Information Routing and Reliability Issues in Distributed Sensor Networks

S. Sitharama Iyengar, *Senior Member, IEEE*, Mohan B. Sharma, and R. L. Kashyap, *Fellow, IEEE*

Abstract—**Communication issues and problems in information routing in distributed sensor networks (DSN's) are considered in this paper. We identify two important communication constraints, *viz.*, the delay constraint and the reliability constraint: the impact of these on information routing are discussed. It is shown that the maximum end-to-end delay in a network depends on the diameter of the network, and efficient distributed algorithms are presented to determine the diameter of asynchronous networks. We present a new distributed algorithm to determine the diameter of an asynchronous tree network, when an arbitrary node in the network initiates the algorithm. It is shown that the start node determines the diameter of the network in $2h_{\max}$ units of time, using exactly $2(n-1)$ messages where n is the number of nodes in the tree. Another efficient algorithm is presented to determine the diameter when multiple nodes initiate the algorithm. It is established that in the worst case, each start node determines the diameter in less than $2h_{\max}(S) + D$ units of time, using at most $4(n-1)$ messages, where $h_{\max}(S)$ is the maximum height of a subtree rooted at any start node, and D is the diameter of the tree. An algorithm to determine the diameter of arbitrary networks is presented, and its message complexity is shown to be $O(n, |E|)$ where $|E|$ is the number of edges in the network.**

The effects of link/node failures on network delay are studied, and important network structure design criterion incorporating a specified degree of reliability, and maximum end-to-end delay are discussed. Finally, the distributed, dynamic routing algorithms are reviewed, and their adaptations to the DSN environments are discussed.

I. Introduction

A. Distributed Problem Solving

In recent years, many multiprocessor architectures have been developed and studied for various applications. These architectures differ mainly in whether the supported processing is centralized or distributed. Advantages offered by distributed computing have been extensively studied and reported in the literature [4], [15], [22], [23], [25]. The main advantages include task decomposition, graceful degradation, reduced bandwidth requirements, and that these architectures provide a natural way to effect the powerful "divide-and-conquer" paradigm by solving the subproblems independently.

Based on a distributed processing domain, computations for solving a problem could be performed at various abstraction levels. Each level has a different conceptual content and generate a corresponding set of issues. Each subproblem solution at some particular abstraction level is called a knowledge source (KS). Distributed problem solving (DPS) is the cooperative solution of problems by a decentralized and loosely coupled collection of knowledge sources, to achieve a global solution to a problem [4], [22], [24]. The knowledge sources cooperate since no single one of them has sufficient information and authority to solve the entire problem; mutual sharing of information is necessary to enable the distributed system to produce an answer as a whole. The major difference between distributed processing and distributed problem solving is that in the former, we are mainly concerned with the architecture issues, load balancing, scheduling, processor interconnection, deadlock prevention, etc., while in the latter, we are faced with the problem of finding problem solving methods involving task decomposition, hypothesis, testing and reduction, data fusion, etc., on a distributed system domain. In this paper, we focus on the distributed problem solving techniques in the area of distributed sensing.

B. Distributed Sensor Networks

The distributed sensor network (DSN) consists of a set of diverse sensors geographically distributed, a communication network, and a set of processing elements trying to achieve a common goal. Fig. 1 shows the schematic of DSN. The purpose is to obtain an integrated picture of the area covered by the sensors. Motivation for DSN comes mainly from the need to increase the sophistication of surveillance systems and tracking mechanisms. Such systems require new architectures and strategies for detecting and tracking multiple targets using data from diverse sensors.

Design of spatially distributed sensing and decision involves the integration of solutions obtained by solving subproblems in data association, hypothesis, and data fusion. It is assumed that the sensors could overlap, and information at one sensor is not sufficient to solve the problem. The major reason for distributing sensing, communication, and processing is to achieve a high degree of reliable coverage and survivability of the system.

Manuscript received November 29, 1990; revised October 18, 1991. The work of S. S. Iyengar was supported in part by ONR Grant N00014-91-J-1306. The work of R. L. Kashyap was supported in part by the Innovative Science and Technology (IST) program of the SDIO monitored under Grant N00014-91-J-4126.

S. S. Iyengar is with the Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803.

M. B. Sharma was with the Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803. He is now with IBM Systems Division, Austin, TX.

R. L. Kashyap is with the Department of Electrical Engineering, Purdue University, West Lafayette, IN 47907.

IEEE Log Number 9203227.

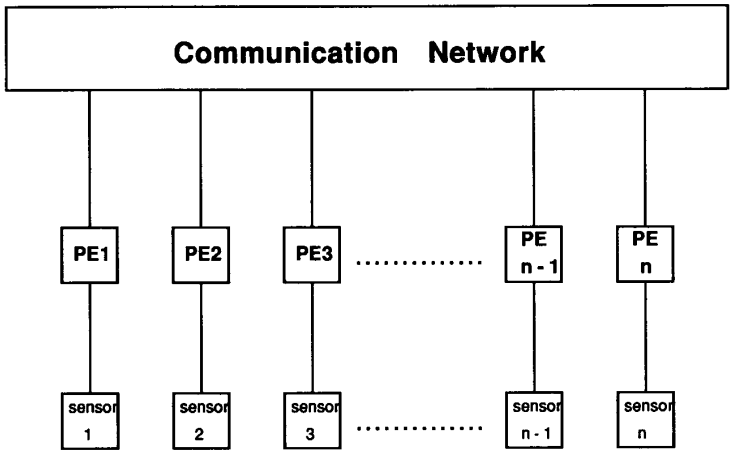

Fig. 1. A schematic of the distributed sensor networks.

