Computing Reliability and Message Delay for Cooperative Wireless Distributed Sensor Networks Subject to Random Failures

Hosam M. F. AboElFotoh, *Member, IEEE*, S. S. Iyengar, *Fellow, IEEE*, and Krishnendu Chakrabarty, *Senior Member, IEEE*

***Abstract*—One of the most compelling technological advances of this decade has been the advent of deploying wireless networks of heterogeneous smart sensor nodes for complex information gathering tasks. A wireless distributed sensor network (DSN) is a *self-organizing*, ad-hoc network of a large number of cooperative intelligent sensor nodes. Due to the limited power of sensor nodes, energy-efficient DSN are essentially *multi-hop* networks. The self-organizing capabilities, and the cooperative operation of DSN allow for forming reliable *clusters* of sensors deployed near, or at, the sites of *target phenomena*. Reliable monitoring of a phenomenon (or event detection) depends on the collective data provided by the *target cluster* of sensors, and not on any individual node. The failure of one or more nodes may not cause the operational data sources to be disconnected from the *data sinks* (*command nodes or end user stations*). However, it may increase the number of hops a data message has to go through before reaching its destination (and subsequently increase the message delay). In this paper, we focus on two related problems: computing a measure for the reliability of DSN, and computing a measure for the expected & the maximum message delay between data sources (sensors) & data sinks in an operational DSN. Given an estimation of the failure probabilities of the sensors, as well as the intermediate nodes (nodes used to relay messages between data sources, and data sinks), we use a probabilistic graph to model DSN. We define the DSN reliability as the probability that there exists an operating communication path between the *sink* node, and at least one operational sensor in a *target cluster*. We show that both problems are #P-hard for arbitrary networks. We then present two algorithms for computing the reliability, and the expected message delay for arbitrary networks. We also consider two special cases where efficient (polynomial time) algorithms are developed. Finally, we present some numerical results that demonstrate some of the applications of our algorithms.**

***Index Terms*—Clustering, distributed sensor networks, expected message delay, expected network diameter, graph-theoretic algorithms, multi-hop networks, probabilistic graph models, reliability, wireless sensor networks.**

Acronyms[1]

DSN	wireless distributed sensor network
GPS	global positioning system
RBN	radio broadcast network
MTTF	mean time to fail
EU	end-user node
BS	base station
GW	gateway node
iff	if and only if

Notations

G(V, E)	a DSN graph
V	set of nodes representing DSN nodes
E	set of edges, $(\mathrm{i},\mathrm{j}) \in \mathrm{E}$ iff i and j are one-hop neighbors, $\mathrm{i} \in \mathrm{V}$, $\mathrm{j} \in \mathrm{V}$
s	a sink node (EU or BS), $\mathrm{s} \in \mathrm{V}$
T	target set of sensors, $\mathrm{T} \subset \mathrm{V}$
$\mathrm{p_v}$	Pr{operation of node $\mathrm{v} \in \mathrm{V} - \{\mathrm{s}\}$}
Rel(G)	Pr{there exists an operational path between s, and an operational sensor $\mathrm{t} \in \mathrm{T}$}
$\delta(\mathrm{G})$	expected length of a shortest operational path between s, and an operational $\mathrm{t} \in \mathrm{T}$
$\lambda(\mathrm{G})$	maximum length of a shortest operational path between s, and an operational $\mathrm{t} \in \mathrm{T}$

I. Introduction

ONE OF THE most compelling challenges for this decade is to design optimal methods for exploiting the new realm of distributed sensor networks (DSN). A DSN consists of a large number of *intelligent* sensor nodes distributed over a widely spread geographical area, providing real-time information about environmental conditions. The sensors detect & measure a certain (*target*) phenomenon via its changing parameters [3]–[5], [22], [24]. Typical applications of DSN include military operations, area surveillance, environmental monitoring, remote sensing, and global awareness. An important feature of DSN is self-organizing capabilities that allow random deployment of the sensors, and dynamic reconfiguration of the network topology in the presence of sensor failures or replacements [3], [4]. Another unique feature of DSN is the cooperative effort of its nodes. A cluster-based network architecture is used in implementing several collaborative signal processing applications. To ensure high reliability and

Manuscript received November 16, 2002. This work was supported in part by DARPA under Grant N660010018946 and in part by the Office of Naval Research under Grant N000140110712. Associate Editor: W. H. Sanders.

H. M. F. AboElFotoh is with the Department of Mathematics & Computer Science, Kuwait University, Safat, Kuwait 13060 (e-mail: hosam@sci.kuniv.edu.kw).

S. S. Iyengar is with the Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803 USA (e-mail: iyengar@bit.csc.lsu.edu).

K. Chakrabarty is with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: krish@ee.duke.edu).

Digital Object Identifier 10.1109/TR.2004.842540

[1]The singular and plural of an acronym are always spelled the same.

fault-tolerance, clusters of large numbers of low-cost sensors are used redundantly together with methods for information integration/fusion (aggregation), and synchronization [7], [10], [16]. The sensors exchange data and/or autonomous agents [25] via wireless links (radio packets). An intelligent sensor is essentially a processing element that is equipped, in addition to its sensing peripherals, with a radio transmitter/receiver. Technology advances now allow manufacturing low-power sensor nodes with signal processing, wireless communications, power sources, and synchronization, with better performance-to-cost ratios [2], [26]. Each sensor can communicate with neighboring sensors within a specified range R (assuming omnidirectional antennas). Due to the limited power of the sensors, DSN require multi-hop operation to avoid sending large amounts of data over long distances [2], [24], [26]. Although the main function of sensors is information gathering, sensors can act as *repeater* nodes to relay (forward) messages toward the data sinks (or end user) [3]. To reduce power consumption at the sensor nodes, contention-free protocols have been proposed (e.g. using Time-Division Multiple Access TDMA [3], [15]).

Due to the harsh nature of the DSN applications environment, the sensors are subject to random failures due to different reasons. Consider, for instance, parachuting sensors over an enemy jungle. Failures are also caused by component wear out, power failures, software bugs, and in some cases by natural catastrophes or radio jamming like in a war-case scenario. Reliable monitoring of a phenomenon (or event detection) depends on the collective data provided by the target cluster of sensors, and not on any individual node. Therefore, the failure of one or more nodes may not cause the operational data sources to be disconnected from the *data sinks* (*command nodes or end user stations*). However, operational sensors in the faulty sensors neighborhood may still be able to communicate with end-users, although, through a larger number of hops resulting in a larger delay of the information. Many routing schemes have been developed [9], [12], [19], [29] which are fault-tolerant, such that a message is guaranteed to reach its intended destination(s) as long as an operational path exists.

In this paper, we consider the problem of computing a measure for the reliability of DSN. We define the *DSN reliability* as the probability that there exists an operating communication path between the *sink* node, and at least one operational sensor in a *target cluster*. We also consider the related problem of computing a measure for a message delay in an operational DSN. A related measure is the diameter of the network, which is the maximum number of hops between any pair of source/sink nodes.

We assume that geographical locations of the nodes are known, either through Global Positioning System (GPS), or through RF-based beacons [6]. Given the location information of each node, and the transmission range, we can determine the topology of the DSN. We use a probabilistic graph model to represent a DSN subject to random failures; and using this model, we investigate the complexity of the two problems, and present methods for computing the reliability, and the expected (and the maximum) message delay (in an operational network). In Section II, we present the graph model, assumptions, and a formulation of the problems. In Section III, we show that both problems are computationally intractable, in particular #P-hard for arbitrary networks. Section IV describes the algorithms for arbitrary wireless networks. In Section V, we consider two special cases for which efficient algorithms are developed. In Section VI, we present some numerical results demonstrating an example application of our algorithms. Section VII contains conclusions, and notes on future work.

