Scalable and Reliable Data Delivery in Mobile Ad Hoc Sensor Networks

Bin Yu, Paul Scerri, and Katia Sycara
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{byu, pscerri, katia}@cs.cmu.edu

Yang Xu, Michael Lewis
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA 15260, USA
{xuy,ml}@sis.pitt.edu

ABSTRACT

This paper studies scalable data delivery algorithms in mobile ad hoc sensor networks with node and link failures. Many algorithms have been developed for data delivery and fusion in static microsensor networks, but most of them are not appropriate for mobile sensor networks due to their heavy traffic and long latency. In this paper we propose an efficient and robust data delivery algorithm for distributed data fusion in mobile ad hoc sensor networks, where each node controls its data flows and learns routing decisions solely based on their local knowledge. We analyze the localized algorithm in a formal model and validate our model using simulations. The experiments indicate that controlled data delivery processes significantly increase the probability of relevant data being fused in the network even with limited local knowledge of each node and relatively small hops of data delivery.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Distributed Artificial Intelligence*

General Terms

Algorithms, Reliability, Experimentation,

Keywords

sensor data fusion, data delivery, reinforcement learning

1. INTRODUCTION

The mobile ad hoc sensor networks of the near future are envisioned to consist of hundreds of UAVs (unmanned aerial vehicles) or robots. These networked mobile sensors play strong roles in military and civilian operations, e.g., battlefield surveillance, disaster search and rescue [9, 15, 13]. One phenomenon in mobile sensor networks is that the raw data from each sensor usually has low confidence. The raw data cannot be used directly for team coordination and has to be fused with other relevant data [11]. Various statistical inference techniques have been studied for distributed data fusion in mobile sensor networks, but these approaches do not have any control on the data flows [12]. Uncontrolled data flows may cause large amount of conflicting plans and have severe effects on the performance of the whole network when mobile sensors work together to conduct high-level tasks. Thus, how to optimize communication strategies for team coordination is an important problem for mobile ad hoc sensor networks [18]

Many protocols have been developed for data delivery and fusion in microsensor networks with small and battery-powered "motes", where a node, called a sink, attempts to gather data from a number of data sources [22]. For example, in directed diffusion, the sink floods a query toward the sensor nodes in a specific area [7]. The source nodes within the specific area then send data back to the sink along the best path. In other words, a source node will not send any data back to the sink until it receives the query. For this reason, these routing protocols are called *reactive protocols*.

Reactive protocols are mainly designed for microsensor networks, where the sensors are stationary after deployment. They are not appropriate for mobile sensor networks for the following two reasons:

- Heavy traffic: Since locations of collected data are not correlated with existing positions of mobile sensors, effective routing of either query or data is not feasible. Sinks have to flood the queries to the whole network when they need the data each time for a specific area. This could overload one or more sinks and cause a lot of traffic in the network.

- Long latency: Sinks usually do not know when source nodes will have the data in mobile sensor networks. If the source nodes follow reactive protocols, they have to wait for the queries from sinks to send the data. This causes long latency and cannot meet the requirements of time-critical missions in many defense and civilian applications for mobile sensor networks [2].

In this paper we present a scalable and reliable approach to data delivery in mobile ad hoc sensor networks with failures. While failures could happen at either team coordination level or data communication level, our approach focuses on communication failures for sensor data delivery among

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

mobile sensors [8]. We assume these mobile sensors such as UAVs are "blind" in some sense and they cannot monitor the activities of other sensors directly.

In our approach there is no querying process and each node proactively forwards the data to one of its neighbors. The constraint here is that nodes have no global knowledge of network topology. Each node has to intelligently deliver data solely based on the knowledge about itself and its neighbors. Moreover, the data delivery processes need to be survivable to failures. The challenge is, with various decisions made by individual sensors, how to minimize the communication overhead and maximize the probability of relevant data being fused in the network.

We propose an efficient and failure-resistant localized algorithm over mobile ad hoc networks, where each node learns routing decisions from its history of data delivery via *Q-learning*. We study the process of neighbors selection for each sensor node and the emergence of a spanning tree in the network for reliable data delivery. Moreover, we analyze the probability of relevant data being fused in a random network in a formal model and validate our model using simulations. The experiments show that controlled data flows significantly increase the probability of relevant data being fused, where the probability is improved by two to five times for the same hop of data propagation.

The rest of this paper is organized as follows. Section 2 summarizes the relevant literature. Section 3 briefly describes the background of mobile ad hoc sensor networks. Section 4 investigates the design of protocols and localized algorithms for data delivery and fusion. Section 5 provides a formal analysis of the data delivery algorithm in mobile sensor networks. Section 6 presents some experimental results for the algorithms. Section 7 discusses the main themes and some directions for future research.