One of the key issues in DSN is the communication between processors [26]. The principal component of DSN is the coordinated computation among the processors, which implies that communication among processors is the backbone of DSN. Each sensor acts as a knowledge source and communicates to some or all other nodes in the network, to initiate the inference process. The interaction among KS's is quite an expensive operation in distributed systems. It is important to minimize the interprocess communication and design efficient ways to route the information in the network.

C. Information Routing Issues in DSN

A processing node receives a bulk of data from the sensor it is associated with at regular intervals, generally at a fixed rate. After some amount of processing at the node, this information has to be sent to some or all other nodes in the network, depending on the problem solving technique. It is imperative that the information is routed to the destination nodes in an efficient manner since the data generation is repetitive.

Generally, data are transmitted to the destination nodes in packets. Some of the requirements in information routing in SDN are as follows.

1) It is desirable to have the entire information generated by a sensor, in one packet or in fewer packets; otherwise, loss of a packet or delay in receiving it might lead to discarding the entire data. This is a deviation from the packet switching needs in any conventional communication network. The main idea behind such a requirement is to speed up the inference process, and to reduce the queue sizes.

2) In most of the SDN applications, the sensor data will be generated and transmitted in each sensing cycle. Since the data exchange is almost continuous, the communication protocols should be designed such that an explicit ACKNOWLEDGE is not used for each packet. This saves enormous traffic on the network considering the size of DSN.

3) An immediate effect of not using acknowledge messages is with the old packets that have not yet been processed when a new packet has arrived. A simple solution to this problem is to ignore the old packets which could be identified by using time stamps in the message protocols. However, it is necessary to see that much data is not lost by ignoring old packets, and hence it is necessary to route the packets within a maximum allowable time.

4) DSN is envisaged to operate under hostile environments. It is therefore necessary to employ reliable point-to-point communication protocols. This topic has been well studied in the context of computer networks. These should be adapted to the DSN domains.

D. Scope and Organization of the Paper

Communication among the cooperating processors is the backbone of DSN, and in this paper we address the issues relating to the interprocess communication. There are two important questions that arise at each processing node. One of them is "what to communicate" and the other is "how to communicate." The former question is complex, problem dependent, and has been discussed in [16], [22]. In this paper, we focus on finding solutions to the latter questions of how to communicate. We present two important communication constraints, viz., the delay constraint, and the reliability constraint. It is established that delay in a network depends on the network diameter, and we present a family of distributed algorithms to determine the diameter. The failure of a node/link in the network increases the network diameter. We present bounds for node/link failures such that the increase in diameter results in acceptable delays. These results aid in the design of reliable structures for DSN.

In the next section, we review some DSN architectures and their properties. The communication constraints are studied in Section III and distributed algorithms are presented to determine the diameter of the graph underlying the network. The key consideration in designing topological configurations for DSN are discussed. In Section

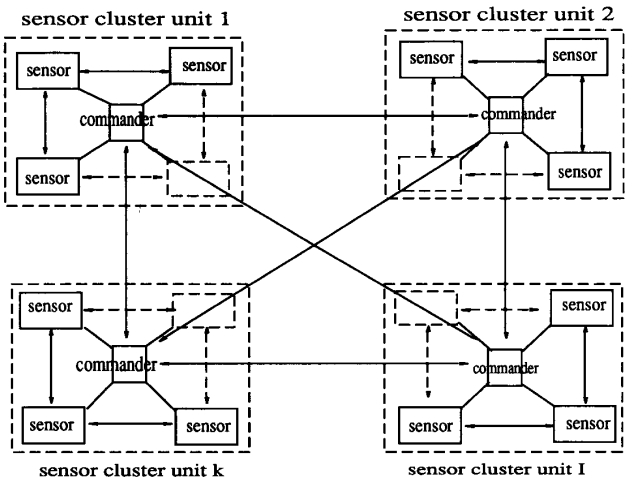

Fig. 2. Sensor domain partition.

IV, we review the distributed routing algorithms, and discuss their adaptation to DSN. The paper concludes with a summary and future research directions in Section V.

II. DSN Architecture

A fundamental issue in DSN development is determining the topological network configuration that best fits the application under consideration. This is a complex issue [10], [23], [26] since the interrelated constraints due to task decomposition, node interconnection, communication, fail safety, and integration problems need be considered while designing the network structure. In this section, we briefly review some network architectures for DSN.

A. Network Structures

Two types of topological configurations for DSN for situation assessment purposes have been studied by Wesson *et al.* [25]. The first of these is the "anarchic committee" organization where each node is able to communicate with the other. This arrangement resembles the "cooperating experts" paradigm in artificial intelligence (AI). In such an organization, task decomposition and communication issues are resolved dynamically and each processing node coordinates with some or all other nodes in the network and the solutions are arrived at, after some iterations of information processing, information exchanging and abstractions. No priority is assigned to any processor and all nodes operate autonomously.

The other topology studied in [25] is the hierarchial organization where the nodes are organized in strict hierarchies of abstraction levels. Each level will refine the information content of the data received from the lower levels to facilitate global inferencing, and transmit the abstracted information to the higher level nodes. Global inference is the responsibility of the nodes in the highest level which can exercise complete control over the network. Task decomposition, authority, and control details are resolved statically to each of the levels. In [25], it is shown that the performance of the anarchic committee is superior to that of the hierarchial organization. However, the same conclusion cannot be extended directly to DSN domains due to the small size of their experiment. Also, large bandwidth requirements of the committee structure pose a major problem for large networks such as DSN.

A novel approach for structuring distributed processing system is described by Lesser and Corkill [15]. The functionally accurate, cooperative (FA/C) organization differs from the conventional ones in its emphasis on handling distribution- caused uncertainty and errors as an integral part of the network problem solving process.

