\ell(t-) =$ NORMAL, where (i,ℓ) is the link under consideration. Suppose first that i and ℓ do not belong to two disconnected sets at time t-. Then, since the set under consideration is synchronized at time t- by the induction hypothesis, it follows that $R_i(t-) = R_\ell(t-)$. Hence <2a''> and <2a'''> do not apply and therefore the only relevant variables that are changed are M_i, R_i, M_ℓ, R_ℓ (lines <2a^{iv}> and on), and this again reduces to (iv). Suppose now that the new link (i,ℓ) does connect two previously disconnected sets. If $R_i(t-) = R_\ell(t-)$, the same argument as before applies. If for example, $R_i(t-) < R_\ell(t-)$, let t' be the time just after execution of <2a''>, but before execution of <2a^{iv}>. Recall that $M_i(t-) = M_\ell(t-) =$ NORMAL and since each of the sets are synchronized at t-, we have $R_r(t-) \geq R_i(t-)$ for all r connected to i and $R_r(t-) \geq R_\ell(t-)$ for all r connected to ℓ, with equality in both cases for those nodes r that have $M_r(t-) =$ NORMAL. Now, from t- to t' all nodes r connected to i raise their cycle number R_r by $R_\ell(t-) - R_i(t-)$ and so do all messages in transient, and hence, the new combined set remains synchronized at t'. Now the transition from t' to t+ is execution of <2a^{iv}> and on, and again this reduces to case (iv), which is treated next. Observe that R_i is increased if and only if M_i is WORK or becomes WORK, and clearly if at t- the set was synchronized, it will remain so at t+. Therefore the only situation that remains to be treated is $M_i \leftarrow$ NORMAL, in which case R_i is not changed. Let R be the value of R_i at time t- (and t+). We must show that for all nodes $j \in S_t$, where S_t is the set of nodes connected to i at time t, we have $R_j(t) \geq R$, with equality if $M_j(t) =$ NORMAL. But since S_t is synchronized at t- by the induction hypothesis, the condition $M_j(t) =$ NORMAL requires $R_j(t) \leq R$ for all $j \in S_t$, and therefore it is sufficient to show that $R_j(t) \geq R$ for all $j \in S_t$. At time t, node i performs $M_i \leftarrow$ NORMAL and let P be the set of nodes k for which $d_i^k(t) < |\bar{V}|$. Nodes $k \in P$ certainly have $R_k(t) \geq R$. Now take any node α such that $\alpha \in S_t$, but $\alpha \notin P$. We want to show that $R_\alpha(t) \geq R$. Observe that there must exist a node $\beta \in S_t$, $\beta \notin P$ such that β is at time t a neighbor of a node $\gamma \in S_t \cap P$ (see Fig. 1). Since $\beta \notin P$, node β was disconnected from γ at some time after $R_\gamma \leftarrow R$. Let $S_{\tau-}^\gamma$ be the connected set containing γ at time $\tau-$, where $\tau < t$ is the time when link (β,γ) was brought up. At that time M_γ was NORMAL and $R_\gamma \geq R$ and by the induction hypothesis, $S^\gamma_{\tau-}$ was at level $\geq R$. In addition, by the second assertion of the Theorem that holds at time $\tau-$ because of the induction hypothesis, the fact that α is connected at time $t > \tau$ to $\gamma \in S^\gamma_{\tau-}$ implies $R_\alpha(t) \geq$ level of $S^\gamma_{\tau-}$ at $\tau- \geq R$. This completes the proof of case (iv) and shows that the connected set remains synchronized. It remains to prove that any node that will become connected to any node in the considered connected set S_t will have at that time counter number $\geq R$.

At time t, every node $i \in S_t$ has $R_i \geq R$ and R_i never decreases. Let t' be the first time after t when a node j' becomes connected to any node $i \in S_t$. Since until that time all connected sets are synchronized, it must hold that $R_{j'}(t') \geq R$ by the same argument as in (iii) above. Consequently, after t' all sets remain synchronized and the same argument shows that the property remains true for all future connections, completing the proof of the Theorem.

