Distributed Network Protocols

ADRIAN SEGALL, SENIOR MEMBER, IEEE

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 to networks with changing topology.

I. INTRODUCTION

Consider the situation where 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 DNPs 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 microcomputers, it is certainly feasible that nodes will have their own processing and memory unit and will serve as communication processors or as switches. In principle, the common goal of 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 DNPs have been proposed or are currently under investigation are routing of information, network resynchronization, shortest path, minimum weight spanning tree, common channel coordination, information broadcast, and others.

The main purpose of this paper is to give a formal description and rigorous validation to a number of DNPs, 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 DNPs 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 DNPs 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 DNPs for networks with fixed topology, in Section VII we extend those protocols to incorporate cases of changing topology.

It is useful to emphasize at this point the importance of formal specification and validation of DNPs. The design of DNPs has two main stages. First, the construction of its general idea and then specification of the node algorithms intended to implement it. As with many computer programs, it is normally hard to make sure that the node algorithms indeed perform their intended purpose under all circumstances and the validation proofs attempt to provide this confidence, as well as deeper understanding of the algorithms. The more formal these proofs are, the more confidence one can have in the correctness of the algorithm. Although intuitive presentation and validation of DNPs is important, it is certainly not sufficient, and the present paper tries to emphasize this point by introducing formal methods for description and validation of DNPs. In spite of this, because of restrictions on the length of papers published in this Transactions, it was inevitable to delete or just outline several validation proofs of the presented protocols or the specification of the node algorithms. The complete details appear in [8].

II. THE GENERAL MODEL

In this section we give the general model and assumptions used in all presented DNPs. Consider a network \((V, E)\) where \(V\) is a set of nodes and \(E \subseteq V \times V\) 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 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) Associated with each link in each direction, say $(i, j)$, there is a link protocol whose purpose is to ensure that each message sent by node $i$ will arrive correctly within finite nonzero undetermined time, and all messages are received at node $j$ in the same order as 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 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 $1, 2, \ldots, |V|$ the nodes that are potentially in the network and, when needed, by $1, 2, \ldots, |V|$ the nodes actually belonging to the network.

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 such as “send messages to all neighbors,” meaning “send messages over all adjacent links.” The collection of all neighbors of node $i$ will be denoted by $\mathbf{N}_i$.

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.”

III. BASIC PROTOCOLS

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

A. 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 “flooding” the network, namely node $j$ transmits a message containing this information to all its neighbors and each other node $k$ in the network, when it receives the first such message, it sends 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.

**PROTOCOL PI:** Variables of the Algorithm at Node $i$.

$m_i$ shows whether 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$(i)$ message received from neighbor $l$.
- START message received from the outside world.

It is assumed that each message MSG 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$^*$ or MSG$(i)$
2. If $m_i = 0$, then $m_i \leftarrow 1$, send messages to all neighbors.

The properties of the protocol appear in the following Theorem whose proof is quite straightforward and appears in [8].

**Theorem PI-I.** Suppose a node $j$ receives START. Then we have the following properties of the protocol.

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). For a link $(i, l)$ let the weight $w_{jl}$ 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 queuing time). Then the collection of links $\{(p_i, l), \text{for all } i \text{ in the network}\}$ forms the tree of shortest distances from $j$ to all nodes.

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.

B. 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.

\footnote{The statement “For \ldots” means “the actions taken by the processor when receiving \ldots.”}

The notation $\langle \rangle$ will always denote the corresponding line in the algorithm under consideration.
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. When receiving any MSG$^i$, an arbitrary node $i$ marks the link from which it was received. When receiving the first MSG$^i$ from neighbor $l$ say, a node $i$ denotes this neighbor with a special mark $p_f^i$, and sends MSG$^i$ to all neighbors except to $p_f^i$. When it observes that it has received MSG from all neighbors, a node $i$ other than $s$ sends MSG$^i$ to $p_f^i$. It is shown below that receipt of MSG$^i$ 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: a) from node $s$ into the network for purposes of propagating information, and b) 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 $i \neq s$:**

- $m_i^i$ shows if node $i$ is currently participating in the protocol (values 0, 1).
- $N_i(l)(t)$ marks receipt of MSG$^s$ from neighbor $l$ (values 0, 1), $l \in \mathcal{S}_i$.
- $p_f^i$ is the neighbor from which MSG$^s$ was received first.

**Messages Sent and Received by the Algorithm at Node $i \neq s$:** MSG$^i$ and MSG$^s(i)$ with the same meaning as MSG in PI.

**Algorithm for Node $i \neq s$:** (Assumption: just before entering algorithm, node $i$ has $m_i^i = 0$, $p_f^i = \text{nil}$, $N_i(l)(0) = 0$ for all $l \in \mathcal{S}_i$).

1. For MSG$^s(l)$
2. $N_i(l)(1) \leftarrow 1$;
3. if $m_i^i = 0$, then $m_i^i \leftarrow 1$; $p_f^i \leftarrow 1$; send MSG$^s$ to all neighbors except $p_f^i$;
4. if $\forall l' \in \mathcal{S}_i$, holds $N_i(l') = 1$, then send MSG$^s$ to $p_f^i$; $m_i^i \leftarrow 0$; set $N_i(l') \leftarrow 0 \forall l' \in \mathcal{S}_i$.

**Algorithm for Node $s$:** For node $s$, the variables are $m_i^i$, $N_i(l)(t)$ for all $l' \in \mathcal{S}_s$, the messages are MSG$^s(i)$ and START and the algorithm is as follows.

1. For START
2. $m_i^s \leftarrow 1$; send MSG$^s$ to all neighbors.
3. For MSG$^s(l)$
   - $N_i(l)(1) \leftarrow 1$;
   - if $N_i(l')(1)$ holds, $\forall l' \in \mathcal{S}_s$, then $m_i^s \leftarrow 0$; set $N_i(l')(t) \leftarrow 0 \forall l' \in \mathcal{S}_s$.

*Note:* 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 analyze the protocol, we shall need the following notation:

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

$t(*)$ — time when event * happens.