Iyengar and Sharma [10] developed a variation of the hierarchial organization which is a compromise between the hierarchial and the anarchic committee organization. In this hybrid structure, the sensor domain is partitioned into blocks of sensors called sensor cluster units (SCU) as shown in Fig. 2. Each SCU is organized internally in a hierarchial manner as shown in Fig. 3. SCU's are formed in such a way that integration of information from each sensor in an SCU is performed in the respective hierarchial system. Each "commander" node (the node at the highest level in SCU) of an SCU is connected to all other peer nodes in the sensor domain. In other words, an anarchic committee of SCU commander nodes is formed. Such an organization could be effectively used for situation assessment in a known problem domain. The hybrid network organization thus incorporates the advantages of both hierarchial and committee organizations and provides a natural way to implement the divide-and-conquer strategy.

III. Communication Constraints

Performance of any coordinated computing system depends on the communication and computational speeds. Communication poses a major problem since the time

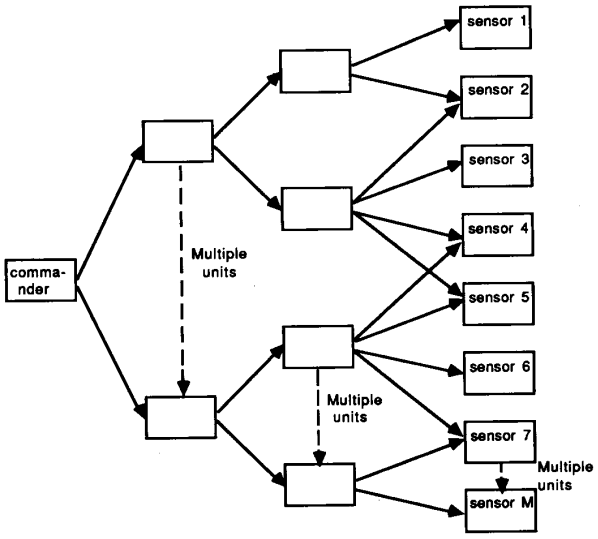

Fig. 3. Sensor cluster unit organization.

needed for communication is large in large networks. In DSN, the sensors generate data repetitively at regular intervals. It is therefore essential to ensure that each datum is delivered to the destination node in finite time, before the data of the next cycle arrived at the node. Hence there is an upper bound on the maximum allowable transmission time in the network. Also, considering the environment that DSN operates in, it is necessary to examine fail-safety requirements. In this section, we investigate mainly two constraints for communication, viz., the delay constraint and the reliability constraints. We present distributed algorithms for determining the maximum transmission delay in the network and study the connectivity requirements to achieve reliable communication in DSN. We also study the impact of node/link failures on the delay.

A. Delay Constraints

When a node in DSN needs to transmit information to other nodes in the network, the communication should be completed before another (set of) data on the next cycle arrives at the node. The problem could be stated slightly differently. A node n_i needs to transmit to other nodes, a unit of information, within time t_d where t_d is the maximum permissible transmission delay. The question is whether this is possible in the present configuration of the network. From time to time this check needs to be done in the network for proper functioning of DSN.

The factors that affect transmission delay in DSN include the capacity of the channel, number of lines on the route, propagation delay, and the packet size. From queuing theory results [3], we have an approximation for the mean total delay T,

$$T = L/\{C(1 - \rho)\} + t_p$$

where

L = mean length of frames,
C = channel capacity,
ρ = load on the line,
t_p = propagation delay.

For a network, the parameters L,C, and ρ are generally known to an acceptable degree of approximation. For a dynamic network, it is necessary to determine t_p. The propagation delay is dependent mainly on the length of the route, and hence the maximum end-to-end distance in the worst case measure of the propagation delay.

A network could be represented as a graph $G(V(G), E(G))$ where $V(G)$ is the set of vertices in the graph denoting the nodes in the network, and $E(G)$ is the set of edges in the graph denoting the communication links of the network. We assume that the links are bidirectional, and the network does not experience any failure during the execution of the algorithm. It is assumed that the messages are delivered to the other nodes in finite time and that there is no loss of messages.

We use the following notations, with respect to a graph $G(V, E)$.

$d(u, v)$ = the length of the shortest path between u and v; $u, v \in (V(G)$.

Definition 1: The diameter $D(G)$ of a connected graph G is the length of the longest $d(u,v)$, for all $u, v \in V(G)$, i.e.,

$$D(G) = \max_{uv \in V(G)} d(u, v).$$

Thus, the diameter of the graph underlying the network gives a measure of the maximum end-to-end transmission delay. We now present distributed algorithms to determine the diameter of the graph underlying a network.

First we consider asynchronous tree network and present diameter finding algorithms considering two cases: when one node starts the algorithm, and when multiple nodes start the algorithm. We discuss the extension of these algorithms to general networks. Diameter could also be determined using the center finding algorithm in [13]. However, the algorithm in [13] does not support multiple nodes starting, and these algorithms are not optimal for finding the diameter.

1) Distributed Diameter Finding Algorithms: We now present two algorithms, SDIA and MDIA, which determine the diameter of asynchronous tree networks when an arbitrary node starts the algorithm, and when an arbitrary number of nodes start the algorithm, respectively. The main idea behind these algorithms is to determine two largest heights, h_1 and h_2, say, of each node, and the node with the largest $h_1 + h_2$, determines the diameter of the tree. If the start node lies on the diameter path, then it computes the diameter, otherwise it receives the diameter information from a node that computed it. Following are the two messages used in these algorithms.

Messages:

Find: A message with only the message header.