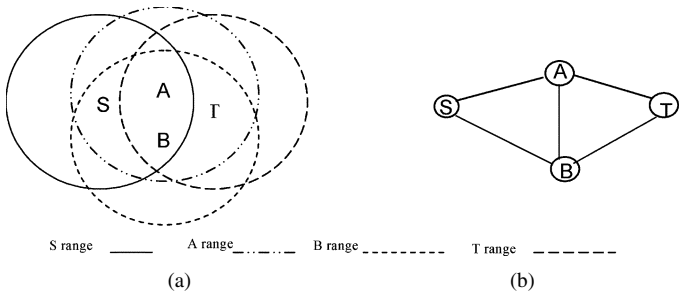

Fig. 1. (a) A wireless network; (b) The graph model.

II. The DSN Model

A DSN is modeled by a probabilistic graph $G = (V, E)$, where every node in the network is represented by a node in V. An edge exists between two nodes iff the two corresponding sites are in the range of each other (*one-hop neighbors*). We associate with every node $v \in V$ an operational probability p_v (failure probability $q_v = 1 - p_v$). We assume that the node failures are statistically independent. Fig. 1 shows a wireless network, and the corresponding graph model. Probabilistic graph models have been used extensively in the literature for studying network reliability problems [1], [8]. In particular, the above model has been used in [1] for studying radio broadcast network (RBN) reliability problems. RBN is an old name for wireless networks (in which nodes communicate through radio transmitter/receivers). We assume that the node mean-time-to-fail (MTTF) is relatively large compared to the message transmission time, the maximum propagation delay, and the time required by the network to adapt to topology changes due to failures. Although sensors are hardly repairable, sensors may be replaceable. In this case, we also assume a relatively large mean-time-to-replace. Therefore, during the message transmission, the state of the network is uniquely determined in terms of failed nodes, and operating ones.

A cooperative DSN uses a fault-tolerant *clustering* protocol [3], [17], [31] that is re-executed in the event of a topology change. This protocol achieves the following goals:

- A cluster is composed of a number of sensor nodes, which are deployed either inside the phenomenon, or very close to it. The sensors within one cluster provide a reliable source of data related to that phenomenon. The failure of a sensor should not disrupt the network operation.
- Each cluster has a *cluster head* which coordinates transmissions within the cluster, and is responsible for routing packets between sink nodes, and sensors (cluster heads may have more functionality, e.g. location awareness, and a longer-range radio [31]). However, because the failure of a cluster head will

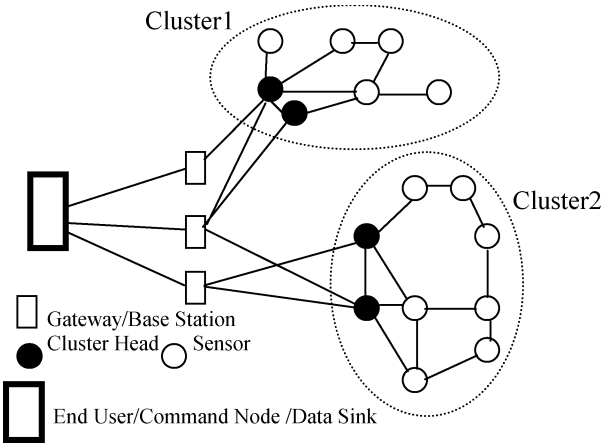

Fig. 2. Clustering, data sink, cluster head, gateway, and sensor nodes.

cause the rest of the cluster to be disconnected from the network, a cluster may be formed with more than one head redundantly used by the end user or base station.

- A *virtual backbone* is formed to connect cluster heads to the *data sinks* (the end user stations or base stations). In the remainder of the paper, we alternatively use the terms *data sink*, and *end user* to refer to a node with high power, and computational resources that can be "a command center," or "a base station," or "an end user station." Nodes on the backbone that are neither cluster heads, nor are data sinks, function as *gateways*, and may belong to one or more clusters (see Fig. 2).
- We assume a contention-free protocol, such as TDMA/FDMA-based protocol, Time Division Multiple Access/Frequency Division Multiple Access [3], [15], and that redundant messages are detected, and ignored.

Assuming identical processing time at all nodes, the message delay between two nodes s & t can be measured as the number of hops. For example, if the message has to go through two nodes to reach t, then the delay is three hops.

A. Reliability and Delay Measures for DSN

In a cooperative DSN, the reliable monitoring of a phenomenon is based on the collaborative operations of sensors within the cluster. Therefore, the conventional end-to-end reliability definition may not be applicable. The terms event-to-sink [28], and sink-to-sensors reliability [23] have been recently proposed in the literature. In [28], a reliable event-to-sink transport protocol is proposed, where the reliability is measured as the number of messages (packets) received by the sink in a predetermined *decision interval*. In [23], a brief discussion (a *Poster*) about some reliability issues is presented. Motivated by the above discussion, we consider the network to be functioning (operational) if the end user station monitoring a certain *target phenomenon* has an operational bi-directional connection path to at least one operational sensor in the cluster assigned to that phenomenon. We call such cluster a *target cluste*r. Further we assume that within each cluster a set of sensor nodes are considered critical to the monitoring operation (or event detection). We call such sensors *target sensors*. Based on the above assumption we define the reliability of DSN as follows:

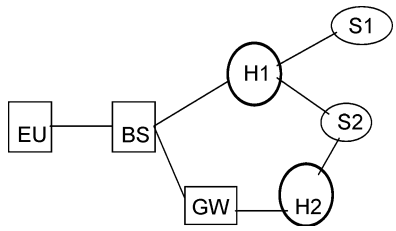

Fig. 3. DSN graph.

Definition 1: For any end user node monitoring certain (target) phenomena, the *reliability* of a DSN is the probability that there exists an operational path between the end user node (sink node), and at least one operational target sensor in the target cluster. □

Definition 2: (DSN Reliability Problem DSNREL): Given a DSN graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, a sink node s, a target set of sensors $\mathrm{T} \subset \mathrm{V}$, and an operational probability $\mathrm{p_v}$ for every $\mathrm{v} \in \mathrm{V} - \{\mathrm{s}\}$, compute the probability that there exists an operational path between s, and any operational target sensor $\mathrm{t} \in \mathrm{T}$, denoted by Rel(G). □

Definition 3: The expected message delay between a sink node, and the target phenomenon in an operational DSN is the expected value of the number of hops between the end user node, and any operational sensor in the target set of sensors in the corresponding target cluster given that the network is operational (i.e. given that there exists an operational path). □

Definition 4: (Computing Sensor Message Delay CSMD): Given a DSN graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, a sink node s, a target set of sensors $\mathrm{T} \subset \mathrm{V}$, and an operational probability $\mathrm{p_v}$ for every $\mathrm{v} \in \mathrm{V} - \{\mathrm{s}\}$, let l be the length of a shortest operational path between s, and an operational $\mathrm{t} \in \mathrm{T}$, compute the expected value of l (denoted by $\delta(\mathrm{G})$), and the maximum value of l (denoted by $\lambda(\mathrm{G})$). □

To illustrate the definitions, we consider a simple DSN with an end-user node (**EU**), one intermediate base station (**BS**), one gateway node (**GW**), and a target cluster with two heads & two target sensor nodes. Let the two cluster heads be **H1**, and **H2** with operational probabilities $\mathrm{P_{H1}}$, and $\mathrm{P_{H2}}$ respectively; and the two target sensors be **S1**, and **S2** with operational probabilities $\mathrm{P_{S1}}$, and $\mathrm{P_{S2}}$ respectively. Although base station nodes, and the gateway nodes are usually more reliable, compared to the sensor nodes, we may assume that BS, and GW have the operational probabilities $\mathrm{P_B}$, and $\mathrm{P_G}$ respectively. Therefore, the reliability of the DSN represented by the graph G in Fig. 3, Rel(G), is

$$\mathrm{Rel(G)} = \mathrm{P_B}\left(\mathrm{P_{H1}}\left(1-(1-\mathrm{P_{S1}})(1-\mathrm{P_{S2}})\right) + (1-\mathrm{P_{H1}})\mathrm{P_G}\mathrm{P_{H2}}\mathrm{P_{S2}}\right)$$

Note that, in this example, we assume that the operation of either S1, or S2 is *critical* to the operation of the network; and that H1, and H2 are relaying only data to the sink, besides their coordination function as cluster heads.