2. RELATED WORK

Mobile sensor networks have been studied by Howard et al. [6], but their work focuses on the coverage problem of a multi-robot system, and does not consider the effective data delivery for fusion. Q-learning and ants algorithms have been studied by [1, 19] for network routing problem. Routing problem is different from ours since the sensor data does not have explicit destination during the data delivery processes. Moreover, we need not consider alternative paths for surplus sensor data. The data propagation paths are deterministic in our approach, while paths in ants algorithms are probabilistic.

Multisensor data fusion has been studied for a long time, where both the data fusion and control algorithms are centralized [4]. We will not discuss the literature on multisensor data fusion since they fall outside of the scope of this paper.

Distributed data fusion in sensor networks has been focused on statistical inference techniques. For example, Niu et al. investigate an optimum local fusion detection threshold for binary hypothesis testing by local sensors [14]; Rosencrantz et al. present a Bayesian technique for decentralized state estimation using particle filters [16]; and Durrant-Whyte et al. consider distributed data fusion as a state estimation problem using information filter (a variant of Kalman filter) [11, 12]. These approaches focus on detection and tracking performance of distributed sensor networks. They do not have any control on the data flows in the network. In this paper we propose an efficient data delivery algorithm for distributed data fusion. Moreover, we study the tradeoff of the efficiency and fusion probability in a network with node and link failures.

3. BACKGROUND

This section provides some background on mobile ad hoc sensor networks.

We assume mobile sensors communicate on a point-to-point basis [17]. A network of these mobile sensors is modeled as a connected undirected graph $G = (V, E)$, where $V = \{a, b, \ldots\}$ is a set of sensor nodes and E consists of edges between any two nodes a and b that can communicate directly. $N(a)$ is the set of a's neighbors and $b \in N(a)$ is any neighbor of node a. Note that neighbors in mobile sensor networks need not be in the proximity and they are neighbors only in the logical sense.

There are two types of communication failures in a mobile ad hoc sensor network: *node failure* (the loss of sensor nodes) and *link failure* (unreliable communication channels). Link failure is temporary and is modelled as a fixed probability for any two nodes in our scenario. Node failure is permanent and cannot be recovered once the node is dead. Following [5], we model the reliability $R_a(t)$ of a node a as a Poisson distribution

$$R_a(t) = e^{-\lambda_a t} \tag{1}$$

where λ_a is the failure rate of the sensor node and t is the time period. $R_a(t)$ captures the probability of not having a failure within the time interval $(0, t)$. $1 - R_a(t)$ describes the probability of a node failure. The higher the failure rate λ_i, the lower the reliability of a sensor node.

Specifically, we consider a UAVs network for target detection, tracking, and classification in the battlefield. We assume there are multiple stationary or moving targets $T_1, T_2, \ldots, T_M$ on the ground. A target T_K may be detected by multiple UAVs at the same time, but each UAV cannot initialize its plans based on its own raw sensor data about the target, e.g., engagement of its missile with a target. The reason is that the raw sensor data from a UAV is uncertain and noisy. Sometimes, a UAV with SAR (Synthetic Aperture Radar) may even confuse friendly targets ($T80$ tanks) with enemy targets ($M1$ tanks). Hence, the low quality sensor data cannot be used directly for high-level plans and has to be delivered to other nodes for fusion in the network [4].

If sensor a detects a target T_K, it will generate an event e_i about target T_K. An event e_i about T_K can be denoted as a tuple $\langle sender, identity, location, TTL, pedigree \rangle$, where

- *sender* is the ID of the sensor that detects the target.
- *identity* is the decision about the target from the sensor. For example, the target could be classified as $\langle T80, 0.4 \rangle$, where $T80$ is the possible vehicle type and 0.4 is the confidence level.
- *location* is the location of the target T_K being detected.
- *TTL (time-to-live)* is the maximal number of hops allowed for the event propagation in the network.
- *pedigree* is the list of nodes event e_i has been visited, denoted as L. Note that pedigree is used to avoid cycles during event propagation.

The processes of sensor data delivery can be described as follows. When sensor a receives or generates an event e_i about T_K, it first tries to fuse e_i with its previous events about target T_K using statistical inference techniques. [1] If the confidence of fused events meets a threshold, sensor node a will stop the propagation of event e_i and become a sink node for relevant events of T_K. Otherwise, sensor a will choose one of its neighbors to continue the delivery unless its TTL is zero. Note that, sensor a will create an information token about target T_K when it becomes a sink for target T_K. Information tokens are then used for high-level team coordination [18].