**Theorem PIF-1:** 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

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

will form a directed spanning tree rooted at $s$; in addition, for all $i$

$$t(\langle 3 \rangle_i) > t(\langle 3 \rangle_s), \quad \text{where } j = p_f^i \quad (3.1)$$

(Note: some nodes may perform $\langle 4 \rangle_i$ before all nodes have performed $\langle 3 \rangle_s$);

b) node $s$ and all connected nodes $i$ will perform $\langle 4 \rangle_i$ in finite time and exactly once; moreover

$$t(\langle 3 \rangle_s) \leq t(\langle 4 \rangle_i) < t(\langle 4 \rangle_s), \quad \text{where } j = p_f^i \quad (3.2)$$

also, when node $s$ performs $\langle 4 \rangle$, all connected nodes will have completed the algorithm, i.e., performed $\langle 4 \rangle$;

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

*Proof:* 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., $\mathcal{S}_k$ such that $p_f^k = k$. Then all neighbors $n$ of $k$ will send MSG to $k$ whenever they perform $\langle 3 \rangle_n$. Node $k$ will receive all these messages and will be able to perform $\langle 4 \rangle_k$. At that time it will send MSG to $p_f^k$. 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 $\langle 4 \rangle$ and the procedure will continue downtree all the way to node $s$. This argument also proves (3.2) and completes the proof of the theorem.

**IV. Connectivity Test Protocols**

The purpose of this class of DNP's is to allow each node to learn what nodes are connected to it. The simplest protocol to accomplish this, referred to as CT1, is to use protocol PI repeatedly: 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 MSG$^j$ with the identity of some other node $j$, it marks...
j as connected and sends a message MSG/ with the identity of j to all neighbors. All further messages with the identity of j are discarded with no action taken.

The main properties of the protocol are given in the following theorem. Its formal specification and validation appear in [8].

**Theorem CT1:**

a) If node j is connected to i and START is delivered to any node connected to j (or to j itself), then the identity of i will be known at j in finite time and, conversely, if i and j belong to disconnected networks, then the identity of i will never be known at j.

b) With protocol CT1, there is no way for node j to know for sure what nodes are disconnected from it. 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.

**Communication Cost:** The number of bits transmitted on each link in each direction is \(|V| \log_2 |V|\). This is because every identity travels exactly once on each link in each direction, there are \(|V|\) identities and it takes \(\log_2 |V|\) bits to describe an identity. The total number of bits in the network is \(2|E||V| \log_2 |V|\), 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 b), namely how to 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/ with the identity of j from neighbor l, say, a node i denotes this neighbor (as in PIF) with a special mark \(p_l\), and sends MSG/ to all neighbors except \(p_l\). When it observes that it has received MSG/ (for any \(j \neq i\)) from all neighbors, node i sends MSG/ to \(p_l\). It is shown in Theorem CT2-1 that after having received MSG/ from all neighbors, node i can distinguish between the set of nodes to which it is connected and the nodes that are disconnected from it. Consequently, the termination property holds for Protocol CT2.

**Variables of the Algorithm at Node i:**

- \(M_i\)  
  WORK while i is participating in the protocol and NORMAL after completing the protocol;
- \(d_i\)  
  designates knowledge at i about connectivity to j (values \(0, 1\)) for all \(j\);
- \(N_i(I)\)  
  shows whether MSG/ has been received already from neighbor l (values \(0, 1\)) for all j and \(l \in \emptyset_i\);
- \(p_i\)  
  neighbor from which MSG/ has been received first, for all \(j \neq i\).

**Algorithm for Node i:** (Assumption: just before node i enters the algorithm, it has \(d_i = 0\), \(N_i(I) = 0\) for all j and all \(l \in \emptyset_i\).

1) For START or MSG'(I), \(j \neq i\),
   1a) if MSG, then \(N_i(I) = 1\);
   2) if \(d_i = 0\), then \(M_i \leftarrow\) WORK, \(d_i \leftarrow 1\), send MSG/ to all neighbors;
   3) if MSG and \(d_i = 0\), then \(d_i \leftarrow 1\); \(p' \leftarrow i\); send MSG/ to all neighbors except \(p_i\);
   4) if \(N_i(l') = 1\) holds \(\forall l' \in \emptyset_i\), then send MSG/ to \(p_i\);
     set \(N_i(l') = 0\) \(\forall l' \in \emptyset_i\).

5) For MSG'(I)
   5a) \(N_i(l) = 1\);
   6) if \(N_i(l') = 1\) holds \(\forall l' \in \emptyset_i\), then \(M_i \leftarrow\) NORMAL; set \(N_i(l') = 0\) \(\forall l' \in \emptyset_i\).

In order to analyze the protocol, we shall need the following notation (see also notations just before Theorem PIF-1): \(\langle . \rangle\) — the event of node i performing line \(\langle . \rangle\) of its algorithm regarding node j (i.e., reacting to receipt of MSG/).

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 you have the following.

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

b) Node j will perform \(\langle 6 \rangle\) in finite time and exactly once, and when this happens, it will have \(d_k = 1\) for all connected nodes k, and \(d_k = 0\) 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-b).

**Proof:** The event \(M_k \leftarrow\) WORK propagates as in PIF and hence will happen in finite time at all nodes k connected to the node that received START. For a given node i, after \(d_i\) becomes one, the event \(d_j \leftarrow 1\) (i.e., \(\langle 3 \rangle\) in the present protocol) propagates in the same way as \(\langle 3 \rangle\) in PIF and hence (cf. Theorem PIF-1) it will happen in finite time at node j, completing the proof of a). The proof of b) proceeds by induction on the nodes of the tree rooted at j referred to in Theorem PIF-1. The details appear in [8].

**Communication Cost:** Observe that by Theorem CT2-1, the communication requirements of CT2 are the same as those of CT1, namely \(|V| \log_2 |V|\) 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_/ of CT1 will now carry the identity of j as well as of all its neighbors, i.e., will have the format MSG_/ (X_), where XJ contains the identities of all neighbors of j. The termination property is achieved using the fact that, if a node k receives MSG_/ (X_), it will eventually receive MSG_/ (X_), as well, for any i \in X_, and the termination signal will occur when node k will have heard from all these nodes.

The algorithm at each node has two stages, where in the first one the node learns the identity of its neighbors and in the second it proceeds with the protocol as described before. The description and validation of the protocol appear in [8].

Communication Cost: On each link in each direction we need \( \log_2 |V| \) bits for the messages belonging to the first stage, and \( |V| (D + 1) \log_2 |V| \) bits for the second stage, where D is the average degree of the nodes (average number of neighbors). Clearly D = 2 \(|E|/|V|\) and hence the communication cost is \( 2(|E| + |V| + 1) \log_2 |V| \) 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 at Node i.