The minimum delay is 3 hops (ES-BS-H1-S1, or S2). The probability of having a delay of 3 denoted by $\mathrm{P}(3) = \mathrm{P_B}(\mathrm{P_{H1}}(1-(1-\mathrm{P_{S1}})(1-\mathrm{P_{S2}}))$.

The maximum delay is 4 (ES-BS-GW-H2-S2). The probability of having a delay of 4 denoted by $\mathrm{P}(4) = \mathrm{P_B}(1 - \mathrm{P_{H1}})\mathrm{P_G}\ \mathrm{P_{H2}}\ \mathrm{P_{S2}}$.

The expected delay given that the network is operational is $(3\mathrm{P}(3) + 4\mathrm{P}(4))/\mathrm{Rel(G)}$.

As a numerical example, let $\mathrm{PB} = 1$, $\mathrm{P_{H1}} = \mathrm{P_{H2}} = \mathrm{P_G} = 0.6$, $\mathrm{P_{S1}} = \mathrm{P_{S2}} = 0.5$

$$\mathrm{Rel(G)} = 0.522, \quad \lambda(\mathrm{G}) = 4$$
$$\delta(\mathrm{G}) = \frac{(3 \times 0.45 + 4 \times 0.072)}{(0.45 + 0.072)} = 3.138.$$

The expected delay measure tells us that most of the time the delay will be 3 hops per message, and on average, for a measurement that requires 1000 messages, the expected delay is 3138 hops.

III. Complexity of DSNREL and CSMD

A. Complexity of Computing the Reliability of DSN (DSNREL)

The complexity of DSNREL follows from the complexity of computing the two-terminal reliability of a wireless network (2REL), which has been shown in [1] to be #P-complete even when restricted to the case of equal operation probabilities. The #P class contains the counting version of problems in NP [8], [13]. The 2REL problem is described as follows. The network is modeled by a graph $G = (V, E)$, where every site is represented by a node $v \in V$, and an edge $(i, j) \in E$ iff i & j can directly communicate with each other. Given a graph model for a wireless network $G = (V, E)$, and a pair of nodes s & t, compute the probability that there exists at least an operational path from s to t.

Theorem 1: Computing the reliability of a DSN (DSNREL) is #P-hard.

Proof: Suppose we have an efficient algorithm for DSNREL. We transform an instance of computing the two terminal reliability of a wireless network (2REL) to DSNREL. The 2REL problem is transformed into DSNREL by considering s to be the end user node, and t to be its target cluster. The target cluster contains one sensor node with operational probability $= 1$. However, 2REL has been shown in [1] to be #P-complete, which is a contradiction. □

B. Complexity of Computing Sensor Message Delay (CSMD)

Given a graph $G = (V, E)$, a sink node s, and a target sensor set T, let $P(l)$ be the probability that the shortest operational s, t-path is of length l, $1 \leq l \leq n-1$, where $t \in T$, $(n = |V|)$. Let $P(\infty)$ be the probability that there exists no path of operating nodes between s & t, i.e. s & t are disconnected, therefore $P(\infty)$ is given by the equation

$$P(\infty) = 1 - \sum_{l=1}^{l=n-1} P(l). \tag{1}$$

The expected message delay δ between the sink node s, and T, given that there exists an s, t-path, can then be computed by

$$\delta = \frac{\left(\sum_{l=1}^{l=n-1} lP(l)\right)}{\sum_{l=1}^{l=n-1} P(l)}. \tag{2}$$

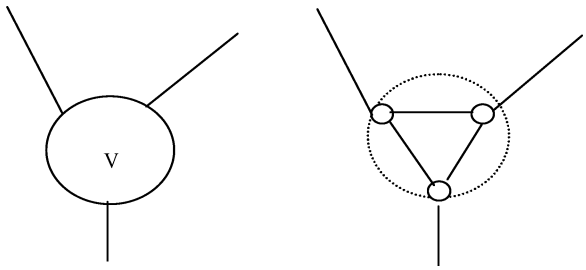

Fig. 4. Transforming v in $\mathrm{G_i}$.

Note that the message always takes the shortest available path; duplicate messages arriving afterward are discarded. If more than one shortest paths exist, the message takes arbitrarily any one of them depending on the routing schemes used. We show in the following section that computing δ for an arbitrary network is a computationally intractable problem, in particular CSMD is #P-hard.

We investigate the complexity of computing the expected delay in a DSN by considering the special case of an end user node (s) with one target sensor node $(\mathrm{T} = \{t\})$ with operational probability $= 1$. Further, we consider the more restricted case of computing δ where nodes have equal operational probabilities. For a given wireless network G, the two-terminal reliability $Rel_2(G)$ is the probability that there exists an s, t-path of operating nodes in G. Therefore

$$Rel_2(G) = 1 - P(\infty) = \sum_{l=2}^{n-1} P(l) \tag{3}$$

assuming $P(1) = 0$, i.e. s & t can not communicate directly. The probability $P(2)$ can be computed in polynomial time using the equation $P(2) = 1 - q^k$, where k is the number of s, t-paths of length 2. This number can be computed in a polynomial time by successively finding the shortest path in G using standard techniques (e.g. breadth first search [14]). If this path has length 2, we delete it, and repeat the process until no more s, t-paths of length 2 can be found.

Theorem 2: Computing $Rel_2(G)$ (2REL) is polynomialy reducible to computing δ.

Proof: An instance of 2REL consists of a probabilistic graph $G = (V, E, p)$, $|V| = n$, with two distinguished nodes s (source), and t (destination). Each node in $V - s - t$ is assigned an operation probability p; we assume that p is a rational number of length $\Theta(n)$. The failure probability is denoted by $q = 1 - p$. Given an instance of 2REL, we transform it into computing δ as follows. We consider the set of graphs $G_1, G_2, \ldots, G_{n-4}$, where G_i, $1 \leq i \leq n-4$, is constructed from G by replacing every vertex $v \in V - \{s, t\}$ by a complete graph of d_v vertices, each with operational probability p_i, where d_v is the degree of v (see Fig. 4). We select the probability values such that $0 < p_1 < p_2 < \ldots < p_{n-4} < 1$.

Every operational path in G of length l corresponds to an operational path in G_i of length $l + l - 1 = 2l - 1$, because of the additional $l - 1$ edges. Let $P_i(l)$ denote the probability that the shortest path in G_i is of length l.