4. ALGORITHMS

Obviously, an event does not need to visit all sensor nodes to meet with other relevant events. However, visiting too few sensors can lead to low probability of relevant events being fused. Here two events e_i and e_j are relevant if and only if they are referring to the same target, denoted as $rel(e_i, e_j) = true$. The key is how to minimize communication overload and maximize the probability of relevant data being fused. In this section, we discuss the efficiency and robustness of three algorithms: *random walks*, *path reinforcement* algorithm and *path learning* algorithm.

Random walks are a simple data delivery technique in mobile sensor networks. In random walks, sensor node a with event e_i first decides if it will stop the propagation of e_i. If not, sensor node a just forwards the event e_i to one randomly chosen neighbor $b \in N(a)$. Once the neighbor receives the data, it repeats the same process until the event is successfully fused with other relevant events or the TTL of the event reaches zero.

Obviously, random walks are not efficient for data delivery, since they do not consider the relationships of events and just pass events randomly in the network. Hence, we design other two algorithms — path reinforcement algorithm and path learning algorithm.

4.1 Path Reinforcement Algorithm

The intuition behind path reinforcement algorithm is that relevant events are likely to be fused earlier if they will follow the same path after they meet. Specifically, if sensor node a has sent an event e_i to one of its neighbors b before, it will send b any events e_j if $rel(e_i, e_j) = true$.

Figure 1 shows an example of data flows in the network using path reinforcement algorithm. The solid lines correspond to directed communication channels between sensor nodes. The arrows in dashed lines represent information flows of relevant events e_i, e_j, and e_k. Event e_i and e_j meet at node b and then they follow the same path (event e_j reinforces the path $\langle b, c\rangle$). As shown in Figure 1, the three events are fused at node c.

From the figure it is easy to understand why path reinforcement algorithm is more efficient than random walks for data delivery. For example, if event e_i and e_j meet at node b and they continue to walk randomly in the network, there will be a very small chance that they can meet again and be fused with event e_k at node c.

[1]Commonly used frameworks for processing uncertain sensor data fusion are Bayesian inference method and Dempster-Shafer theory, but the details are outside of the scope of this paper. Readers may refer to [3] for more information.

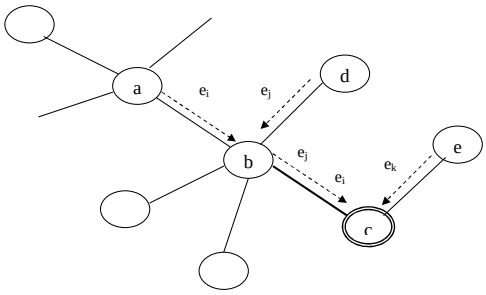

Figure 1: An example of data flows in the network using path reinforcement algorithm, where e_i, e_j, and e_k are three relevant events.

4.2 Path Learning Algorithm

Path reinforcement algorithm is quite efficient when there is no failure in the network. One phenomenon in path reinforcement algorithm is that a sensor never checks if its neighbor has successfully received the data. This will cause severe data loss problem and decrease the probability of relevant data being fused when node or link failures happen.

To remedy to the problem we can allow each node to retransmit the data to another neighbor when a failure is detected. Path reinforcement algorithm with retransmission is robust, but it is not efficient when failures happen. The reason is that sensor a may choose a failed neighbor again when a selects its neighbor randomly. Hence, the key is how to model the reliability of each neighbor, so we can reduce the number of retransmission. Here we describe another algorithm, called path learning algorithm, where each node keeps track of statistics on which neighbor leading to successful data delivery.

Formally, the probability of successfully passing events to a neighbor $b \in N(a)$ at time t is defined as $Q^t(a, b)$ and $0 < Q^t(a, b) < 1$. For any neighbor $b \in N(a)$, $Q^t(a, b)$ is initialized as 0.5 when $t = 0$. Given an event e_i, node a will send e_i to a neighbor $b \in N(a)$ with maximal Q value. Node a will randomly choose a neighbor for e_i if there are any two neighbors b, $c \in N(a)$ and both have maximal Q value at time t, e.g., $Q^t(a, b) = Q^t(a, c)$.

After node a sends out an event e_i to b, a waits for a feedback from b in a period of $\mathcal{T}$. The feedback from b can be defined as

$$feedback = \begin{cases} 1 & \text{if } b \text{ is a sink node for } e_i \\ 0 & \text{if no feedback} \\ max_{z \in N(b)} Q^{t-1}(b, z) & \text{otherwise} \end{cases}$$

where $Q^{t-1}(b, z)$ is the feedback from b and z has the maximal probability from the point of b at time $t - 1$. Here a sink node means b successfully fuses e_i with other relevant events.