- \( M_i \): same meaning as in CT2;
- \( d_i^1 \): 0 before entering algorithm, 1 while looking for identity of neighbors, 2 while looking for all connected nodes;
- \( d_i^2 \): 0 when i knows nothing about j (for j \neq i), 1 when i knows j only as a neighbor of another node, 2 while i knows j directly;
- \( N_i(l) \): shows if WAKE has been received from neighbor l (values 0, 1).

Messages Sent and Received by the Algorithm at Node i:

START:

\[ D_i = \{d_i^1, d_i^2, \ldots, d_i^{|F_i|}\} \]

message sent;

D(i) message received from neighbor l; we denote its contents by \( \{d_i^1, d_i^2, \ldots, d_i^{|F_i|}\} \).

Algorithm for Node i: (Assumption: just before node i enters algorithm, it has all \( d_i^1 = 0 \) and \( N_i(l) = 0 \) for all j and all \( l \in \hat{\mathbb{S}}_j \))

1. For START
   1.1 \( d_i^1 \leftarrow 1; \) \( M_i \leftarrow \text{WORK}; \) send \( D_i \) to all neighbors.
   1.2 For D(i)

2a) If \( D(i) = \{0, 0, \ldots, 0, 1, 0, \ldots, 0\} \), then

2b) \( N_i(l) \leftarrow 1; \)

2c) If \( d_i^1 = 0 \), then same as 1a);

2d) \( d_i^2 \leftarrow 1; \)

3) If \( N_i(l) = 1 \) holds \( \forall l \in \hat{\mathbb{S}}_j \), then

3a) \( d_i^1 \leftarrow 2; \) set \( N_i(l) = 0 \) \( \forall l \in \hat{\mathbb{S}}_j \); send \( D_i \) to all neighbors;

4) If \( D(i) \neq \{0, 0, \ldots, 0, 1, 0, \ldots, 0\} \) and \( M_i \neq \text{WORK} \), then

4a) If \( d_i^1 \neq 2 \), then set \( d_i^k = \max\{d_i^k, d_i^k\} \forall k \)

5) Else, if \( \exists j \) such that \( d_i^k = 2 > d_i^k \), then

5a) \( d_i^k \leftarrow \max(d_i^k, d_i^k) \forall k \); send \( D_i \) to all neighbors;

6) If \( d_i^1 = 2 \) or 0 holds \( \forall j \), then \( M_i \leftarrow \text{NORMAL} \).

Observe that messages of type \( \{0, 0, \ldots, 0, 1, 0, \ldots, 0\} \) replace the messages of the first stage in 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 below.

Theorem CT4: Suppose START is delivered to some node. Then we have the following.

a) Exactly one message \( \{0, 0, \ldots, 0, 1, 0, \ldots, 0\} \) 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 \( d_i^1 \geq 1 \) for all neighbors l of i.

c) No more than \( |V| \) messages with format \( \neq \{0, 0, \ldots, 1, 0, \ldots, 0\} \) traverse each link in each direction.

Proof: Lines 1.1, 1.2, and the fact that the propagation is as in PI imply a) and b). From the algorithm it is clear that \( d_i^1 \) can only increase and from 3a) and 5a) follows that a message \( \neq \{0, 0, \ldots, 1, 0, \ldots, 0\} \) can be sent by i only when some \( d_i^1 \) is set from 0 to 1 or 2, and this can happen once for each j. Hence c).

For a given node i, the increase of \( d_i^1 \) to 2 is triggered by \( d_i^1 \leftarrow 2 \) and propagates as PI, except that it is stopped at nodes i for which \( d_i^1 \neq 2 \) (cf. 5a) \( \forall a \)) and the propagation continues as soon as \( d_i^1 \leftarrow 2 \) (cf. 3a). This proves the first part of d). To prove the second part, observe from 2d) \( \forall a \), 4a) \( \forall a \) that after a message arrives at node i say, \( d_i^k \geq d_i^k \) holds for any k (where \( d_i^k \) is an entry in the vector \( D_i \) and \( d_i^k \) the corresponding entry in the arriving message). Also b) shows that whenever \( d_i^1 \leftarrow 2 \), we have \( d_i^1 \geq 1 \) for all neighbors k of i. Consequently \( d_i^1 = 2 \) implies \( d_i^k \geq 1 \) for all neighbors k of i. This completes the proof of d). Since \( d_i^1 \) must remain 0 for all disconnected nodes, 6) will occur.
in finite time with \( d_i^j = 2 \) for all connected nodes, and \( d_i^j = 0 \) for all disconnected ones. It remains to show that \( \langle 6 \rangle \) cannot occur at node \( i \) before \( d_i^j \) becomes 2 for all connected nodes \( j \). Suppose the contrary. Then there must exist a set of nodes \( \mathcal{S} \) containing \( i \), where \( \mathcal{S} \) is not the entire network, and \( d_i^j = 2 \) for \( n \in \mathcal{S} \) and \( d_i^j = 0 \) for \( n \notin \mathcal{S} \). From d) it then follows that \( d_i^k \geq 1 \) for all \( k \) that are neighbors of any node in \( \mathcal{S} \), contradicting \( d_i^k = 0 \) for all \( k \notin \mathcal{S} \). This completes the proof of e) and of the theorem.

**Communication Cost:** Each message contains 2 \( |V| \) bits and hence at most \( 2 |V| (|V| + 1) \) bits will travel on each link in each direction.

Protocol C13 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(\( \mathcal{S}_i \)), a node \( i \) consults its table containing \( \{d_i^k \} \). If \( d_i^k = 2 \), 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 \( d_i^k < 2 \), then \( d_i^k \) and the MSG is sent to all neighbors, but now, before sending MSG(\( \mathcal{S}_i \)), the following pruning operation is performed.

For all \( k \in \mathcal{S}_i \), if \( d_i^k > 1 \), then \( k \) is deleted from \( \mathcal{S}_i \); otherwise \( k \) is not deleted from \( \mathcal{S}_i \) and the variable \( d_i^k \) receives value 1. Then MSG(\( \mathcal{S}_i \)) is sent to all neighbors. Node \( k \) can indeed be 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, 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. Hence the communication cost is bounded by \( 2 |V| \log_2 |V| \) bits per link in each direction.

**V. 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 control messages 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 MH:** 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 it 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 \( |V| \) for all \( k \) (values \( 0, 1, \ldots, |V| \));
- \( p_i^k \): 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 \(-1, 0, 1, \ldots, |V| - 1\));
- \( M_i \): shows if node \( i \) is currently participating in the protocol (values NORMAL, WORK);
- \( N_i(l) \): level of last message received on link \((i, l)\) (values \(-1, 0, \ldots, |V| - 1\)), for \( l \in \mathcal{S}_i \);

**Messages Sent and Received at Node i:**

- MSG(LIST, \( \_ \)): message sent by node \( i \),
- MSG(\( l, LIST \) \( = \) MSG(LIST) received on link \((i, l)\),
- START.