Then

$$P_i(3) = p_i P(2)$$

and

$$P_i(2l-1) = (p_i)^{l-1}P(l), \quad \text{for } 3 \le l \le n-1.$$

Let δ be the expected message delay for G, and δ_i be the expected message delay for G_i. Applying (2) to G

$$\delta \sum_{l=2}^{l=n-1} P(l) = \sum_{l=2}^{l=n-1} lP(l).$$

Hence

$$(\delta - 2)P(2) = \sum_{l=3}^{l=n-1} (l-\delta)P(l). \tag{4}$$

Similarly, for G_i

$$(\delta_i - 3)P_i(3) = \sum_{l=5}^{l=2n-3} (l-\delta_i)P_i(l). \tag{5}$$

Substituting for $P_i(l)$ in terms of $P(l)$, we get

$$(\delta_i - 3)p_iP(2) = \sum_{l=3}^{l=n-1} (2l-1-\delta_i)p_i^{l-1}P(l). \tag{6}$$

Having algorithms for computing δ and $P(2)$, we can obtain from (4) & (6) a system of $n-3$ linear equations in $P(l)$, $3 \le l \le n-1$ ($n-3$ unknowns) as follows:

First, we construct the graphs G_1 through G_{n-4} as described earlier. Second, we compute P(2), and apply the algorithm for computing δ & δ_1 through δ_{n-4}. The coefficient matrix is nonsingular; because $p_1 < p_2 < \ldots < p_{n-4}$, the resulting coefficient matrix is a variation of the Vandermonde matrix with nonzero determinant [20]. Hence, there exists a unique solution that can be found in polynomial time using Gaussian elimination, and back substitution. We assume the given probabilities are rational numbers which allow doing the computation on integers of length $\Theta(n)$. Once the set of coefficients $P(l)$, $2 \le l \le n-1$ are known, we can compute $Rel_2(G)$ using (3). □

Corollary 3: Computing δ is #P-hard.

Proof: The proof follows from *Theorem* 2, and the fact that 2REL has been shown in [1] to be #P-complete. Y

In the next section of the paper, we present two new algorithms for CSMD in arbitrary networks.

IV. Algorithms for Computing DSN Reliability and Sensor Message Delay

A. Problem Statement

Given a DSN graph $G = (V, E)$, an end user (sink) node s, a set of target sensor nodes $T \subset V$, and an operation probability p_v for every vertex $v \in V - s$, compute the probability that there exists an operational path between s, and an operational node $t \in T(Rel)$; and, the expected, and the maximum length of that path (δ and λ respectively).

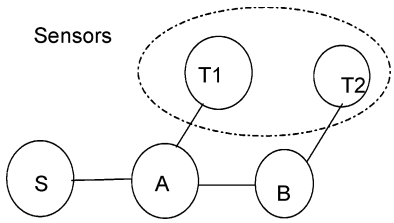

Fig. 5. DSN graph G.

First, we design an algorithm based on complete state enumeration which can be applied to a graph or a subgraph with a limited size.

B. Algorithm 1

A state of the network is defined as a subset $S \subseteq V$, where all nodes in S are operating, and all nodes in $V - S$ are failed. The probability of a state S is $P(S) = \prod_{v \in S} p_v \prod_{w \in V-S}(1-p_w)$. Let $\mathrm{n} = |\mathrm{V}|$. The network has 2^{n-1} possible states. If the state contains an (operating) s,t-path, for any operating $\mathrm{t} \in \mathrm{T}$, then it is called a *pathset*. Otherwise, u & T are disconnected, and the state is called a *cutset*. Each pathset contributes to exactly one of the path probabilities $P(l)$, $1 \le l \le n-1$. Note that, consistent with our assumption that the routing methods discover any operational path, and use one of the available shortest paths, more than one pathset may contribute to the same $P(l)$. All cutsets contribute to $P(\infty)$. Therefore we can compute Rel, and δ using the following algorithm:

Algorithm 1:
Begin {ENUMERATE}
1. for $l = 1$ to $n-1$ do $P(l) = 0$.
2. *Enumerate* all states of G and
for each state S
if S contains at least one operating $t \in T$
find the length of the shortest s, t-path
if one exists
let the length be l, $P(l) = P(l) + P(S)$
endfor
3. Substitute for $P(l)$, $1 \le l \le n-1$, *in* (2) and (3) to compute Rel and δ.
4. Compute λ = the maximum value of l for which $P(l) \neq 0$.
End {ENUMERATE} Y

Complexity: For each state, the algorithm uses breadth-first-search to find the length of the shortest u, t-path. The search stops when an operational node $t \in T$ is reached, if one exists. Therefore, the time complexity is $O(m2^{n-1})$, where n is the number of nodes, and m is the number of edges.

An Example: Consider the graph G in Fig. 5. G has 2^4 states. Table I illustrates only the possible operational states (pathsets).

TABLE I
ENUMERATION OF THE STATES OF G

State	probability	length of shortest path
S1	$P_A P_{T1} P_B P_{T2}$	2
S2	$P_A P_{T1} q_B P_{T2}$	2
S3	$P_A P_{T1} q_B q_{T2}$	2
S4	$P_A P_{T1} P_B q_{T2}$	2
S5	$P_A q_{T1} P_B P_{T2}$	3

Let

$$P_A = P_B = 0.9, \quad P_{T1} = P_{T2} = 0.5.$$
$$P(2) = P(S_1) + P(S_2) + P(S_3) + P(S_4) = 0.45$$
$$P(3) = P(S_5) = 0.9 \times 0.5 \times 0.9 \times 0.5 = 0.2025$$
$$Rel(G) = 0.45 + 0.2025 = 0.6525$$
$$\delta = \frac{(2 \times 0.45 + 3 \times 0.2025)}{(0.45 + 0.2025)} = 2.3103$$
$$\lambda = 3.$$

This example clearly shows the simplicity of the algorithm. Furthermore, it shows that computing λ & δ for the induced subgraphs formed by clustering can be possibly done in a reasonable time for clusters of limited size.

C. Algorithm 2

In this computational sequence, we avoid enumerating all network states as is described in *Algorithm 1*. Instead, we recursively generate shortest s, t-paths. New paths are generated recursively by considering the two states of each vertex on the current path: *failed* state, and *up* (*or operating*) state. This technique is based on the following *factoring theorem*, which is analogous to the factoring theorem used for computing $Rel_2(G)$ [1], [8]. Let $P(l,G)$ be the probability that the shortest (operating) s, t-path in G is of length l. Then, by considering the two states of any vertex v in $V - s$, we have the following factoring theorem:

Theorem 4: $P(l,G) = p_v P(l, G \bullet v) + q_v P(l, G - v)$, where $G \bullet v$ denotes the network G with node v operating (always up), and $G - v$ denotes the network with node v failed (or equivalently deleted).

Proof: The probability $P(l,G)$ is the sum of all probabilities of pathsets where there exists at least one operating target sensor node $t \in T$, there exists an operating s, t-path, and the length of the shortest operating s, t-path is l. The pathsets can be partitioned into two subsets P_u, and P_f, where v is operating (up) in P_u, and failed in P_f. Now P_u is the probability that the shortest s, t-path in G is of length l given that v is operating, and P_f is the probability that the shortest s,t-path in G is of length l with v failed. Therefore, $P_u = p_v P(l, G \bullet v)$, and $P_f = q_v P(l, G - v)$. □

Now, consider a shortest s,t-path $(s, v_1), (v_1, v_2), \ldots, (v_{l-1}, t)$. We have

$$P(l,G) = p_{v_1} P(l, G \bullet v_1) + q_{v_1} P(l, G - v_1)$$
$$P(l,G) = q_{v_1} P(l, G - v_1) + p_{v_1} \left(p_{v_2} P(l, G \bullet v_1 \bullet v_2) + q_{v_2} P(l, G \bullet v_1 - v_2)\right)$$

Similarly, by applying Theorem 3 $l-1$ times, we have

$$\begin{aligned} P(l,G) = {} & q_{v_1} P(l, G - v_1) + p_{v_1} q_{v_2} P(l, G \bullet v_1 - v_2) + \cdots \\ & + p_{v_1} p_{v_2} \cdots p_{v_{i-1}} q_i P(l, G \bullet v_1 \bullet v_2 \ldots \bullet v_{i-1} - v_i) + \cdots \\ & + p_{v_1} p_{v_2} \cdots p_{v_{l-2}} q_{l-1} P(l, G \bullet v_1 \bullet v_2 \ldots \bullet v_{l-2} - v_{l-1}) \\ & + p_{v_1} p_{v_2} \cdots p_{v_{l-1}} . p_t \end{aligned} \quad (7).$$