The probability to deliver event e_i is updated as

$$Q^t(a, b) = (1 - \alpha)Q^{t-1}(a, b) + \alpha * feedback \qquad (2)$$

where α is the learning rate and $\alpha = 0.5$ in this paper if not specified otherwise. In the case of no response, node a will decrease the value of $Q^t(a, b)$ and choose another neighbor

Algorithms	Efficiency	Robustness
Random walks	bad	bad
Path reinforcement algorithm without retransmission	good	bad
Path reinforcement algorithm with retransmission	average	good
Path learning algorithm	above average	good

Table 1: The comparison of four data delivery algorithms.

to resend the event. The adjusted possibility reduces the chance of choosing b in the future.

Algorithm 1 presents the path learning algorithm, where e_i is an event for target T_K from either node a or other nodes and $\Theta_a(T_K)$ is the set of events about T_K that node a has. Other parameters include: L the pedigree of event e_i and $N(a)$ the set of neighbors of sensor a. The algorithm has two basic parts: (1) deciding if node a will stop propagating event e_i (line 1-3); (2) choosing a reliable neighbor $z \in N(a)$ to deliver e_i and z is not in e_i's pedigree L (line 5-18). Finally, sensor a updates its Q value about neighbor z if $feedback \neq 0$ (line 23).

Algorithm 1 The path learning algorithm

```
1: Θ_a(T_K) = Θ_a(T_K) ∪ e_i
2: if Θ_a(T_K) has high confidence then
3:    Return success
4: else
5:    repeat
6:       Choose z ∈ N(a) with maximal Q^t(a,z) and z ∉ L
7:       TTL = TTL − 1
8:       L = L ∪ {z}
9:       Send event e_i to neighbor z
10:      if No response from z within time T then
11:         Q^t(a,z) = (1 − α)Q^{t−1}(a,z)
12:         if Q^t(a,z) < ε then
13:            N(a) = N(a) − z
14:         end if
15:         TTL = TTL − 1
16:         L = L − {z}
17:      end if
18:   until feedback ≠ 0 or |N(a)| = φ
19: end if
20: if |N(a)| = φ then
21:    Return fail
22: else
23:    Q^t(a,z) = (1 − α)Q^{t−1}(a,b) + α ∗ feedback
24:    Return success
25: end if
```

Path learning algorithm also helps to maintain the topology of the network when node failures happen. Node a will remove the dead node z from its neighbors when $Q^t_{z \in N(a)}(a,z) < \epsilon = 0.1$ (line 12-13). Intuitively, node a cannot locally distinguish a link failure from a node failure for one time data loss, but it can recognize a neighbor b as a dead node if sensor b continuously fails to deliver data for several times. Also, in order to avoid network partition, node a needs to pass a request in the network when $N(a)$ is small. Any node with small number of neighbors can accept the request and become a neighbor of node a.

4.3 Spanning Tree

From the local view of node a, the feedback from b seems only reinforces the path $\langle a, b\rangle$. But, if we look at the data delivery processes in the whole network, we can find that data flows generate multiple trees, where each tree represents a successful data delivery process for a set of relevant events. The probability $Q^t(a,b)$ serves as the gradient of the path $\langle a, b\rangle$ and controls the direction of data flows.

Once we have multiple trees in the network, there will be two cases for sensor a to deliver an event: (1) if node a is on a tree and is not the root node, it just forwards the event according to the gradient of the path; (2) otherwise, node a just chooses a neighbor with maximal Q value to pass the event. The latter may extend the existing tree or connect the tree with another one.

An important question is how to avoid cycles when a tree connects with another one. A simple way is to check the pedigree of each event before it was sent out. For example, if node x attempts to send event e_i to node a and a already appears in the pedigree of event e_i, x will first reset the gradient between them to 0.5, e.g., $Q^t(x,a) = 0.5$ and then pass e_i to another neighbor y with the highest Q value, where $y \in N(z)$ and $y \neq a$.

Multiple trees will evolve and merge with each other and finally one spanning tree may emerge in the network. The advantage of a spanning tree is that the tree keeps track of the node and link failures and connects most of nodes for reliable data delivery. Note that, in mobile sensor networks, any node on the spanning tree can fuse relevant events and stop the data delivery processes. Hence, our approach is different from other work on building spanning trees in static microsensor networks, e.g., [10], in which nodes on the spanning tree transmit all data to a single sink node and only the sink node has the fusion function.

Table 1 summarizes the properties of four algorithms in terms of efficiency and robustness. Here efficiency denotes the number of messages for data delivery, including both data and feedbacks. The efficiency of path reinforcement algorithm with retransmission is worse than that of path learning algorithm since the former randomly chooses a neighbor, which may be found as a dead node before. Robustness denotes the data loss rates when failures happen. Both random walks and path reinforcement algorithm are vulnerable since they do not have retransmission mechanisms. They do not check if a neighbor has received the data or not.