**Algorithm for Node i:** (Assumption: just before node \( i \) enters algorithm, it has \( p_i^k = \text{nil}, d_i^k = |V| \) for all \( k \), \( Z_i = N_i(m) = -1 \) for all \( m \in \mathcal{S}_i \)).

\begin{itemize}
  \item[(1)] For START or MSG(\( l, LIST \)),
  \item[(2)] if \( Z_i = -1 \), then \( d_i^l \leftarrow 0, M_i \leftarrow \text{WORK}, Z_i \leftarrow 0 \), \( \text{LIST} = \{ i \} \), send MSG(LIST) to all neighbors;
  \item[(3)] if MSG and \( M_i = \text{WORK} \), then
  \item[(4)] \( N_i(l) \leftarrow N_i(l) + 1 \);
  \item[(5)] \( \forall k \in \text{LIST} \), then
  \item[(5a)] if \( d_i^k > N_i(l) + 1 \), then \( d_i^k \leftarrow N_i(l) + 1 \), \( p_i^k \leftarrow l \);
  \item[(6)] if \( Z_i \leq N_i(m) \) holds \( \forall m \in \mathcal{S}_i \), then
  \item[(6a)] \( Z_i \leftarrow Z_i + 1 \), \( \text{LIST} = \{ k \mid d_i^k = Z_i \} \), send MSG(LIST) to all neighbors;
  \item[(7)] if \( \text{LIST} = \phi \), then \( M_i \leftarrow \text{NORMAL} \).
\end{itemize}

**Note:** Observe that the variable \( p_i^k \) is not needed by the algorithm, and only designates the neighbor corresponding to the MH path to \( k \).

The following definition and theorem summarize the main properties of the protocol. The proof is given in the Appendix.

**Definition:** The number of links on the minimum-hop path from \( i \) to \( k \) is called the hop-distance from \( i \) to \( k \).
Theorem MH-1: Suppose START is delivered to a node (or several nodes). Then every connected node i a) will enter the protocol (i.e., perform (2)) in finite time; b) will complete the protocol (i.e., perform (7)) in finite time, with $d_i$ corresponding to the minimum-hop path from $i$ to $k$ for all connected nodes $k$ and with $d_i = |V|$, $p_i = \text{nil}$ for all disconnected nodes $k$.

Communication Cost: From the proof of Theorem MH-1 in the Appendix follows that the identity of every node travels exactly once on each link, and hence we need $|V| \log_2 |V|$ bits on each link in each direction.

VI. 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-tree 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 "destination" node $s$. The protocol is very similar to the PIF protocol, except for two features: first, a tree is initially available and the protocol moves first up and then down on that tree, and second, when moving down tree, the protocol updates the initial tree, so that the resulting paths provide an improvement over the old ones.

Protocol PU: Variables of the Algorithms at Node $i \neq s$:

- $N_i^s(l)$, same as in protocol PIF;
- $d_i^l$—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), $l \in \delta_i$;
- $d_i^l$—estimated distance from $i$ to $s$ on the preferred path; $p_i^l$—"preferred" neighbor of $i$ for $s$;
- $D_i^l$—storage for $d_i^l + d_i^{p_i}$, for $l \in \delta_i$;
- $m_i^l - 1$ after performing (3) and before performing (4); 0 otherwise.

Assumption: just before START is delivered to $s$, all connected nodes $i$ have

a) $p_i^l$, $d_i^l$ with the property that the collection of links $(i, p_i^l)$ form a directed tree rooted at $s$ and also $d_i^l > d_i^k$ where $k = p_i^l$, i.e., $d_i$ is strictly decreasing while moving down tree;

b) $m_i^l = 0$, $N_i^s(l) = 0$ for $l \in \delta_i$.

Messages Received and Sent by the Algorithm at $i$:

- $\text{MSG}^i(d_i^l)$—message sent;
- $\text{MSG}^i(l, d_i^s)$—message received.

Algorithm for Node $i \neq s$:

1. For $\text{MSG}^i(l, d_i^s)$
2. If $l = p_i^l$, then $d_i^l = \min D_i^l$ over $l'$ such that $N_i^s(l') = 1$, $m_i^l - 1$, send $\text{MSG}^i(d_i^l)$ to all neighbors, except $p_i^l$;
3. if $N_i^s(l') = 1$ holds $\forall l' \in \delta_i$, then send $\text{MSG}^i(d_i^s)$ to $p_i^l$, $p_i^l - k^*$ that achieves $\min D_i^l(l')$ over $l' \in \delta_i$, $m_i^l - 0$, set $N_i^s(l') = 0 \forall l' \in \delta_i$.

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

Theorem PU-I: Suppose the Assumptions a) and b) given in the protocol hold. Then

a) theorem PIF-1 holds, where $p_i^l$ refers (only in this part) to the initial preferred neighbors;

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

i) $m_i^l \leq m_i^{p_i}$;

ii) if $m_i^l = m_i^{p_i} = 0$, then $d_i^l > d_i^{p_i}$;

c) for each link $(i, l)$ the "distance" $d_i^l$ 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 $\{p_i^l\}$ is not identical to the shortest path tree in terms of the measured $\{d_i^l\}$, 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 $\text{MSG}$ is received from $p_i^l$ (and not as soon as the first $\text{MSG}$ is received, as in PIF), the new quantities $d_i^l$, $D_i^l(l)$, $d_i^{p_i}$ are introduced and the preferred neighbor $p_i^l$ is changed in (4).

Now, if we denote by $P_i^l$ the initial tree, (3) and (4) propagate here exactly as in PIF, provided that in that protocol a $\text{MSG}$ traverses any link in $P_i^l$ 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 $t$—throughout the network, 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 $p_i^l$ 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 $m_i^l(t-1) = 0$, we have by the induction hypothesis $m_i^l(t) = 0$ for any $j$ for which $p_i^l(t) = i$ and hence b) i) continues to hold for $j$ and $i$ after time $t$. On the other hand, when performing (3), node $i$ receives $\text{MSG}^l$ from $p_i^l$, so that $p_i^l$ must have performed (3) before $t$ and has not performed (4) yet, implying that $m_i^{p_i}(t) = 1$ and, since $m_i^{p_i}(t+) = 1$, assertion b) ii) continues to hold after $t$ for $i$ and $p_i^l$ as well.