Return (d, pd): Each return message is received with two parameters, the first, d, being the maximum height of the subtree rooted at the sender; the second, pd, being the maximum end-to-end distance at the sender.

a) Algorithm SDIA: Let $T(V(T), E(T))$ be the tree of order n, (i.e., $|V(T)| = n$). We assume that only one node starts the algorithm. Suppose a node s starts the algorithm. The tree T can now be viewed as a tree rooted at s with subtrees $T_1, T_2, \cdots, T_l$, $1 \leq l \leq n-1$; i.e., $T_s = \{T_1, T_2, \cdots, T_l\}$. It is clear that $V(T_i) \cap V(T_j) = \{s\}$, $i \neq j$. The algorithm is as follows. All nodes are initially in IDLE state. The start node s broadcasts the FIND message to its neighbors and changes to the WAIT state.

When an IDLE node receives the FIND message,

a) If it is a leaf node, then it sends the RETURN (d, pd) where d is 0 and pd is 0. It then quits the algorithm.

b) An internal node broadcasts the message to all its neighbors except the one from where it received the message and transits to the WAIT state.

When a node in WAIT state receives the RETURN (d, pd) message from a neighbor,

a) it increments the count d of the message and computes two largest d's, h_1 and h_2, (say) received so far. It computes the maximum of the pd's received.

and

b) if it has received the RETURN message from all its neighbors in response to the FIND message, then it sends RETURN (max (h_1, h_2), max $(pd, h_1 + h_2)$) to the neighbor from where it had received the FIND message. It then exits the algorithm. Otherwise, it remains in the same state.

The algorithm terminates when the start node s has received the RETURN message over all its neighbor nodes. Formal version of the algorithm is given in the Appendix A.

Properties: Suppose a node s starts the algorithm. Then,

P1) A node changes its state exactly once. Once from IDLE state to WAIT state and from WAIT state, it exits the algorithm. □

P2) A node ($\neq s$) in WAIT state does not receive a FIND message from any of its neighbors.

Proof: Suppose a node n_i in WAIT state receives a FIND message from its neighbor n_j; $n_i, n_j \in V(T)$, $i \neq j$. Let n_k be the node that sent FIND message to an IDLE n_i. Since s is the only initiator of the algorithm, there exist two paths from s to n_i: one path via n_j and the other via n_k. A contradiction to the assumption that T is a tree. □

Lemma 3A.1: The algorithm SDIA correctly determines the diameter of the tree, and terminates in finite time.

Proof: The FIND message, initiated at some start node s traverses down all the subtrees rooted at s, to each leaf. Each internal node n_i computes the maximum end-to-end distance of the subtree rooted at n_i in T_s (pd of the return message). Since the diameter of the graph is the maximum end-to-end distance, some internal node computes it. If s is on the diameter path, then s computes it, otherwise s receives the computed diameter information from a node on the diameter path, through its neighbor.

Since the RETURN message moves up the tree starting from the leaf, the start node receives the return message from all its neighbors, thus terminating the algorithm in finite time. □

Let $h(T_i)$ denote the height of the tree T_i, and $h_{\max} = \max\{h(T_j)|$ for all $T_j \in T_s\}$.

Theorem 3A.1: Suppose a node s starts the algorithm SDIA. Then, s determines the diameter of the tree T in at most $2h_{\max}$ units of time, if each message is delivered over a link in at most one unit of time.

Proof: The message FIND moves down the tree rooted at s and the RETURN message retraces the path of FIND message. Since each internal node broadcasts the FIND message, the maximum time it takes for the FIND message to reach a leaf is to the one at the farthest distance from s. Hence, the total time taken for finding diameter is $2h(T_i)$ where T_i is the subtree at s which has the maximum height. □

Theorem 3A.2: The algorithm SDIA finds the diameter of the tree T using exactly $2(n-1)$ messages and is optimal in message complexity.

Proof: It is clear from the algorithm SDIA that the FIND message travels on all the edges of the tree, and the RETURN message retraces the FIND message. Thus, there are two messages on each edge, and hence the algorithm uses exactly $2(n-1)$ messages.

For any diameter finding algorithm, the message has to be sent over each link of the tree and hence the problem of finding diameter has an obvious message complexity lower bound of $O(n)$. The algorithm is hence message optimal. □

b) Algorithm MDIA: We described a simple distributed algorithm to determine the diameter of a tree when a single node starts the algorithm. We now describe the generalization of this algorithm when multiple nodes start the process of finding the diameter.

Suppose a set of nodes $S = \{s_1, s_2, \cdots, s_k\}$, $S \subseteq V(T)$, start the process of finding the diameter. The algorithm proceeds as follows.

Each start node s_i sends the FIND message to all its neighbors. If s_i is a leaf, it behaves as if it has received a FIND message and sends the RETURN message to its neighbor.

When a node n_i in IDLE state receives a FIND message,

a) if n_i is a leaf, then it changes its state to ACTIVE and sends RETURN (d, pd) with $d = pd = 0$;

otherwise,

b) it changes the state to ACTIVE and broadcasts FIND message to all its other neighbors.

When a node n_i in IDLE state, receives a RETURN (d, pd) message;
{can happen when some leaf starts the algorithm}
a) if it has received RETURN from all its neighbors except one n_j, say, then it computes (d, pd) from the received messages and sends the RETURN (d, pd) to n_j. The node is set to ACTIVE state.
b) if it has received RETURN from all its neighbors, then the node transits to the TERMINAL state; computes pd from all the received messages and broadcasts the SETDIA (pd) message to all its neighbors and exits the algorithm.
otherwise
c) computes d_1, d_2 as the two largest d's received and the maximum of the received pd's.; changes to ACTIVE state; and broadcasts FIND message to all other neighbors.