Therefore, given a shortest s, t-path $(s, v_1), (v_1, v_2), \ldots, (v_{l-1}, t)$, we can add $p_{v_1} p_{v_2} \cdots p_{v_{l-1}} p_t$ (*increment*) to $P(l,G)$, then recursively compute further terms in the above equation. Based upon the above formulation, the second algorithm has the following steps:

Algorithm 2:
Begin {FACTOR}
1. for $l = 1$ to $n-1$ do $P(l) = 0$.
2. *GeneratePath* $(G, 1)$
3. use (2) and (3) to compute Rel and δ.
4. Compute λ = the maximum l such that $P(l) \neq 0$.
End {FACTOR} Y

Where procedure *GeneratePath* is defined as follows:

procedure *GeneratePath*(G: *graph*, M: *multiplier*)
begin
1. Find a shortest s, t-path in G
If *not found Exit*
else Let this path be $(s, v_1), (v_1, v_2), \ldots, (v_{l-1}, v_l = t)$
2. Initialize an empty sequence of vertices U
Initialize $IncP = M$.
3. For $v = v_1$ *to* v_l do
If v is not *marked up* then
$IncP = IncP * p_v$
add v to U
endfor
4. $P(l) = P(l) + IncP$
5. Let the sequence U be $u_1, u_2, \ldots, u_k$
Call *GeneratePath*$(G - u_1, q_{u1} * M)$
if $k = 1$ *exit;*
mark u_1 *up*
Call *GeneratePath*$(G - u_2, p_{u1} * q_{u2} * M)$
mark u_2 *up*
Call *GeneratePath*$(G - u_3, p_{u1} * p_{u2} * q_{u3} * M)$
...
mark u_{k-1} *up*
Call *GeneratePath*$(G - u_k, p_{u1} * p_{u2} * \ldots p_{u_{k-1}} * q_{u_k} * M)$
end procedure {GeneratePath}. Y

Remarks on the Correctness: The correctness follows from the above formulation based upon factoring theorem for $P(l)$ (Theorem 4). Each recursive call corresponds to computing a

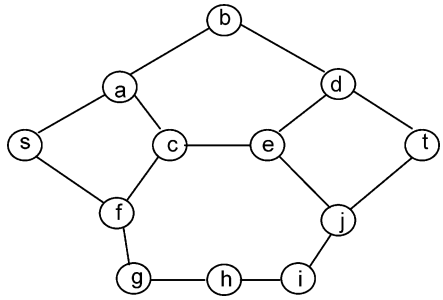

Fig. 6. An example graph for illustration of path generation using Algorithm 2.

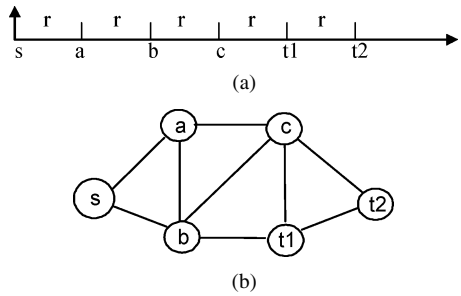

Fig. 7. (a) 6 nodes in line, $\mathrm{r} = 0.4\mathrm{R}$, (b) An interval graph G.

term in (7). A multiplier is used to reflect the states of vertices factored upon so far. The multiplier is multiplied by q_v for a deleted vertex, and by p_v for an operating (up) vertex. Step 3 determines the set of vertices on the new found s, t-path which are not already marked up so that a vertex can not be factored upon twice. Any shortest s,t-path found at step 1 must have at least one unmarked vertex. Otherwise, this path would have been found in an earlier call. But this is impossible because every recursive call differs from the originating call by deleting one vertex on the current shortest s, t-path.

Complexity Analysis of Algorithm 2: Let the number of shortest s, t-paths be n_p, and the number of s, t-cutsets be n_c. Algorithm 2 requires time $O(m\ n_c\ n_p)$. To see this, note that in step 1, the procedure call is terminated if no s, t-path exists. However, the number of ways an s, t-path is generated is exactly the number of ways the set of shorter s, t-paths fail. An upper bound on the latter number is the number of s, t-cutsets (n_c). Therefore, for most practical cases, the number of states generated (corresponding to procedure calls) by Algorithm 2 is expected to be substantially less than $2^{\mathrm{n}-1}$ (the number of network states). For example, consider the graph in Fig. 6. Algorithm 2 generates only 42 states compared to the 2^{11} $(= 2048)$ states which would be generated by Algorithm 1.

The path probabilities computed by Algorithm 2 are

$$\begin{aligned}
P(4) &= p_a p_b p_d p_t,\\
P(5) &= q_a p_f p_c p_e p_d p_t + q_a p_f p_c p_e q_d p_j p_t + p_a q_b p_c p_e p_d p_t\\
&\quad + p_a q_b p_c p_e q_d p_j p_t + p_a p_b q_d p_c p_e p_j p_t\\
P(6) &= q_a p_f q_c p_g p_h p_i p_j p_t + q_a p_f p_c q_e p_g p_h p_i p_j p_t\\
&\quad + p_a q_b q_c p_f p_g p_h p_i p_j p_t + p_a q_b p_c q_e p_f p_g p_h p_i p_j p_t\\
&\quad + p_a p_b q_d q_c p_f p_g p_h p_i p_j p_t + p_a p_b q_d p_c q_e p_f p_g p_h p_i p_j p_t.
\end{aligned}$$

Among the 42 states generated, only 12 states contain operational paths. Exactly 1 state contributes to P(4), 5 states contributes to P(5), and 6 states contributes to P(6). Note that 6 is the maximum number of hops in a shortest (minimal) s, t-path.

V. Efficient Algorithms for Restricted Classes

Although the problem of computing the reliability, and expected message delay for an arbitrary DSN graph has been shown in Section III to be #P-hard, polynomial time algorithms can still be developed for some restricted cases. In this section, we consider two such cases, namely, disjoint paths, and interval graphs. These cases provide examples of the general idea of exploiting the special architecture of the network topology to develop efficient algorithms (i.e. polynomial in the size of the network).

A. An Algorithm for Disjoint Paths

We consider the case where the sink node is connected to the target cluster through multiple disjoint paths. Based on the assumption that a message from sink (s) to a target sensor node (t) always takes the shortest available s,t-path, for any two paths P, and $P^{\backprime}$ of lengths l, and $l^{\backprime}$ respectively, P is used iff l is less than $l^{\backprime}$. As mentioned earlier, if more than one shortest path exist, the routing scheme chooses arbitrarily any of them.

Suppose that the set of shortest s, t-paths are $P_1, P_2, \ldots, P_k$ where $P_i \cap P_j = \varnothing$ for $i \neq j$, (i.e. vertex disjoint paths), and length of P_i is less than or equal the length of P_j for $i < j$. Note that P_i is the shortest s, t-path in $G - \{P_1 \cup \ldots \cup P_{i-1}\}$. In this case, the probability that path P_i is the shortest available operating s,t-path is the probability that all paths P_1 through P_{i-1} are failed, times the probability that P_i is operating $(i > 1)$.

The probability that a given path P_i is operating is the product of the operational probabilities of its vertices $= \prod_{v \in P_i} p_v$.

The probability that all paths P_1 through P_{i-1} are failed is $\prod_{j=1}^{j=i-1}(1 - \prod_{v \in P_j} p_v)$.