Now suppose (4) happens at some node $i$ at time $t$. Observe that at that time, $i$ has already received $\text{MSG}$ from all neighbors and it performs $m_i^l - 0$. Consider first any node $j$ such that $p_i^l(t) = j$. If $p_i^l(t_0) = i$, where $t_0$ is the time the protocol started, then receipt of $\text{MSG}$ at $i$ from $j$ means

4We write the time in parentheses to indicate the value of a parameter as a specific time. Also, $t -$ and $t +$ denote the time just before and after time $t$, respectively.
that \( j \) has performed \( \langle 4 \rangle \) before time \( t \). If \( p_j(t_o) \neq i \), then \( j \) has changed \( p_j \) before time \( t \) and again this shows that it has performed \( \langle 4 \rangle \) before time \( t \). Consequently, in either case holds \( m_j^{(i)}(t) = 0 \) and hence b) i) continues to hold after time \( t \) for \( i \) and \( j \). Also, from the way \( d_j \) is calculated and \( p_j \) is chosen follows that

\[
d_{ji}^j \geq D_{ji}^j(i) = d_i^j + d_{ji} > d_i^j,
\]

where the last inequality follows from the assumption \( d_{ji} > 0 \) (see definition of \( d_{ji} \)). Consequently b) ii) continues to hold after time \( t \). Now, consider the pair \( i \) and \( k^* = p_i(t + \). The inequality in assertion b) ii) holds trivially after \( t \) for \( i \) and \( k^* \) since \( m_i^{k^*} = 0 \), 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 \( m_l^{(i)} = 0 \), and going around the loop this would imply by b) ii) that \( d_{ji}^j > d_{ji}^i \). 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].

**Communication Cost:** Clearly there is exactly one message on each link in each direction. Its size in bits depends on the number of bits assigned to \( d_i^j \).

**Protocol PU (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 \( \langle (i, p_i^j) \rangle \) for all “destinations” \( j \) in the network. In order to save space, we shall not present the network here explicitly, but only mention that this can be done by using protocol CT2 with additions of variables like \( d_i^j \) and \( D/(i) \) with the same meaning as in protocol PU. For details, see [8].

**VII. Topological Changes**

The protocols presented so far assume fixed topology of the network. As such, the CT and MH 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 Section II should be changed accordingly. In particular, assumptions a) and c) of Section II will be changed now to

a') link \( (i, j) \) fails/recover at the same time that link \( (j, i) \) fails/recover, so that \( (i, j) \) belongs to the network if and only if \( (j, i) \) belongs to the network;

b') 1) each message sent by node \( i \) on link \( (i, j) \) arrives correctly in finite nonzero undetermined time or the link fails in finite time;

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

3) failure or recovery of a node is considered as failure/recovery of all adjacent links.

To make 2) above precise, let \( F_i(l) \) 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 a) if \( F_i(l) = F(l) = UP \) and \( F_i(l) = DOWN, \) then \( F(l) \) becomes DOWN in finite time and before \( F_i(l) \leftarrow UP; \) b) if \( F_i(l) = F(l) = DOWN \) and \( F(l) = UP \), then in finite time holds either \( F(l) \leftarrow UP \) or \( F_i(l) = F(l) = DOWN. \)

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 labeled with increasing numbers \( R \), every node \( i \) remembers the highest cycle number known to it so far, denoted by \( R_i \), 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 \( R_i \), and acts as if it has received START for a new cycle with this number. Here “resetting 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, \( p_i^j \leftarrow nil, d_i^j \leftarrow |V| \) for all \( k \) and \( Z_j \leftarrow -1, N_i(m) \leftarrow -1 \) 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 \( i \) receiving a message with higher cycle number \( R \) than the highest known to it, \( R_i \), resets its own variables, increases its registered maximal cycle number \( R_i \) accordingly, and acts as if it enters the protocol now (i.e., the corresponding cycle of the extended protocol). In this way the cycle with higher number will “cover” the lower number cycles, 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 because of multiple topological changes, 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, as opposed to protocols where only the information affected by the topological change is discarded, while the rest of the network adapts smoothly to the new situations. For the PU protocol, such a protocol appears in [2, 4, 7], but for others this is still an open question.

The protocol described so far will be referred to as version \( \Lambda \) of the extended protocol. As an example, we shall consider here the Extended MH protocol. To save space we do not indicate here the explicit algorithm at each node, but we may indicate that it is identical to the
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 in the final connected network will have $M_i = \text{NORMAL}$ with the same cycle number $R_i$, with correct $d^k_i$ and $p^k_i$ for all connected nodes $k$ and with $d^k_i = |V|, p^k_i = \text{nil}$ for all disconnected nodes $k$.

Proof: 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, and every change in a node status affects a finite number of nodes. Let $\{t_n\}$ 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 $\{t_n\}, \{t_n\}$ are finite. Let $R = \max\{R_i(t_n)\}$ 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 $i \in \{t_n\}$ 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 Theorem MH-1 holds here, completing the proof.

In version A of EMH as presented above, as well as all other similar extended protocols, there is the problem that the cycle counter numbers $\{R_i\}$ increase 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 ensure bounded counter numbers and here we 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 will introduce the important features of the procedure and then explain the correctness between versions B and C.

Protocol EMH (extended MH) Version B: Variables Used by the Algorithm at Node $i$. Same as in MH with the addition

- $R_i$: highest cycle number known to $i$ (values: $0, 1, \ldots$),
- $t_i(l)$: status of link $(i, l)$ as known by $i$ (DOWN, UP).

Messages Sent And Received at Node $i$:
- $\text{MSG}(R_i, \text{LIST})$: sent,
- $\text{MSG}(l, R, \text{LIST}) = \text{MSG}(R, \text{LIST})$ received on link $(i, l)$.

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

1. $\text{Node } i \text{ becomes operational (Note: Node } i \text{ becoming operational forces all operating links } (i, l) \text{ with } l \text{ operating, to become operational for } l')$
   1a. $F_i(l') \leftarrow \text{UP}$ for all operating adjacent links $(i, l')$ with $l$ operating;
   1b. $F_i(l') \leftarrow \text{DOWN}$ for all nonoperating adjacent links $(i, l')$ or those links $(i, l')$ with $l'$ nonoperating;
   1c. reset variables: $R_i \leftarrow 0; M_i = \text{NORMAL};$
   1d. for all $l'$ for which $F_i(l') = \text{UP}$, proceed as in $\langle 2a \rangle$.