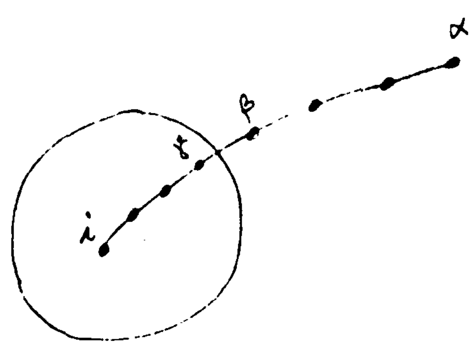

Fig. 1

Having proved the main properties of Version B, we can now make a few observations about this (non-distributed) Version that will allow us to introduce an equivalent distributed version.

Lemma MHE-B-1

In MHE - Version B, the following properties hold:

a) if <2a'> holds at time t and nodes i and ℓ are connected at time t-, then $R_i(t-) = R_\ell(t-)$ and hence <2a"> - <2a'"> are not performed.

b) Theorem MHE-A-1 holds for version B as well.

c) for any node r, R_r is nondecreasing and, unless R_r is increased by <2a"> or <2a'">, it has increments of +1; if a node i sends two consecutive messages on a given link with counter numbers R', R" respectively, then $R'' \geq R'$ and if the second message is not related to the performing of <2a"> or <2a'">, then $R'' = R'$ or $R'' = R' + 1$.

d) if node i receives MSG (ℓ,R, LIST) and $R > R_i$, then LIST = {ℓ} and this message was sent by ℓ while increasing R_ℓ, i.e. either in <2c> or <3b>, but not in <6b>.

e) the values of R and R_i are not necessary for the algorithm; we only need to know if $R < R_i$, $R = R_i$ or $R > R_i$.

Proof

a) follows Theorem MHE-B-1. Part b) can be proved exactly as Theorem MHE-A-1. For c), the fact that R_r and the counter numbers in consecutive messages can only increase is obvious from <2a>, <3b>, <2a">, <2a'">. The rest can be proved by a common induction as follows: suppose both properties hold until time t-. The counter R_r can be increased at time t only in <2a> and <3b> and in both cases only by +1. Furthermore, the message with R" can be sent by i only in <2a> while incrementing R_i by 1, in <3b> while incrementing R_i to R which is exactly $R_i + 1$ by the induction hypothesis, or in <6b> while maintaining R_i to the previous level. This proves both c) and d). Finally e) is clear from the algorithm.

Protocol MHE - Version C

Variables used by the algorithm at node i

Same as in MH and in addition $F_i(\ell)$ as in Version A and:

$Q_i(\ell)$ whose meaning is $R_i - XR_i(\ell)$ where $XR_i(\ell)$ is the largest counter number received from neighbor ℓ in version B.

Messages sent and received at node i

MSG (ΔR_i, $LIST_i$) - sent, where ΔR_i has the meaning of the difference between R_i in Version B and last R_i sent on this link.

MSG (ℓ,ΔR,LIST) - received.

Algorithm for node i

Definition: "reset variables" has the same meaning as in version A.

1. Node i becomes operational (same Note as in Version A)

1a.-1b. same as in Versions A and B;

1c. reset variables; $Q_i(\ell') \leftarrow 0$ for all ℓ for which $F_i(\ell') = UP$; $M_i \leftarrow NORMAL$.

1d. if there is an operational link (i,ℓ') for which $M_{\ell'} = NORMAL$, proceed as in <2a'>; else wait until this happens and then proceed as in <2a'>.

2. Adjacent link (i,ℓ) fails

2a. $Q_i(\ell') \leftarrow Q_i(\ell') + 1$ $\forall \ell' \neq \ell$ for which $F_i(\ell') = UP$; proceed to <2b>.

2'. Adjacent link (i,ℓ) is operational and $F_i(\ell) = F_\ell(i) = DOWN$ and $M_i = M_\ell = NORMAL$.

2a'. $Q_i(\ell) \leftarrow 0$; $Q_i(\ell') \leftarrow Q_i(\ell') + 1$ $\forall \ell'$ for which $F_i(\ell') = UP$;

2b. $F_i(\ell) \leftarrow$ DOWN or UP according to new status;

2c. reset variables; $M_i \leftarrow$ WORK; $d_i^i \leftarrow 0$; $Z_i \leftarrow 0$; $LIST_i = \{i\}$;
send MSG(1,$LIST_i$) to all m for which $F_i(m) = UP$.

3. For MSG(ℓ,ΔR,LIST)

3'. <u>if</u> $\Delta R < Q_i(\ell)$, then: $Q_i(\ell) \leftarrow Q_i(\ell) - \Delta R$.