When a node in ACTIVE state receives a FIND message, the message is ignored.
{implies that some other node/s started the algorithm in the subtree}

When a node n_i in ACTIVE state receives RETURN (d, pd) from a neighbor,
a) if the RETURN message is received from all its neighbors, it changes its state to TERMINAL and broadcasts the SETDIA (max (d_1 + d_2), max (pd)), message to all its neighbors and exits the algorithm.
b) if the RETURN message is received from all neighbors except one, n_j, then RETURN (d, pd) id sent to n_j.
Otherwise, computes d_1, d_2 and pd.

When a start node s_i receives a SETDIA(dia) message, it records the diameter and broadcasts the received message to other neighbors, and exits the algorithm.

The algorithm terminates after each node receives the SETDIA message. Formal version of the algorithm is given in Appendix B.

Lemma 3B.1: Suppose a set of nodes $S = \{s_1, s_2, \cdots, s_l\}$, $S \subseteq V(T)$, start the algorithm. There will be exactly one node or two adjacent nodes that reach the TERMINAL state, in finite time.

Proof: A node reaches TERMINAL state whenever it has received the RETURN message from all its neighbors. Also, a node sends RETURN message to its neighbor if and only if it has received RETURN from all its neighbors except one or if it is a leaf. So, if a node n_i that receives RETURN from a neighbor n_j, it is implied that the necessary computations are completed with the subtree rooted at n_j.

Suppose there are three nodes n_i, n_j and n_k, $i \neq j \neq k$ (not necessarily adjacent) that reach the TERMINAL state. This implies that RETURN message was sent by n_i to the subtrees that include n_j and n_k; and by n_j to n_i and n_k; and by n_k to n_i and n_j. A contradiction to the facts stated above.

It is clear that there could be one node that reaches the TERMINAL state. If two nodes n_i and n_j then both reach the TERMINAL state, both of them must have sent the RETURN message to each other, and hence must be adjacent. □

Theorem 3B.1: The algorithm MDIA correctly finds the diameter of the tree in finite time and terminates in finite time.

Proof: The FIND message is initiated by all the start nodes $S = \{s_1, s_2, \cdots, s_l\}$, $S \subseteq V(T)$. Each node $s_i \in S$ functions as it if is the only start node. A node sends the RETURN message only when it has received RETURN from all its neighbors except one. Hence, when a node receives the RETURN message from its neighbor, it is implied that all the necessary computations are completed in the subtree rooted at the neighbor. Each internal node determines the maximum end-to-end distance in the subtree. Some node n_j will eventually receive the RETURN message from all its neighbors and hence computes the diameter of the tree. All the nodes terminate the algorithm after receiving the SETDIA message, and hence the algorithm terminates in finite time. □

Let $h(T)$ denote the height of the tree T and $D(T)$ the diameter of the tree. Let $h_{\max} = \max\{h(T_i)$: for all $i \in S\}$.

Theorem 3B.2: Suppose nodes $S = \{s_1, s_2, \cdots, s_l\}$, $1 \leq l \leq n$, and $S \subseteq V(T)$ start the algorithm MDIA. Then, each of them will determine the diameter of the tree in less than $2h_{\max} + D(T)$ units of time if each message is delivered in one unit of time.

Proof: The maximum distance traversed by the FIND message originating at a start node s_i is equal to the maximum height of the subtree rooted at s_i. The RETURN message retraces this path, and these two messages together account for the first term in the time complexity. Once a node reaches the TERMINAL state, it broadcasts the SETDIA message to all other nodes in the network, which traverses less than the diameter path length which accounts for the second term in time complexity relation. □

Theorem 3B.3: The algorithm MDIA finds the diameter of the tree using at most $4(n - 1)$ messages.

Proof: In the worst case, all nodes in the tree might start the algorithm simultaneously and each will send the FIND message to their neighbors, except the leaf nodes which send the RETURN message. This amounts to $2(n - 1)$ messages. There will be $(n - 1)$ RETURN messages before a node can get to the TERMINAL state. A node in TERMINAL state broadcasts the SETDIA message which amounts for $(n - 1)$ messages, and hence the message complexity of the algorithm MDIA is at most $4(n - 1)$. □

Finding diameter distributively of an arbitrary graph is expensive in terms of communication complexity. The algorithm to determine the center of a graph in [13] could be employed to determine the diameter of an arbitrary graph. The algorithm is outlined as follows.

i) Construct a spanning tree of the graph rooted at a node s, using any of the existing algorithms [20].
ii) Each node in the tree then initiates a shortest path finding algorithm [14] and determines its eccentricity.

iii) Each node then computes its maximum end-to-end distance, and sends this information to the root.

iv) The root node upon receiving the end-to-end distance information from each of the nodes in the network computes the largest of these, which is the diameter of the graph. Then the diameter information is broadcast to all other nodes in the network.

The communication complexity of the algorithm is computed as follows. The first step requires $O(|E|)$ messages, and the shortest path finding algorithm initiated at each node requires $O(|E|)$ messages. Thus, the total number of messages required to compute the shortest paths by all nodes is $O(n.|E|)$. It needs $O(n^2)$ messages to perform step iii, and the last step of the algorithm requires $O(n)$ messages. Hence, the message complexity is $O(n.|E|)$, which in the worst case is $O(n^3)$.

B. Reliability Constraints

A vital consideration for automatic routing and proper functioning of DSN is the survivability of the network. Nodes and links in DSN may fail due to several reasons, considering the fact that DSN works in hostile environments. It is therefore necessary to maintain the message flow in the network. In this section, we describe the connectivity requirements that need to be taken into account while designing the topological structure for DSN that will achieve fail-safety. We also discuss the importance of these failures on the end-to-end delay in the network.