2. Adjacent link $(i, l)$ fails;
   2a. $R_i \leftarrow R_i - 1$ and proceed to $\langle 2b \rangle$.

3. Adjacent link $(i, l)$ becomes operational;
   3a. wait until $M_i = M_i \leftarrow \text{NORMAL}$ and then
   3a". if $R_i < R_i$, then $R_i \leftarrow R_i + (R_i - R_i)$ 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 $(R_i - R_i)$;
   3a"'. if $R_i < R_i$, proceed as $\langle 2a \rangle$ for nodes connected to $i$ with increments $(R_i - R_i)$;
   3b. $R_i \leftarrow R_i + 1$ and proceed to $\langle 2b \rangle$.

4. For $\text{MSG}(l, R, \text{LIST})$;
   4a. if $R \geq R_i$, then
   4b. if $R > R_i$, then $R_i \leftarrow R_i$; same as $\langle 2c \rangle$;
   4c. if $M_i = \text{WORK}$, then
   4d. same as $\langle 4 \rangle-\langle 7 \rangle$ in MH, except that $\text{MSG}$ has format $\text{MSG}(R_i, \text{LIST})$.

Note that $\langle 2 \rangle$ and $\langle 3b \rangle$ here correspond to $\langle 2 \rangle$ in MH, while $\langle 3c \rangle$ corresponds to $\langle 3 \rangle$ in MH. Clearly, similar extended protocols can be given for the CT and also for the PUI protocols.

The main property of version B is given in Theorem EMH-B-1, but first we need several definitions.

Definitions: A link $(i, l)$ is said to be operating if $F_i(l) = F_i(l) = \text{UP}$. 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 $M_i = \text{WORK}$.
b) There is at least one node \( i \in S \) with \( M_i = \text{NORMAL} \) and 
   i) \( R_j = R_i \) holds \( \forall j \in S \) with \( M_j = \text{NORMAL} \) and 
   ii) \( R_j \geq R_i \) holds \( \forall j \in S \) with \( M_j = \text{WORK} \).

**Theorem EMH-B-1:** In EMH-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 \( \tau \) and \( t - \) had \( R_j \leq \text{level of} \ S' \) at time \( t \).

The events that can happen at time \( t \) and affect the properties of the theorem are a) a node becomes operational, b) a link fails, c) a link is brought up, and d) \( M_i \) 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 \( \langle 1d \rangle \) holds, so that this case reduces to c). 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 d). Observe next that c) can happen only if \( M_i(t -) = M_j(t -) = \text{NORMAL} \), 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 \( R_j(t -) = R_i(t -) \). Hence \( 2a'' \) and \( 2a''' \) do not apply, and therefore the only relevant variables that are changed are \( M_i, R_i, M_j, R_j \) (lines \( 2a'' \) and on), and this again reduces to d).

Suppose now that the link \((i, l)\) does connect two previously disconnected sets. If \( R_j(t -) = R_i(t -) \), the same argument as before applies. If for example, \( R_j(t -) < R_i(t -) \), let \( t' \) be the time just after execution of \( 2a'' \), but before execution of \( 2a''' \). Recall that \( M_j(t -) = M_i(t -) = \text{NORMAL} \) and since each of the sets are synchronized at \( t - \), we have \( R_j(t -) \geq R_i(t -) \) for all \( r \) connected to \( i \) and \( R_i(t -) \geq R_j(t -) \) for all \( r \) connected to \( l \), with equality in both cases for those nodes \( r \) that have \( M_i(t -) = \text{NORMAL} \). Now, from \( t - \) to \( t' \), all nodes \( r \) connected to \( i \) raise their cycle number \( R_r \), by \( R_j(t -) \rightarrow R_i(t -) \), and so do all messages in transit, and hence, the new combined set remains synchronized at \( t' \). Now the transition from \( t' \) to \( t + \) is the execution of \( 2a'' \) and on, and again this reduces to case d), 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 \( M_i \sim \text{NORMAL} \), in which case \( R_i \) is not changed. Let \( R \) be the value of \( R_i \) at time \( t \) and at \( t + \). We must show that for all nodes \( j \in S \), where \( S \) is the set of nodes connected to \( i \) at time \( t \), we have \( R_j(t) \geq R \), with equality if \( M_j(t) = \text{NORMAL} \). But since \( S \) is synchronized at \( t - \) by the induction hypothesis, the condition \( M_j(t) = \text{NORMAL} \) requires \( R_j(t) \leq R \) for all \( j \in S \), and therefore it is sufficient to show that \( R_j(t) \geq R \) for all \( j \in S \).

At time \( t \), node \( i \) performs \( M_i \sim \text{WORK} \) and let \( P \) be the set of nodes \( k \) for which \( d_k(t) \leq |V| \). Nodes \( k \in P \) certainly have \( R_k(t) \geq R \). Now take any node \( \alpha \) such that \( \alpha \in S \), but \( \alpha \notin P \). We want to show that \( R_\alpha(t) \geq R \). Observe that there must exist a node \( \beta \in S \), \( \beta \notin P \) such that \( \beta \) is at time \( t \) a neighbor of a node \( \gamma \in S \cap P \) (see Fig. 1). Since \( \beta \in P \), node \( \beta \) was disconnected from \( \gamma \) at some time after \( R_\gamma \rightarrow R \). Let \( S_\gamma \) be the connected set containing \( \gamma \) at time \( t - \), where \( \tau < t \) is the time when link \((\beta, \gamma)\) was brought up. At that time \( M_\gamma \) was \text{NORMAL} and \( R_\gamma \geq R \) and by the induction hypothesis, \( S_\gamma \) was at level \( \geq R \). In addition, by the second assertion of the Theorem that holds at time \( t - \) because of the induction hypothesis, the fact that \( \alpha \) is connected at time \( t \) requires \( R_\alpha(t) \geq R \). This completes the proof of case d) 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 \( S \) will have at that time counter number \( \geq R \).

At time \( t \), every node \( i \in S \) has \( R_i \geq R \) and never decreases. Let \( t' \) be the first time after \( t \) when a node \( j' \) becomes connected to any node \( i \in S \). Since until that time all connected sets are synchronized, it must hold that \( R_j(t') \geq R \) by the same argument as in c) 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.