Therefore a simple algorithm can be developed where the set of shortest paths are generated in order of their lengths. The probability of each path is then computed, and added to the corresponding $P(l)$ in (2). Generating each path requires $\Theta(m)$ time. An upper bound on the number of disjoint paths is $n - 2$. Hence the algorithm requires $O(nm)$ time.

B. Interval Graphs

Consider the case where network nodes are deployed along a line, or a thin rectangular area with width $\mathrm{w} \lll \mathrm{R}$ (the radius of the coverage area). In this case, the network can be represented by an interval graph. A graph G is an *interval* graph iff G is the intersection graph of intervals on the real line [11]. The importance of interval graphs arises from the remark that they represent special cases of wireless topologies where the range of transmissions (R) can be represented by intersection of intervals on the real line (X-axis). Let the X-coordinate of a vertex v be X[v]. Then, for each vertex v, we associate the open interval $(\mathrm{X[v]} - \mathrm{R}/2; \mathrm{X[v]} + \mathrm{R}/2)$ on the X-axis. The interval is centered around v, and has length R. In the corresponding interval graph, we add an edge between two vertices I, and j iff their corresponding intervals intersects. Clearly two intervals $\mathrm{I_v}$, and $\mathrm{I_w}$ corresponding to nodes v, and w intersect iff $|\mathrm{X[v]} - \mathrm{X[w]}| < \mathrm{R}$. Therefore, an edge (v, w) exists iff v & w are in the range of each other. Consider, for example, the set of nodes $\mathrm{V} = \{\mathrm{s, a, b, c, t1, t2}\}$ laid along a line (X-axis) with $\mathrm{equal\,separation} = \mathrm{r}$ (see Fig. 7(a)). Let the radius of transmission coverage be $\mathrm{R} = 2.5 \times \mathrm{r}$. The corresponding interval graph is shown in Fig. 7(b).

1) Definitions: A vertex v is called *simplicial* if the set of vertices adjacent to v induces a complete subgraph (clique). Every interval graph has at least two simplicial vertices. An ordering $\sigma = (v_1, v_2, \ldots, v_n)$ of vertices of a graph G is called a *perfect elimination sequence* if each vertex v_i is a simplicial vertex of the induced graph $G - \{v_1, v_2, \ldots, v_{i-1}\}$. Every interval graph has a perfect elimination sequence which can be computed in linear time together with the set of maximal cliques of G [11], [27]. A property of any interval graph G is that the maximal cliques of G can be linearly ordered such that the maximal cliques containing any one vertex occur consecutively [11], [14]. For the case of a DSN, σ starts with the source node s ($v_1 = s$), and ends with the set of target sensor nodes T. This can be easily seen if we remark that any vertex that appears in σ before s, or after T, is irrelevant. The perfect elimination sequence (σ), and the set of maximal cliques (C) for the graph G in Fig. 7(b) are $\sigma = \{\mathrm{s, a, b, c, t1, t2}\}$, $\mathrm{C} = \{\{\mathrm{s, a, b}\}, \{\mathrm{a, b, c}\}, \{\mathrm{b, c, t1}\}, \{\mathrm{c, t1, t2}\}\}$. The following algorithm employs a dynamic programming technique based on the perfect elimination sequence, and the set of maximal cliques.

2) Algorithm:

Input: an interval graph $G = (V, E, P)$, a sink node s, a target set T, $V = \{v_1, v_2, \ldots, v_n\}$, $v_1 = s$, $v_n = t \in T$. $P = \{p_1, p_2, \ldots, p_n\}$, where p_i = operational probability of v_i, $p_1 = 1$.

Output: $Rel(G)$, $\lambda(G)$, $\delta(G)$, the DSN reliability, the maximum and the expected delay of G.

Notation: Let $right(i)$, and $left(i)$ be the two end points of the interval corresponding to v_i. A property of the sequence σ is that for any two vertices i, and j in σ, $right(i) \leq right(j) \Leftrightarrow i \leq j$. Let $P(i, l, k)$ be the probability that the shortest path from s to v_i has length l hops given that only vertices up to (and including) v_k are allowed. We denote the minimum shortest path length from s to v_i by $Min(i)$, and the maximum shortest path length by $Max(i)$.

Steps:

0. (pre-processing step). Find σ, and the corresponding set of maximal cliques $C_1, \ldots, C_m$, $2 \leq m \leq n-1$ ($m = 1$ for a complete graph). Re-label the set of vertices in the same order they appear in σ. The following steps assume that the set of vertices V are labeled in exactly the same order they appear in the perfect elimination sequence σ.
Mark all vertices in $V - \{s\}$ as '*new*'.

1. For every vertex v_i in $C_1 - s$: $P(i, 1, 1) = 1$, mark v_i as '*old*', $Min(i) = Max(i) = 1$.

2. For $c = 2$ *to* m, do

2.1. Let the set of vertices in C_c marked '*new*' be N.

2.2. For every vertex x in N

2.2.1 mark x as '*old*'.
Let the set of vertices (in C_c) marked '*old*' that appear before x in σ be O.
Let $O = \{O_1, O_2, \ldots, O_k\}$, where k is the cardinality of O.
Let $Min(O)$ be the minimum of $Min(v)$ over all vertices v in O, and $Max(O)$ be the maximum of $Max(v)$.

2.2.2. For $j = 1$ to k

2.2.2.1. Order the subset (of O) $O_j = \{O_1, O_2, \ldots, O_j\}$ in ascending order of $left(v)$, and let the sorted set be $\pi = \{\pi 1, \pi 2, \ldots, \pi j\}$

2.2.2.2. For $l = Min(O)$ to $Max(O)$ do

$$P(x, l+1, \pi_j) = p_{\pi 1} P(\pi_1, l, y) + q_{\pi 1} p_{\pi 2} P(\pi_2, l, y) + \ldots + q_{\pi 1} q_{\pi 2} \ldots p_{\pi j} P(\pi_j, l, y),$$

where y is the highest order vertex that precedes C_c in σ (i.e. precedes all vertices in C_c).

3. Let the set of target sensors $\mathrm{T} = \{t1, \ldots tm\}$ be ordered as they appear in σ; let y be the highest order vertex preceding $t1$ in σ; and let $Lmin$, and $Lmax$ be the minimum path length, and the maximum path length respectively, to any target sensor in T (with nonzero probabilities).
For $L = Lmin$ *to* $Lmax$

$$P(L) = p_{t1} P(t1, L, y) + q_{t1} p_{t2} P(t2, L, y) + \ldots + q_{t1} \ldots q_{tm-1} p_{tm} P(tm, L, y)$$

4. Compute $Rel = \sum_{L=L\min}^{L\max} P(L)$

5. Compute $\delta = \sum_{L=L\min}^{L\max} LP(L) / \sum_{L=L\min}^{L\max} P(L)$, $\lambda = Lmax$.

End.

Correctness and Complexity: The correctness of the above algorithm can be verified using the following remarks.

1. For a given maximal clique, the path probabilities of a '*new*' vertex x depend only on path probabilities of vertices in the *old* set (O) that precede it in σ. This is necessary because, for any vertex y that follows x in σ, there exists no shortest path of the form $(s, \ldots, y, x, \ldots, t)$ because $left(x)$ is less than or equal left(y).
2. The path probabilities expressed by the terms summed up in the right hand side in *Step* 2.2.2.2. do not include any vertex in the O set.
3. Ordering the vertices by their left end (π set in Step 2.2.2.1) implies that the probability terms are exhaus-

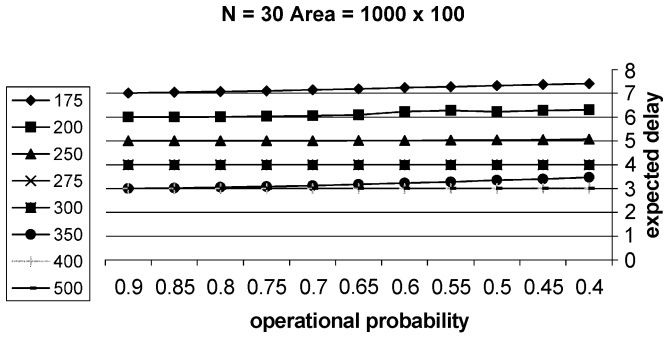

Fig. 8. Expected delay vs. probability for different sensor ranges.