In the experiments we will evaluate only three of them: random walks, path reinforcement algorithm (without retransmission) and path learning algorithm (only for the spanning tree).

5. ANALYSIS OF FUSION PROBABILITY

In this section we analyze the probability of relevant events being fused in a random network using random walks.

For simplicity, we do not consider link and node failures

in the formal model. We assume there are some relevant events e_1, e_2, ..., e_F generated by multiple sensors at the same time. For any sensor node a, the probability that node a knows e_i at hop 0 is $P_a(e_i, 0)$, where $1 \leq i \leq F$.

The probability that sensor node a knows event e_i at hop h as $P_a(e_i, h) \rightarrow [0, 1]$. In a random network with homogeneous sensor nodes, we can model the probability that any sensor node a knows event e_i at hop 0 is

$$P_a(e_i, 0) = 1/|V| \tag{3}$$

where $a \in V$ and $|V|$ is the total number of sensor nodes. Given the relevant events e_1, e_2, ..., e_F, the probability that sensor node a has all these events at hop h is

$$P_a(\Delta, h) = P_a(e_1, h)P_a(e_2, h) \dots P_a(e_F, h) \tag{4}$$

where Δ is the fused result of events e_1, e_2, ..., and e_F.

The probability of all relevant events being fused by any member of the network V is,

$$P_V(\Delta, h) = 1 - ((1 - P_a(\Delta, h))(1 - P_b(\Delta, h)) \dots) \tag{5}$$

where $V = \{a, b, \dots, \}$. $P_V(\Delta, h)$ is called the probability of fusion. We assume the condition $P_a(\Delta, h) = P_b(\Delta, h) = \dots$ holds in a homogeneous logical network, and we can rewrite the probability that the team can do the fusion at hop h as,

$$P_V(\Delta, h) = 1 - (1 - P_a(\Delta, h))^{|V|} \tag{6}$$

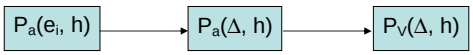

Figure 2: The relationships of three probabilities in the formal model

Figure 2 gives relationships among three probabilities in the formal model: $P_a(e_i, h)$, $P_a(\Delta, h)$, and $P_V(\Delta, h)$. From Equation 4 and Equation 6, we can find that the computation of the probability of fusion $P_V(\Delta, h)$ is trivial once we have the value of $P_a(e_i, h)$. Next we will describe how to compute the probability $P_a(e_i, h)$ for random walks.

Initially we have $P_a(e_i, 0) = 1/|V|$ for any sensor a. Its probability of knowing event e_i at hop 1, $P_a(e_i, 1)$, depends on the follows,

- Its probability of knowing event e_i at hop 0, $P_a(e_i, 0)$,
- The probability of being told e_i from its neighbors at hop $h = 1$.

The latter depends on two parts: (1) the probability of knowing e_i for one of its neighbors b at hop zero ($h = 0$), $P_b(e_i, 0)$; and (2) the probability of whether sensor a can get e_i from any neighbor b, denoted as $\delta(a, h)$.

The challenge is how to estimate $\delta(a, h)$ for random walks. Intuitively, random walks can be viewed as a random walker moving around in the network. If the walker's position is on node b, then it has $1/m$ probability to visit one of its neighbor. If not, sensor b cannot pass e_i to sensor a even when b knows e_i. Figure 3 shows an example of random walks for event e_i in the network, where the random walker visits node c, b, j, and g and its existing position is at g. In this case, b cannot pass event e_i to a, although a is b's neighbor and b knows event e_i.

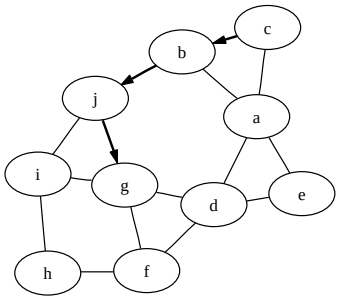

Figure 3: A example of random walks in the network, where the random walker visits node c, b, j, and g.

In other words, the event can be propagated from b to one of its neighbors a only and if only the walker's position is on b. For a random walk of e_i with $h - 1$ hops, b is one of the nodes that e_i visits, then we have $\delta(a, h) = 1/h$ for a $h - 1$ hops random walk.