When a node/link fails, it is necessary to find alternate routes to route messages. Distributed protocols for such dynamic routing are discussed in the next section. However, it is necessary that the topological configuration of the network has sufficient connectivity such that when some links and/or nodes fail, the network remains connected. This requirement should be taken into consideration while designing network topologies. Suppose at most L links fail in the network. Then the graph underlying the network must be $(L + 1)$ edge connected for the network to remain connected despite the failure of L links. Similarly, for the network to remain connected if k nodes fail, the graph must be $(k + 1)$-connected. Thus, it is possible to keep the network connected despite link/node failures by increasing the connectivity of the graph. However, a node/link failure can increase the diameter of the graph, thus increasing the end-to-end delay in the network.

Suppose $D(G)$ is the diameter of the graph G underlying a network. We assume that we are given a k-connected graph of order n, and that there will be at most m node failures at any instant of time in the network. Let us now derive a condition for k such that under m node failures, the diameter of the network does not exceed the maximum permissible diameter D', for which the transmission delay does not exceed t_p. In [3], it is shown that deletion of m nodes in a k-connected graph of order n $(m < k)$, the diameter of the resulting graph D' is bounded by

$$D' \leq \frac{n - m - 2}{k - m} + 1.$$

It is clear that for D' to be within the delay constraint, under a fixed m, the connectivity k of the graph could now be computed using the above inequality, such that not only network connectedness is maintained, but also the node failures do not result in unacceptable delays. Also, for a λ edge-connected graph of order n, deletion of $\lambda - 1$ results in a graph of diameter D' such that

$$D' \leq \lambda D + \lambda - 1$$

where D is the diameter of the original graph [3].

We have shown the effect of link/node failures on the connectedness of the graph and the delay in the network. These results provide means for retaining the network connected, and also to ensure that the delay constraints for DSN are met with.

IV. Distributed Routing Algorithms

Under the problem solving strategies for DSN, it is imperative that efficient routing schemes are developed for information dissemination in the network. Each sensor generates a bulk of data which need be transmitted to some or all other nodes in DSN, depending on the problem solving technique. It is clear that the routing scheme depends on both topology and problem solving strategies. Topology changes could occur in DSN due to link failures, interceptions, jamming, etc. In this section, we review some of the fail-safe, dynamic, distributed routing protocols [1], [6], [11], [12], [17], [18], and discuss modifications of these algorithms to suit the DSN domains.

There are basically two methods for information routing in any network. First of these is by providing every node with the entire topological information; and the other method uses the routing tables, containing the shortest path information. Subtle issues in algorithms for these two methods have been discussed in [11].

Distributed topology learning algorithms have been discussed in [18], [19], [20]. Sharma *et al.* [20] present an efficient distributed topology learning algorithm which could be employed for DSN. However, due to the large size of DSN, it would be prohibitive to store the topology information at every node. Also, this scheme would increase the traffic on the network whenever topology updates have to be made. (Topology update messages are sent to nodes in the networks, when node/link failure occurs, and when a link/node comes up.)

The schemes that use shortest paths are better suited for DSN. However, dynamic determination of optimal routes is almost infeasible for large networks like DSN. The hierarchial routing schemes of Kleinrock and Kamoun [12] and Baratz and Jaffe [1] focus on solving the dynamic routing problem in large networks. The main idea is as follows. The network is partitioned in to set of disjoint clusters, and into k-levels. The shortest path information

with respect to a cluster, is maintained at each node in the cluster. A node also keeps an additional information which is the shortest path information to each supercluster. This scheme is shown to achieve optimality in path length, asymptotically.

The hybrid topology configuration discussed in Section II, is similar to the hierarchial structure discussed above. Each sensor cluster unit in the hybrid structure could be considered as the cluster of nodes in the scheme described in [12]. Due to the small size of an SCU, we could employ the cluster shortest path information at each node in cluster. Employing the algorithms in [1], [12], we could find an optimal routing path in DSN. Details of these algorithms are beyond the scope of this paper, and can be found in [1], [11], [12], [17]. For complexity analysis in routing algorithms see [27].

V. Discussion

We have discussed the communication issues in DSN, and addressed certain basic aspects that serve as the key design criteria for topological configuration of the network. Two important communication constraints were identified, namely, the delay and the reliability constraints. Study of delay is crucial to proper functioning of DSN, and we presented distributed diameter finding algorithms to determine the maximum end-to-end delay in the network. The algorithms SDIA and MDIA provide means to monitor the delay constraints in the network. It should be noted that these algorithms function correctly irrespective of whether the mode of communication is synchronous or asynchronous and properties of such algorithms are discussed in [21].

The hybrid network structure provides a basic structure for problem solving using the divide-and-conquer paradigm. This structure is shown to be convenient in adapting to the hierarchial, distributed dynamic information routing schemes described in [12].

The distributed algorithms presented in this paper assume that no node/link failures occur during the execution of the algorithm. Although the execution time of the algorithm is linear in the number of nodes, it is necessary to incorporate fail-safety strategies into the algorithm. It should also be noted that the diameter finding algorithm for an arbitrary graph is expensive in terms of message complexity. It is therefore necessary to design heuristic algorithms to determine the diameter, with low message complexities.

Appendix A

Notations used in the Appendixes are as follows.

MSG_j Message sent by node j.
Upon RCV (MSG_j)) Upon receiving message MSG from node j.

Variables at node i:

N_i = Set of neighbors of node i.
d_1, d_2 = Integers, initially 0.
lpd = Diameter, initially 0.
$State_i$ = State of the node, initially IDLE.
SF_i = Sender of FIND message, initially nil.
RC = Return message count, initially 0.