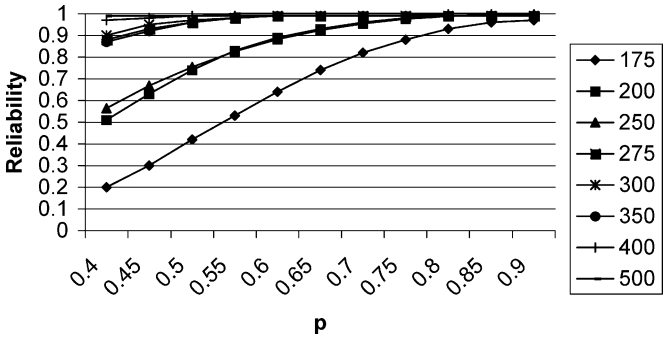

Fig. 9. Reliability vs. p for different sensor ranges.

tive & disjoint for all paths to x through the O set. The same remark applies to step 3.

An easy upper bound on the complexity of the algorithm is $O(n^3)$.

VI. Numerical Results

In this section, we present sample numerical results obtained by applying Algorithm 2 (based on factoring) to some examples. The main reason for including such numerical results is to show that, although the algorithm has an exponential worst-case complexity, it still can yield results for networks with a relatively large number of nodes in a few minutes time. As expected, an implementation of Algorithm 1 did not yield results in a reasonable time for a DSN with more than 25 nodes. Recall that Algorithm 1 is based on state enumeration of the network. For a network of 25 nodes, the number of states enumerated is 2^{24}. The numerical results have been obtained using a C program on a PC, and the maximum experiment time was about 3 minutes (for a DSN with 40 nodes distributed randomly over an area of 2000 by 1000 meters, and $\text{transmitter/receiver range} = 200\ \text{meters}$). In Fig. 8, we display the computed expected delay of an operational network (expected number of hops), between the furthest pair of nodes (s & t), for 30 nodes distributed randomly over an area of 1000 by 100 meters. The delay is computed for different operational probabilities, and different coverage ranges (175–500 meters). We chose to compute the delay between the furthest pair of source-destination nodes because it can be used as an estimation of the expected diameter of the network. Fig. 9 shows the probability of having an operational path for the same network of Fig. 8. For example, for a sensor range

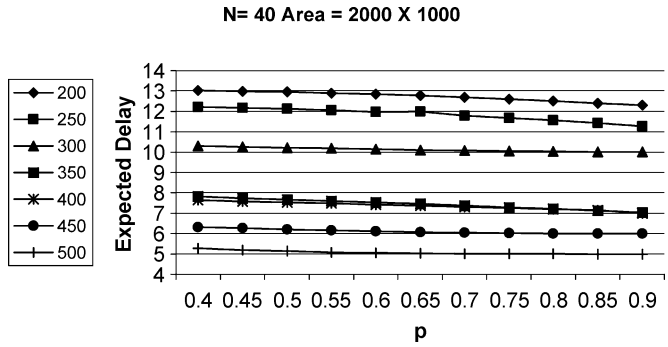

Fig. 10. Expected delay vs. probability for different sensor ranges.

of 175 meters, and $\mathrm{p} = 0.75$ for the operational probability of the sensor node, from Fig. 9, the probability of having an operational path between s & t is 80%. From Fig. 8, the expected path length (delay) will be about 7 hops. As expected, as the range of the sensor nodes decreases, the average delay increases, and the reliability decreases.

It has been noticed from different experiments run on DSN with random distribution of nodes that the effect of the operational probability of a node is not as significant as the effect of the range. This is illustrated in the example DSN in Fig. 10. This may suggest that minimizing the delay requires using sensors with larger ranges. However, the sensor range is proportional to at least the square of the sensor power [26], which requires much more expensive sensor nodes.

VII. Conclusion

In a cooperative wireless sensor network, clusters of sensors are deployed near the target phenomenon to provide information to sink nodes or end user stations. The reliable monitoring of events at the sink is based on the collective information provided by the cluster, and not on any individual sensor node.

In this paper, we investigate the problem of computing probabilistic measures for the reliability of multi-hop wireless distributed sensor networks, and the expected & maximum message delay in an operational network. We assume that nodes are subject to random failures with known failure probabilities. This problem is important in the context of the topology analysis of DSN, where it is required to relay a large number of messages within a given time interval to far end-points. We show that both problems are computationally intractable for arbitrary networks, in particular #P-hard. We present two algorithms for arbitrary networks. Naturally, these algorithms are exponential in the size of the network. However, it can be applied (as demonstrated in our numerical examples) to graphs of limited size; in our tests, we applied a second algorithm to networks of up to 40 nodes. Note that the *clustered* organization of a DSN implies that an intra-cluster delay will be affected only by the cluster size, and similarly, the delay between backbone nodes will be affected only by the backbone size. The second algorithm requires polynomial time in the number of pathsets & cutsets, which may be substantially smaller than the number of network states. Also we present efficient algorithms for two restricted cases, namely disjoint paths, and interval graphs.

In this paper, we assume bidirectional links, and shortest-path routing algorithms. For future work, networks with asymmetric links, or different transmitter/receiver ranges (which requires directed graph models), and other routing schemes may be investigated. Also, efficient algorithms for computing the expected delay, and other related performance measures may be sought for other restricted topologies of DSN.

ACKNOWLEDGMENT

The authors would like to thank Prof. Mansour Al-Zanaidi of Kuwait University for his support on this work. The authors also would like to thank the anonymous referees for their helpful comments.

REFERENCES

[1] H. AboElFotoh and C. J. Colbourn, "Computing 2-terminal reliability for radio-broadcast networks," *IEEE Trans. Reliab.*, vol. 38, no. 5, pp. 538–555, 1989.

[2] J. Agre and L. Clare, "An integrated architecture for cooperative sensing networks," *IEEE Comput.*, vol. 5, pp. 106–108, 2000.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[4] B. Badrinath and M. Srivastava, "Smart spaces and environments," *IEEE Personal Commun.*, vol. 7, no. 5, Oct. 2000.

[5] R. R. Brooks and S. S. Iyengar, *Multi-Sensor Fusion—Fundamentals and Applications with Software*, New Jersey: Prentice Hall PTR, 2000.

[6] N. Bulusu, J. Heidmann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices," *IEEE Personal Commun.*, vol. 7, no. 5, pp. 28–34, Oct. 2000.

[7] A. Cerpa and D. Estrin, "ASCENT: adaptive self-configuring sensor networks topologies," *IEEE Trans. Mobile Comput.*, vol. 3, no. 3, pp. 272–285, 2004.

[8] C. J. Colbourn, *The Combinatorics of Network Reliability*: Oxford University Press, 1987.

[9] S. De, C. Qiao, and H. Wu, "Meshed multipath routing with selective forwarding: an efficient strategy in wireless sensor networks," *Computer Networks*, vol. 43, pp. 481–497, 2003.

[10] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Scalable coordination in sensor networks," in *Proc. MobiCom 99*, pp. 263–270.

[11] D. R. Fulkerson and O. A. Gross, "Incidence matrices and interval graphs," *Pacific J. Math.*, vol. 15, pp. 835–855, 1965.