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

**Lemma EMH-B-1:** In EMH-version B, the following properties hold

a) If \( 2a'' \) holds at time \( t \) and nodes \( i \) and \( l \) are connected at time \( t - \), then \( R_j(t -) = R_l(t -) \) and hence \( 2a'' \rightarrow 2a''' \) are not performed.

b) Theorem EMH-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 \( R' \) and \( R'' \) 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\((i, R, LIST)\) and \( R > R_\ast \), then
LIST = \{\} and this message was sent by \( l \) while increasing
\( R_\ast \), i.e., either in \( (2c) \) or \( (3b) \), but not in
\( (6b) \).

e) The values of \( R \) and \( R_\ast \) are not necessary for the
algorithm; we only need to know if \( R < R_\ast \), \( R = R_\ast \),
or \( R > R_\ast \).

Proof: a) follows from Theorem EMH-B-1. b) can be
proved exactly as Theorem EMH-A-1. For c), the fact that
\( R_\ast \) 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 \( t - 1 \). The counter
\( R_\ast \) 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 \( R_\ast \) by 1, in
\( (3b) \) while incrementing \( R_\ast \) to \( R \) which is exactly \( R + 1 \)
by the induction hypothesis, or in \( (6b) \) while maintaining
\( R_\ast \) to the previous level. This proves both c) and d).
Finally, e) is clear from the algorithm.

Protocol EMH-version C: Variables Used by the Algo-
rithm at Node \( i \). Same as in MH and in addition \( F_i (1) \) as in
version A and \( Q_i (l) = R_i - X R_i (l) \) where \( X R_i (l) \) is the
largest counter number received from neighbor \( l \) in version
\( B \).

Messages Sent and Received at Node \( i \):

MSG\((\Delta R, LIST)\) - sent, where \( \Delta R \) has the meaning of
the difference between \( R_\ast \) in version \( B \) and last \( R \) sent
on this link;

MSG\((\Delta R, LIST)\) - received.

Algorithm for Node \( i \): Definition: ‘‘reset variables’’ has
the same meaning as in version A.

\begin{itemize}
  \item \( (1) \) Node \( i \) becomes operational (same Note as in
version A);
  \item \( (1a-1b) \) same as in versions A and B;
  \item \( (c) \) reset variables; \( Q_i (l') = 0 \) for all \( l' \) for
which \( F_i (l') = UP \); \( M_i \leftarrow NORMAL \);
  \item \( (d) \) if there is an operational link \( (i, l') \) for
which \( M_{l'} = NORMAL \), proceed as in
\( (2a') \); else wait until this happens and
then proceed as in \( (2a') \);
  \item \( (2a) \) \( Q_i (l') = Q_i (l') + 1 \) \( \forall l' \neq l \) for which
\( F_i (l') = UP \); proceed to \( (2b) \);
  \item \( (2') \) Adjacent link \( (i, l) \) is operational and \( F_i (l)
= F_i (l) = DOWN \) and \( M_i = M_{l} =\)
NORMAL;
  \item \( (2a') \) \( Q_i (l) = 1 \); \( Q_i (l') = Q_i (l') + 1 \) \( \forall l' \) for
which \( F_i (l') = UP \);
  \item \( (2b) \) \( F_i (l) \leftarrow DOWN \) or \( UP \) according to new
status;
  \item \( (2c) \) reset variables; \( M_i \leftarrow WORK \); \( d_i \leftarrow 0 \); \( Z_i
\leftarrow 0 \); \( LIST_i = (i) \); sent MSG\((i, LIST)\)
to all \( m \) for which \( F(m) = UP \).
  \item \( (3) \) For MSG\((i, \Delta R, LIST)\)
  \item \( (3a) \) if \( \Delta R < Q_i (l) \), then: \( Q_i (l) = Q_i (l) -
\Delta R \);
  \item \( (3b) \) else if \( \Delta R > Q_i (l) \) (note: i.e., \( \Delta R =
1, Q_i (l) = 0 \), then
  \item \( (3b') \) \( Q_i (l') = Q_i (l') + 1 \) \( \forall l' \) for which
\( F_i (l') = UP \);
  \item \( (3c) \) same as \( (2c) \);
  \item \( (3d) \) \( Q_i (l) = 0 \);
  \item \( (3e) \) if \( M_i = WORK \), then
  \item \( (4)-(7) \) same as in \( (4)-(7) \) in MH, except that
MSG has format MSG\((0, LIST)\).
\end{itemize}

Note: Observe that knowledge by \( i \) of the variables \( F_i (1) \)
and \( M_i \) requires a local protocol on every link \( (i, l) \).

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 EMH-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 \( t - 1 \),
where \( t \) is the time of the event occurrence, then in version
\( B \) we have \( R_i (t - 1) = R_i (t) \) from Lemma EMH-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 \( t - 1 \),
the effect of bringing \( R_i (t - 1) \) and \( R_i (t - 1) \) 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 MHE-C-1:

a) The counter numbers \( R \) in MSG take values 0 and 1
only.

b) For every \( i \) and \( l \), the variable \( Q_i (l) \leq |E| \), 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
\( Q_i (l) \) can increase only if, while link \( (i, l) \) is operating,
node \( i \) keeps sending MSG\((i, LIST)\) to \( l \), but \( l \) does not
respond. After the cycle corresponding to the first of these
messages covers the entire network (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 \( Q_i (l) \) 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.

VIII. CONCLUSION

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 Section III that form building blocks and a unifying framework for the more complex ones, we introduced three classes of DNP's—connectivity test, minimum-hop paths, and path-updating. For each we provided 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 presented a unified way to extend those protocols to the case of changes in the network topology.

Appendix

The proof of Theorem MH-1 is given in the following two lemmas. The first one indicates several preliminary properties of Protocol MH connected to message exchanges and variable updates, while in the second we use Lemma MH-1 to validate the basic properties of the protocol.

Lemma MH-1: Suppose START is delivered to a node (or several nodes). Then for any connected node i the following properties hold

1. i will enter the protocol in finite time;
2. messages are sent by node i if and only if Z, is incremented at the same time; if MSG is sent by i while Z, = Z, receipt of the MSG at neighbor l will cause N,(i) \rightarrow Z;
3. Z, and N,(m) for each m \in \delta_i change only by increments of +1;
4. for each m \in \delta_i, holds N,(m) = Z, or Z, + 1 and there is at least one m for which N,(m) = Z, - 1 (note: this implies Z, = \min_m{N,(m) + 1};
5. no message can arrive on links (i, m) for which N,(m) = Z, + 1;
6. if Z, is incremented at time t, then for all m \in \delta_i holds N,(m)(t + 1) = Z,(t + 1) or Z,(t + 1) - 1.