Algorithm at node $i \neq$ start node s:

When node i is in IDLE state;
⟨1⟩. Upon RCV ($FIND_j$)
⟨2⟩. **if** i = leaf node, **then**
send (RETURN (0, 0)) to j:
EXIT.
else
send (FIND) to all k, $k \in N_i j$.
$SF_i = j$; $State_i$ = WAIT;
When node i is in WAIT state;
⟨1⟩. Upon RCV (RETURN $(d, pd)_j$)
⟨2⟩. $d = d + 1$; {increment the distance}.
RC = RC + 1;
compute d_1 and d_2 as the two largest d's received
lpd = max (lpd, pd, $d_1 + d_2$);
⟨3⟩. if (RC = $|N_i| - 1$) then
send(RETURN (max (d_1, d_2), lpd) to SF_i;
EXIT.
Algorithm at node $i = s$, a start node.
⟨1⟩. Send(FIND) to all $k \in N_i$;
$state_i$ = WAIT;
Upon receiving the RETURN message, the algorithm differs in line ⟨3⟩ above and is shown below.
⟨3⟩. if RC = $|N_i|$ then
lpd = diameter of the tree;
EXIT.

Appendix B

The variables are the same as in Appendix A except that RC is a set variable.
Algorithm at node i:

When node i is in IDLE state;
⟨1⟩ Upon RCV($FIND_j$) or Upon START,
$State_i$ = ACTIVE;
if i = leafnode **then**
send(RETURN (0, 0)) to j;
else
$SF_i = j$ if i is not a start node;
send(FIND) to all k, $k \in N_i$ SF_i;
⟨1⟩. Upon RCV (RETURN $(d, pd)_j$)
{some leaf node started the algorithm}
⟨2⟩. $d = d + 1$;
RC = RC $\cup$ $\{j\}$;
compute d_1 and d_2 as the two largest received d's.
lpd = max (pd, lpd, $d_1 + d_2$);
⟨3⟩. **if** RC = N_i **then**
{lpd is the tree diameter}
$state_i$ = TERMINAL;
send(SETDIA (lpd) to all k, $k \in N_i$;
EXIT.

else
if $|RC| - |N_i| = 1$ **then**
{RETURN received from all but one neighbor}
$State_i$ = ACTIVE;
send(RETURN (max (d_1, d_2), lpd);
else
⟨4⟩.
$State_i$ = ACTIVE;
send(FIND) to all k, $k \in N_i \{ j \}$;

When node i is in ACTIVE state;
⟨1⟩. Upon RCV(FIND$_j$), ignore the message;
⟨1⟩. upon receiving the RETURN message, it behaves similar to an IDLE node except in step ⟨4⟩ above, where the send action is not performed.
⟨1⟩. Upon RCV(SETDIA(pd)$_j$)
$lpd = pd$; {diameter of the tree};
send(SETDIA (lpd)) to all k, $k \in N_i \{ j \}$.
EXIT.

Acknowledgment

The authors would like to thank all the reviewers and the Associate Editor Dr. J. A. Eldon for their constructive comments on this paper. More specifically, the authors would like to thank Dr. J. Chen for her stimulating comments on this paper.

References

[1] A. E. Baratz and J. M. Jaffe, "Establishing virtual circuits in large computer networks," *Comput. Networks*, vol. 12, pp. 27-37, 1986.
[2] J. A. Barnett, "DSN problems—an overview," in *Proc. Distributed Sensor Networks Workshop, CMU*, 1978, pp. 37-40.
[3] J. Bond and C. Pyerat, "Diameter vulnerability in networks," in *Graph Theory with Application to Algorithms and Computer Science*, Y. Alavi *et al.*, Eds. New York: Wiley, 1985, pp. 125-149.
[4] B. Chandrasekaran, "Natural and social system metaphors for distributed problem solving: Introduction to the issue," *IEEE Trans. Syst., Man, Cybern.*, pp. 1-4, Jan 1981.
[5] Y. Cheng and T. G. Robertzani, "Communication and computation tradeoffs for a network of intelligent sensors," in *Proc. IEEE Comput. Networking Symp.*, Apr. 1988, pp. 152-161.
[6] I. Cidon and R. Rom, "Fail-safe end-to-end protocols in computer networks with changing topology," *IEEE Trans. Commun.*, vol. COM-35, no. 4, pp. 410-413, Apr. 1987.
[7] D. Cohen, "Protocols for DSN," in *Proc. DSN Workshop, CMU*, 1978, pp. 124-125.
[8] R. E. Cullingford, "Integrating knowledge sources for computer understanding tasks," *IEEE Trans. Syst., Man, Cybern.*, pp. 52-60, Jan. 1981.
[9] L. R. Erman *et al.*, "The Hearsay-II Speech Understanding System: Integrating knowledge to resolve uncertainty," *Comput. Surveys*, vol. 12, pp. 213-254, 1980.
[10] S. S. Iyenagar and M. B. Sharma, "Information routing in distributed sensor networks," in *Proc. Indo-U.S. Workshop* (New Delhi, India), Nov. 27-29, 1989.
[11] J. M. Jaffe, A. E. Baratz, and A. Segall, "Subtle design issues in the implementation of distributed dynamic routing algorithms," *Comput. Networks*, vol. 12, no. 1, pp. 146-158, Aug. 1986.
[12] L. Kleinrock and F. Karmoun, "Hierarchial routing for large networks—performance evaluation and optimization," *Comput. Networks*, vol. 1, no. 3, pp. 155-174, Jan. 1977.
[13] E. Korach, D. Rotem, and N. Santoro, "Distributed algorithm for finding centers and medians in networks," *ACM Trans. Programming Languages Syst.*, vol. 6, no. 3, pp. 380-401, July 1984.
[14] K. B. Lakshmanan, K. Thulasiraman, and M. A. Comeau, "An efficient distributed protocol for finding shortest paths in networks with negative weights," *IEEE Trans. Software Eng.*, vol. 15, no. 5, pp. 639-644, May 1989.
[15] V. R. Lesser and D. D. Corkill, "Functionally accurate, cooperative distributed *D* systems," *IEEE Trans. Syst., Man, Cybern.*, pp. 81-96, Jan. 1981.
[16] V. Lesser, D. Corkill, J. Pavlis, L. Lefkowitz, E. Hudlicka, and R. Brooks, "A high level stimulation test bed for cooperative distributed problem solving," in *Proc. DSN Workshop* (M.I.T. Lincoln Laboratory, Lexington, MA), Jan. 6-7, 1982, pp. 247-270.
[17] P. M. Merlin and A. Segall, "A fail-safe distributed routing protocol," *IEEE Trans. Commun.*, vol. COM-27, no. 9, pp. 1280-1287, Sept. 1979.
[18] M. Raynal, *Networks and Distributed Computation: Concepts, Tools and Algorithms*. Cambridge, MA; M.I.T. Press, 1988.
[19] A. Segall, "Distributed network protocols," *IEEE Trans. Inform. Theory*, vol. IT-20, no. 1, pp. 23-35, Jan. 1983.
[20] M. B. Sharma,S. S. Iyengar, and N. K. Mandyam, "Efficient distributed depth-first-search algorithms," *Inform. Processing Lett.*, vol. 32, no. 4, pp. 183-186, Sept. 1989.
[21] D. N. Jayasimha, S. S. Iyengar, and R. L. Kashyap, "Information integration and clock synchronization on DSN," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 5, Sept-Oct. 1991.
[22] R. G. Smith, *A Framework for Distributed Problem Solving*. Michigan: UMI Press, 1981.
[23] R. Smith and R. Davis, "Distributed problem solving: The contract net approach," in *Proc. Second Nat. Conf. Canadian Soc. Computat. Studies Intell.* (Toronto, Canada), July 1978, pp. 278-287.
[24] R. G. Smith, "Frameworks for cooperation in distributed problem solving," *IEEE Trans. Syst., Man, Cybern.*, pp. 61-70, Jan. 1981.
[25] R. Wesson *et al.*, "Network structures for distributed situation assessment," *IEEE Trans. Syst., Man, Cybern.*, pp. 5-23, Jan. 1981.
[26] Y. Yemini, "DSN, an attempt to define issues," in *Proc. DSN Workshop, CMU*, 1978, pp. 53-66.
[27] M. B. Sharma, J. Chen, and S. S. Iyengar, "Distributed algorithms for locating centers and medians in communication networks," submitted for publication.