[12] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly resilient, energy efficient multipath routing in wireless sensor networks," *Mobile Computing and Communication Review (MC2R)*, vol. 1, no. 2, 2002.

[13] M. R. Garey and D. S. S. Johnson, *Computers and Intractability—A Guide to the Theory of NP-Completeness*. New York: Freeman and Company, 1979.

[14] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*. London: Academic Press, 1980.

[15] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocols for wireless microsensor networks," in *Proc. 33rd Hawaiian Int. Conf. Systems Science*, vol. 8, Jan. 2000.

[16] S. S. Iyengar, D. N. Jayasimha, and D. Nadig, "A versatile architecture for the distributed sensor integration problem," *IEEE Trans. Comput.*, vol. 43, no. 2, pp. 175–185, 1994.

[17] D. N. Jayasimha, S. S. Iyengar, and R. L. Kashyap, "Information integration and synchronization in distributed sensor networks," *IEEE Trans. Syst., Man Cybern.*, vol. 21, no. 5, pp. 1032–1043, 1991.

[18] R. Kannan, S. Sarangi, S. S. Iyengar, and L. Ray, "Sensor-centric quality of routing in sensor networks," *IEEE INFOCOM*, vol. 22, no. 1, pp. 692–701, 2003.

[19] P. Krishna, N. Vaidya, M. Chatterjee, and D. Pradhan, "A cluster-based approach for routing in dynamic networks," *ACM SIGCOMM Computer Communications Review*, pp. 372–378, 1997.

[20] P. Lancaster and M. Tismenetsky, *The Theory of Matrices*, 2nd ed: Academic Press, Inc., 1985.

[21] C. G. Lekkerkerker and J. C. Bolan, "Representation of a finite graph by a set of intervals on the real line," *Fundamenta Mathematica*, vol. 51, pp. 45–64, 1962.

[22] W. W. Manges, "Wireless sensor network topologies," *Sensors Mag.*, vol. 17, no. 5, May 2000.

[23] S. Park and R. Sivakumar, "Sink-to-sensors reliability in sensor networks," in *MobiHoc'03*, Jun. 1–3, 2003.

[24] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, no. 5, pp. 551–558, May 2000.

[25] H. Qi, S. S. Iyengar, and K. Chakrabarty, "Multiresolution data integration using mobile agents in distributed sensor networks," *IEEE Trans. Syst. Man Cybern.*, vol. 31, no. 3, pp. 383–401, 2001.

[26] J. M. Rabaey, M. J. Ammer, J. L. da Silva, and D. P. Roundy, "PicoRadio supports ad hoc ultra-low power wireless networking," *IEEE Comput.*, vol. 7, pp. 42–48, 2000.

[27] D. J. Rose and R. E. Tarjan, "Algorithmic aspects of vertex elimination," in *Proc. 7th ACM Symp. Theory of Computing*, 1975, pp. 245–254.

[28] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz, "ESRT: event-to-sink reliable transport in wireless sensor networks," in *Proc. MobiHoc'03*, Jun. 1–3, 2003, pp. 177–188.

[29] C. Schurgers and M. B. Srivastava, "Energy efficient routing in wireless sensor networks," in *Military Communications Conference, MILCOM 2001 Communications for Network-Centric Operations: Creating the Information Force*, vol. 1, pp. 357–361.

[30] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE Personal Commun.*, pp. 17–27, October 2000.

[31] R. E. Van Dyke and L. E. Miller, "Distributed sensor processing over an ad-hoc wireless network: simulation framework and performance criteria," in *Proc. IEEE Milcom, McLean*, Oct. 2001.

[32] J. Warrior, "Smart sensor networks of the future," *Sensors Mag.*, Mar. 1997.

[33] W. Ye and J. Heidemann, "Medium Access Control in Wireless Sensor Networks,", USC/ISI Tech. Rep ISI-TR-580, Oct. 2003.

Hosam M. F. AboElFotoh received the B.Sc. degree in Electrical Engineering, Computer and Automatic Control Section, from Ain-Shams University in Cairo in 1977, the M.Math and Ph.D. degrees in Computer Science from the University of Waterloo, Ontario, in 1984 and 1989, respectively. He is currently an Associate Professor at the Department of Mathematics and Computer Science at Kuwait University. His interests include computer networks, network reliability, distributed systems and databases, and design of algorithms.

S. S. Iyengar is the Chairman and Roy Paul Daniels Chaired Professor of Computer Science at Louisiana State University and is also Satish Dhawan Chaired Professor at Indian Institute of Science. He has been involved with research in high performance algorithms, data structures, sensor fusion, data mining, and intelligent systems. He has directed over 34 Ph.D. students, many of whom are faculty at major universities worldwide or scientists or engineers at national labs/industry around the world. His publications include 13 books (authored or co-authored, edited; Prentice Hall, CRC Press, IEEE Computer Society Press, John Wiley & Sons, etc.) and over 300 research papers in refereed journals and conferences in areas of high-performance parallel and distributed algorithms and data structures for image processing and pattern recognition, and distributed data mining algorithms for biological databases. His books have been used by researchers at Purdue University of Southern California, University of New Mexico, etc. at various times. His forthcoming book on Distributed sensor networks will be released in October, 2004. He was a visiting professor at the Jet Propulsion Laboratory-Cal. Tech, Oak Ridge National Laboratory, the Indian Institute of Science, and at the University of Paris and other places. He has been on the prestigious National Institute of Health NLM Review Committee, in the area of Medical Informatics for 4 years. Dr. Iyengar is a Fellow of the Association of Computing Machinery (ACM), Fellow of the American Association of Advancement of Science (AAAS), Fellow of the Institute of Electrical and Electronics Engineering (IEEE), and has served on numerous panels for the US National Science Foundation, National Research Council (Reviewed Proposals), and the Defense Advanced Research Projects Agency, Member of the European Academy of Sciences, etc. He received the Prestigious Distinguished Alumnus Award from Indian Institute of Science, Bangalore in 2003. He has been the Program Chairman for many national/international conferences. He has given over 60 plenary talks and keynote lectures at numerous national and international conferences.

Krishnendu Chakrabarty received the B.Tech. degree from the Indian Institute of Technology, Kharagpur, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1992 and 1995, respectively, all in Computer Science and Engineering. He is now Associate Professor of Electrical and Computer Engineering at Duke University. During 2000-2002, he was also a Mercator Visiting Professor at University of Potsdam in Germany.

Dr. Chakrabarty is a recipient of the National Science Foundation Early Faculty (CAREER) award and the Office of Naval Research Young Investigator award. His current research projects include design and testing of system-on-chip integrated circuits; embedded real-time-systems; distributed sensor networks; modeling, simulation and optimization of microfluidic systems; and microfluidics based chip cooling. Dr. Chakrabarty is a co-author of two books: (CRC Press, 2002) and (Kluwer, 2002), and the editor of (System–on–a–Chip) Testing for Plug and Play Test Automation (Kluwer 2002). He has published over 160 papers in journals and refereed conference proceedings, and he holds a U.S. patent in built-in-self-test. He is a recipient of a best paper award at the 2001 Design, Automation and Test in Europe (DATE) Conference. He is also a recipient of the Humboldt Research Fellowship, awarded by the Alexander von Humboldt Foundation, Germany. Dr. Chakrabarty is an Associate Editor of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, an Editor of the Journal of Electronic Testing: Theory and Applications (JETTA), and a member of the editorial board for Sensor Letters and Journal of Embedded Computing. He has also served as an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: ANALOG AND DIGITAL SIGNAL PROCESSING. He is a member of ACM and ACM SIGDA, and a member of Sigma Xi. He serves as Vice Chair of Technical Activities in IEEE's Test Technology Technical Council, and is a member of the program committees of several IEEE/ACM conferences and workshops.