Proof: a) holds since propagation of (2) happens as in PI. Assertion b) holds since Z, is incremented whenever MSG is sent (cf. (4)) and both are initialized \(z_1-1\). 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, = 0 and \(Z_{\min} = -1\), or Z, = 1 and \(Z_{\min} = 0\), 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,(\cdot) or Z, can change at time t only if a MSG is received, from neighbor l say. Let Z,(t - ) = Z. We have several cases:

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

ii) \(N,(t - ) = Z, + 1\) and \(\exists m \neq l\) with \(N,(m)(t - ) = Z, - 1\); then \(N,(t +) = Z, + 1\) and \(Z,(t +) = Z, + 1\), since (6) holds at t and \(d, = 1\) and continued to hold at time t + ;

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

iv) we claim that \(N,(t - )\) cannot be \(Z, + 1\). Suppose \(N,(t - ) = Z, + 1\). Then \(N,(t +) = Z, + 2\), and from b) it follows that at time \(t < t\), node l has sent MSG(LIST) while \(Z, = Z, + 2\). From (6), (6a) we have \(Z,(t - ) = Z, + 1\) and \(N,(t +) = Z, + 1\). This means that \(\exists m 2 \leq t\) when i has sent MSG(LIST) to l, while

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, is incremented only in case ii) and f) clearly holds in this case, completing the proof of the Lemma.

Lemma MH-2: Recall the definition of the term hop-distance just before Theorem MH-1. Under the same conditions as in Lemma MH-1, the following properties hold.

a) If a node i has nodes at hop-distance r, then it sets \(Z, \rightarrow r\) in finite time and then sends MSG(LIST,). The node i sends (i) propagates as in PI and hence will happen at all nodes in finite time. Now 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 it and all its neighbors m have nodes at hop-distance (r - 1) and, by the induction hypothesis, they set \(Z, \rightarrow (r - 1)\) and send MSG(LIST). When such a message arrives at i, it sets \(N,(m) \rightarrow (r - 1)\) and after all such messages arrive, (6) will hold with \(Z, = (r - 1)\). This causes \(Z, \rightarrow r\). At this time we have from Lemma MH-1, \(N,(m) = r\) 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) while \(Z, \rightarrow (r - 1)\) and hence was received at i while \(N,(m) \rightarrow (r - 1)\), but was sent by no neighbor m' while \(Z, \rightarrow (r - 1)\). Hence at the time \(Z, \rightarrow r\) we have \(d, = r\) and therefore k is sent in MSG(LIST). From (5a) it is clear that this \(d,\) and the corresponding \(p,\) are final and correct. A similar argument shows that nodes at hop-distance other than r cannot be included in the LIST, considered above.

b) First consider a node i such that \(S, = \min\{S,\}\) where the min is over all nodes in the network. All its neighbors m have nodes at distance \(S,\) and by a) they send MSG(LIST,). While \(Z, \rightarrow S,\) all these messages arrive to i, Z, will become \(S, + 1\), but since i has no nodes at hop-distance \(S, + 1\), holds LIST, = \(\emptyset\) hence i performs (7). Now suppose by induction that b) holds for all nodes i for which \(S, \leq S - 1\). Consider a node j with \(S, = S\). Node j has a node k at hop-distance S and k is included in LIST, when j sends MSG(LIST) while \(Z, \rightarrow S\). For an arbitrary neighbor m of j, node k is at hop-distance \((S - 1)\), S or \((S + 1)\) from m and hence \(S, \geq S - 1\). If \(S, \geq S\), then a) implies that \(Z,\) will become \(S\) in finite time. If \(S, = S - 1\), then \(Z,\) will become \(S\) in finite time from the induction hypothesis. Hence from Lemma MH-1 b), \(N,(m)\) will become \(S\) in finite time.
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 of the lemma.

Lemma MH-1 a) and Lemma MH-2 a), b) are exactly Theorem MH-1 and this completes the proof of the theorem.

REFERENCES


On Secret Sharing Systems

EHUD D. KARNIN, STUDENT MEMBER, IEEE, JONATHAN W. GREENE, STUDENT MEMBER, IEEE, AND MARTIN E. HELLMAN, FELLOW, IEEE

Abstract—A "secret sharing system" permits a secret to be shared among $n$ trustees in such a way that any $k$ of them can recover the secret, but any $k - 1$ have complete uncertainty about it. A linear coding scheme for secret sharing is exhibited which subsumes the polynomial interpolation method proposed by Shamir and can also be viewed as a deterministic version of Blakley's probabilistic method. Bounds on the maximum value of $n$ for a given $k$ and secret size are derived for any system, linear or nonlinear. The proposed scheme achieves the lower bound which, for practical purposes, differs insignificantly from the upper bound. The scheme may be extended to protect several secrets. Methods to protect against deliberate tampering by any of the trustees are also presented.

1. INTRODUCTION

CRYTOGRAPHY is extremely useful for making data files unintelligible to anyone who does not possess the secret key in which they were enciphered. But what happens if the legitimate owner of the file loses the key or is himself lost through incapacity or death?

There is a clear need for providing a backup copy of the key to protect against these eventualities. A safe deposit box can easily store a backup copy of the key on a punch card or similar data storage medium since most keys will be between 50 and 1000 bits long. But even a safe deposit box is vulnerable (e.g., to the "silverfish threat").

The advantage of this method is also a disadvantage: if even one of the $v_i$ is destroyed, the legitimate owner is unable to reconstruct the key from the remaining backup information.

The secret $s$ can be recovered even if $n - 1$ pieces have been destroyed, but theft of even one piece compromises the secret.

A different approach protects against the threat of theft, but aggravates the "silverfish threat." Divide the secret key $s$ into $n$ pieces $v_1, v_2, \ldots, v_n$ in a manner such that no information about $s$ is learned from any $n - 1$ pieces. This can be accomplished by letting $v_1$ to $v_{n-1}$ be independent random variables, uniformly distributed over $S$, the set of all possible secret keys, and letting

$$v_n = s + (v_1 + v_2 + \cdots + v_{n-1})(\mod q),$$

where $q = |S|$ is the cardinality of $S$. As a small example, when $q = 2^n$, $v_n$ is the exclusive or of $v_1$ through $v_{n-1}$. While a 1-bit key is of no value, the technique can be applied to successive bits of the key.

The advantage of this method is also a disadvantage: if even one of the $v_i$ is destroyed, the legitimate owner is unable to reconstruct the key from the remaining backup information.

Motivated by a desire to protect against both threats, several researchers have investigated the following secret sharing problem.