Now let's estimate the probability being told e_i from its neighbors at hop h. Obviously, we simply have $\delta(a, 1) = 1$ when $h = 1$. Assume b has m neighbors, where m is the average degree of the network. The probability that b will forward e_i to a at hop one is $\delta(a, 1) * 1/m$. The probability of being told e_i by one of the neighbor at hop one is $P_b(e_i, 0) * \delta(a, 1) * 1/m$. If sensor node a has m neighbors, the probability a knows e_i from any of its neighbors is

$$P_b(e_i, 0) * \delta(a, 1) * 1/m * m = P_b(e_i, 0)$$

Similarly, the probability a knows e_i from its neighbors at hop h is $P_b(e_i, h - 1) * \delta(a, h)$, where $\delta(a, h) = 1/h$ refers to the probability of the random walker on node b.

The probability a knows e_i at h is

$$P_a(e_i, h) = P_a(e_i, h - 1) + (1 - P_a(e_i, h - 1))P_b(e_i, h - 1)/h \tag{7}$$

Up to now we consider to fuse all of the relevant events in the network. Fusing all relevant events is infeasible since, sometimes, we may only the range of F for a given target and we do not know the exact number of F. It is also unnecessary when the confidence of fused data is already high. Therefore, we may only need to fuse a subset of the events if the number of relevant events F is big,, e.g., f out of F events and $f \leq F$. We will try to stop the propagation of the rest of relevant events in the network. As illustrated in Figure 4, sensor a will stop propagating event e_4 if it already fused other three relevant events e_1, e_2, and e_3.

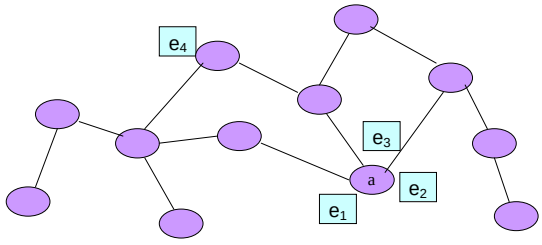

Figure 4: A subset of relevant events in the network

For similar sensors with fixed probabilities of false alarms, Thomopoulos et al. have proved that the probability of detection for fused results is maximized if the number of sensors is equal to or greater than three [20]. For the rest of this paper, we choose $f = 3$ and $P_a(e, h)$ as the probability of fusing three events at hops h. If $F = f = 3$, we need to fuse all three of them and it can be described as $f/F = 3/3$. If $F > 3$ and $f = 3$, we may only fuse any three of them together.

Now we try to derive the relationship between $P_a(e, h)$ and $P_a(\Delta, h)$. The probability of fusion for 3/4 case is

$$P_a(e,h)+(1-P_a(e,h))P_a(e,h)+\ldots+(1-P_a(e,h))^3P_a(e,h) \quad (8)$$

When $F > f = 3$, the equation can be generalized as

$$P_a(e,h)(1+(1-P_a(e,h))+\ldots+(1-P_a(e,h))^{c_1-1}) \quad (9)$$

where $c_1 = C_F^f$. We can find that $P_a(e, h)$ is still very small when we increase h to 10, e.g., $P_a(e, 10) = 0.001$. Hence, we have

$$1+(1-P_a(e,h))+\ldots+(1-P_a(e,h))^{c_1-1} \approx C_F^f$$

The probability of fusing any three of them can be estimated as

$$P_a(\Delta', h) = c_1 P_a(e, h)$$

where $c_1 = C_F^f$.

The probability of fusion for $F > f = 3$ can be estimated as

$$P_V(\Delta', h) = 1 - (1 - P_a(\Delta', h))^{|V|}$$

Obviously, we have $P_V(\Delta', h) \geq P_V(\Delta, h)$ when we only need to a subset of relevant events.

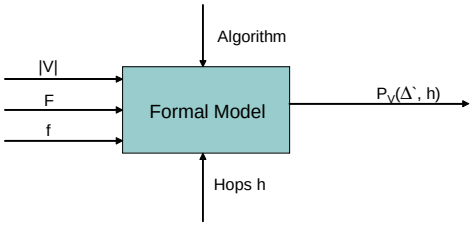

Figure 5: The inputs and outputs of the formal model

Figure 5 describes the inputs and outputs of our formal model. Given the size of networks $|V|$, two values F and f, we can easily compute the probability of fusion for random walks at hop h.

Figure 6 visualizes the outputs of the formal model — the probability of fusion $P_V(\Delta', h)$ — for a network with 100 nodes, where $f = 3$. While we may not use random walks in practice, the results here provide a baseline to test our proposed algorithms and estimate the low bound of TTL for a given network and the number of relevant events F.

6. EXPERIMENTAL RESULTS

In this section, we empirically study the data delivery algorithms. We use a random network of 100 nodes and each of them has, on average, four neighbors. Nodes are randomly chosen as the source of relevant events and they propagate the events according to different algorithms.