3a. <u>else</u>

3b. <u>if</u> $\Delta R > Q_i(\ell)$ (note: i.e. $\Delta R = 1$, $Q_i(\ell) = 0$), then:

3b'. $Q_i(\ell') \leftarrow Q_i(\ell') + 1$ $\forall \ell'$ for which $F_i(\ell') = UP$;
same as <2c>;

3b:. $Q_i(\ell) \leftarrow 0$;

3c. <u>if</u> M_i = WORK, then

4.-7. same as in <4> - <7> in MH, except that MSG has format
MSG(0,$LIST_i$).

We have numbered the lines in Version C to correspond to the appropriate lines in Version B. The note appearing in <3b> holds because of Lemma MHE-B-1 c). Observe that <2a'> in Version C is equivalent to <2a">, <2a'"> of Version B. This is because if i and ℓ are connected at time t-, where t is the time of the event occurence, then in version B, $R_i(t-) = R_\ell(t-)$ from Lemma MHE-B-1 a) and <2a'> in version C says exactly the same thing. If, on the other hand, i and ℓ are disconnected at time t-, the effect of bringing $R_i(t-)$ and $R_\ell(t-)$ to the same level while raising accordingly all appropriate counter numbers is equivalent to <2a'> of Version C. This implies that Versions B and C are equivalent. Furthermore, Version C is distributed and the counter numbers are bounded as shown below.

Theorem MIE-C-1

a) The counter numbers ΔR in MSG take values 0 and 1 only.

b) For every i and ℓ, the variable $Q_i(\ell) \leq |E|$ where $|E|$ is the number of links in the network.

Proof

All messages sent in the algorithm have $\Delta R = 0$ or 1 and this proves part a). To see that b) holds, observe that $Q_i(\ell)$ can increase only if, while link (i,ℓ) is operating, node i keeps sending MSG(1,$LIST_i$) to ℓ, but ℓ *does not* respond. After the cycle corresponding to the first of these messages covers the entire netowrk (or is covered by another cycle), no link can be brought up, since lack of response from ℓ does not allow any other node k to return to M_k = NORMAL. Therefore the worst case is when all links fail one after the other in such a way that each increments $Q_i(\ell)$ and the total number can be no higher than $|E| - 1$ (for all links except (i,ℓ)) plus 1 for the case when (i,ℓ) just came up.

8. Conclusions

In this paper we have addressed the problem of providing formal description and validation to a number of Distributed Network Protocols. After introducing two simple basic protocols in Sec. 3 that form building blocks and unifying framework for the more complex ones, we introduce three classes of DNP's - connectivity test, minimum-hop paths and path-updating. For each we provide the algorithm for the nodes participating in the protocol and formal proof of its validation, extensively using the properties of the basic protocol on which it is based. Finally, we present a unified way to extend those protocols to the case of changes in the network topology.

Footnotes

1. The statement "For..." means "the actions taken by the processor when receiving ..."

2. The notation <·> will always denote the corresponding line in the Algorithm under consideration.

3. We use superscript s throughout the description of the present protocol to explicitly indicate the node that propagates the information.

4. We write the time in parentheses to indicate the value of a parameter at a specific time. Also, t- and t+ denote the time just before/after time t.

References

1. R.G. Gallager, A shortest path routing algorithm with automatic resynch, unpublished note, March 1976.

2. A. Segall, P.M. Merlin & R.G. Gallager, A recoverable protocol for loop-free distributed routing, Proc. of ICC, June 1978, Toronto.

3. S.G. Finn, Resynch procedures and a failsafe network protocol, IEEE Trans. on Comm., Vol. COM-27, Nr. 6, pp. 840-846, June 1979.

4. P.M. Merlin & A. Segall, A failsafe distributed routing protocol, IEEE Trans. on Comm., Vol. COM-27, Nr. 9, pp. 1280-1288, Sept. 1979.

5. P.M. Merlin, Specification and validation of protocols, IEEE Trans. on Comm., Vol. COM-27, Nr. 11, pp. 1671-1681, Nov. 1979.

6. A. Segall, Optimal distributed routing for virtual line-switched data networks, IEEE Trans. on Comm., Vol. COM-27, No. 1, pp. 201-209, Jan. 1979.

7. A. Segall, Advances in verifiable failsafe routing procedures, submitted to IEEE Trans. on Comm.