S. Sitharama Iyengar (M'88-SM'89) received the Ph.D. degree in 1974 from Mississippi State University.

He is currently a Professor and Chairman of the Department of Computer Science at Louisiana State University, and has been the Director of the Robotics Research Laboratory since its inception in 1986. One of his major research areas has been high performance algorithms and data structures. He has been the principal investigator on research projects supported by the Office of Naval Research; National Aeronautics and Space Administration; National Science Foundation/Laser Programme; Jet Propulsion Laboratory—California Institute of Technology; Department of Navy-NORDA; Department of Energy through the Oak Ridge National Laboratory; LEQFS-Board of Regents; and the APPLE Company Inc. He has coauthored two volumes of *A Tutorial on Autonomous Intelligent Systems* (IEEE Computer Society Press, to be published) and has edited two books and over 150 publications, including 85 archival journal papers on high performance parallel and distributed algorithms and data structure for image processing and pattern recognition; autonomous navigation; and distributed sensor networks. He was a Visiting Professor (fellow) at the Jet Propulsion Laboratory—California Institute of Technology, Oak Ridge National Laboratory, and Indian Institute of Science, India. He is also an ACM National Lecturer. He is a Series Editor of Neurocomputing of Complex Systems and Area Editor of the *Journal of Computer Science and Information*. He was a Coguest Editor of the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING (1988), IEEE COMPUTER MAGAZINE (1989), IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, IEEE TRANSACTIONS ON DATA AND KNOWLEDGE ENGINEERING and the *Journal of Computers and Electrical Engineering*.

Dr. Iyengar was awarded the Phi Delta Kappa Research Award of Distinction at LSU (1989). He won the Best Teacher Award in 1978. He was awarded the Williams Evans Fellowship of the University of Ottago, New Zealand, in June 1991.

Mohan B. Sharma received the bachelor's and master's degrees, both in electronics engineering, from Mysore University, India, and the Ph.D. degree from the Department of Computer Science, Louisiana State University, Baton Rouge, in 1990.

Currently, he is a member of the Distributed Processing Group of IBM in Austin, TX.

R. L. Kashyap (M'70–SM'77–F'80) received the Ph.D. degree from Harvard University in 1966.

He is currently a Professor of Electrical Engineering and Associate Director of the Engineering Research Center on Intelligent Manufacturing Systems at Purdue University, West Lafayette, IN. He has held visiting professor positions at Harvard University, the University of California, Berkeley, and the Indian Institute of Technology, Kanpur. He is an Area Editor for the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, *CVGIP: Graphical Models and Image Processing*, and *The Journal of Intelligent and Robotic Systems*. He has been Guest Coeditor of the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING (1988), the IEEE COMPUTER MAGAZINE (1989), and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS (1991). He has authored one book and over 300 publications, including 90 archival journal papers in areas such as pattern recognition and image processing, signal processing and system identification, random field models, intelligent data bases, and intelligent manufacturing systems.

Dr. Kashyap received the 1990 King Sun Fu Award given by the International Association for Pattern Recognition for fundamental contributions to pattern recognition and computer vision, the fellowship of the Institute of Electrical and Telecommunication Engineers (India), the Best Research Paper Award at the National Electronics Conference (1967), and the J. C. Bose Award for the best engineering science paper appearing in the *Journal of the Institute of Electrical and Telecommunication Engineers* (1991).