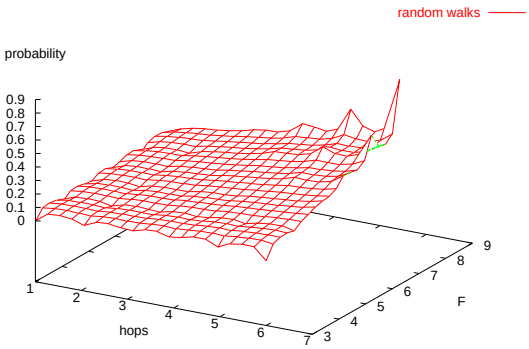

Figure 6: Probability of fusion for random walks for a network with 100 nodes (f = 3).

The experiments here serve as two purposes in this paper. First, we would like to validate our formal model for random walks using simulation results. For example, in the formal model, we assume each node has the same probability of knowing event e_i at hop h, which is not always true in practice. Second, we would like to study the robustness and efficiency of our algorithms.

6.1 Probability of Fusion

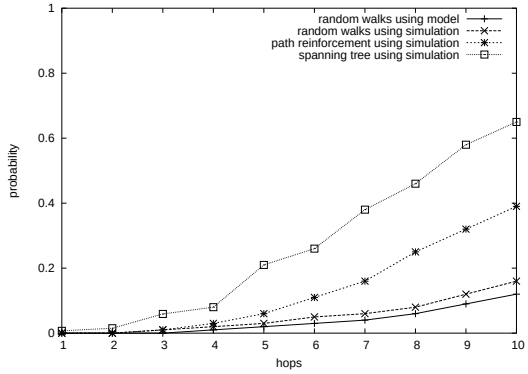

Figure 7: Probability of fusion for random walks, path reinforcement algorithm, and spanning tree (3/3 case).

We first study the probability of fusion for random walks, path reinforcement algorithm (without retransmission) and spanning tree when there is no failure. The probabilities of fusion from simulations are averaged over 1000 experiments.

Figure 7 and Figure 8 compare the probabilities of fusion for 3/3 and 3/5 cases using our formal model (only for random walks) and simulations. We can find that the results for random walks are consistent with those from the formal model. Also, random walks are quite easy to beat since they do not consider the relevance between events. For both 3/3 and 3/5 cases, path reinforcement algorithm perform better

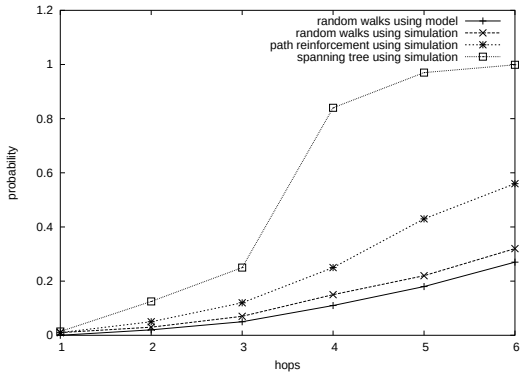

Figure 8: Probability of fusion for random walks, path reinforcement algorithm and spanning tree (3/5 case).

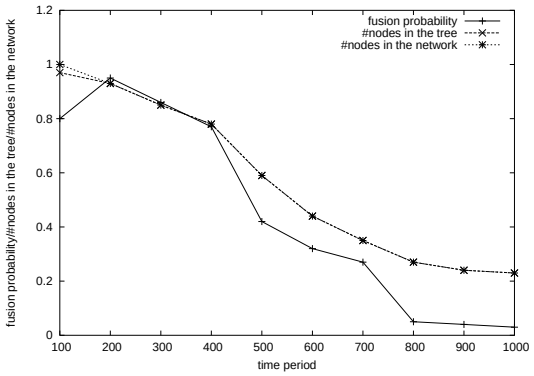

Figure 9: The fusion probability, percentages of active nodes in the tree and in the network (the line for #nodes in the tree is overlaid by the line for #nodes in the network).

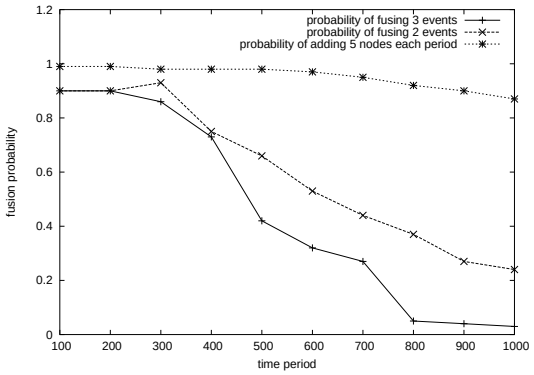

Figure 10: The fusion probability in the network when adding more nodes.

than random walks.

We also study the effects of spanning tree for data delivery and fusion in the network. We use the first eight sets of relevant events to train the spanning tree and then we test the probability of fusion for the spanning tree. The results indicate that learning is quite effective for sensor data delivery. For the 3/5 case, the local learning scheme at least doubles the probability of fusion for path reinforcement algorithm when the hop of events is greater than three. The probability of fusion for 3/5 is guaranteed to be high if we allow the set of events to be propagated 6 hops in the network with 100 nodes.

6.2 Robustness

In this section and next section, we study if our localized algorithm is robust to failures in the network. We choose a random number between 0 and 0.01 as failure rate λ_a for any sensor a. We introduce a random number between 0 and 0.01 as link failure rate between a node and any of its neighbors.

Figure 9 describes the fusion probability and percentages of active nodes in the tree and the network when $F = 5$. The dead nodes are detected after every 100 time periods. The figure tells us that trees evolve and merge and quickly one spanning tree covers most of the sensor nodes. Moreover, the tree always connects the nodes in the remained network. Another result is the probability of fusion decreases when the size of the network decreases. The reason is that the number of relevant events for fusion F becomes small, e.g., $F < 5$.

We consider the worst case of a mobile sensor network, where more than 50% nodes are destroyed after 500 time periods. In the hostile environments like this, e.g., a battlefield, we probably need to reduce the necessary number of events for fusion f . For example, if we reduce f from 3 to 2 ($F = 5$ on the network of 100 nodes), we can still reach the fusion probability of 70% after 500 time periods (Figure 10). Another solution is to incrementally deploy more sensor nodes to the sensor networks, as suggested by [6]. Figure 10 shows that, if we add 5 nodes to the network after every 100 time periods, we can still keep pretty high probability of fusion.

One concern of node failures is whether fused events can be lost when the fusion node happens to be destroyed. We study the fusion process and find this is not a problem when F is relatively big. For example, there are multiple copies of fused events at different nodes for 3/5 case when each event is propagated to 6 hops. The redundancy of fused events increases the survivability of the information and success rates of coordinating plans and actions for the network.

6.3 Efficiency

One question we did not answer in the formal model is if path learning algorithm is more efficient than path reinforcement algorithm. Since a node using path learning algorithm needs the feedback from its neighbors to learn their reliabilities. This will at least double the traffic in the network.

From Figure 11 we can find that if the events are allowed to propagate longer in the path reinforcement algorithm, both algorithms have very high probability of fusion. However, path reinforcement algorithm is vulnerable to node and link failures. Its probability of fusion becomes low when node failures happen. One reason is that the events are lost during information delivery in path reinforcement algorithm. In path learning algorithm we maintain the reliable data delivery of events through a retransmission mechanism,

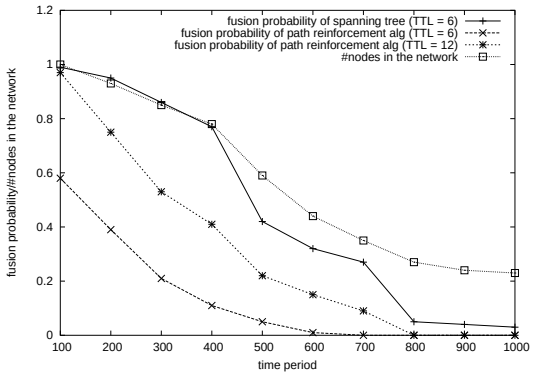

Figure 11: Probability of fusion and efficiency for path reinforcement algorithm and spanning tree.

where an event is delivered to another neighbor if a node or link failure is found.

7. CONCLUSION

This paper studies localized data delivery algorithms for mobile ad hoc sensor networks with failures. The empirical results show that our path learning algorithm significantly increases the probability of fusion in the network with node and link failures.

In future work we will develop methods for tuning system parameters, such as learning rate α in path learning algorithm, for different numbers of relevant events F, and node and link failure rates. Moreover, we have previously shown that small-world networks are more efficient than random networks for passing tokens in large teams [21, 18]. We plan to study the effects of different network topologies on probabilities of fusion for our path learning algorithm, especially when both node and link failures happen in the network.

Acknowledgements

This research has been sponsored in part by AFRL/MNK Grant F08630-03-1-0005 and AFOSR Grant F49620-01-1-0542. We are indebted to the anonymous reviewers for their helpful comments.

8. REFERENCES

[1] Justin A. Boyan and Michael L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems*, pages 671–678, 1994.

[2] Chee-Yee Chong and Srikanta P. Kumar. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.

[3] David L. Hall, editor. *Mathematical Techniques in Multisensor Data Fusion*. Artech House Inc., 1992.

[4] David L. Hall and James Llinas. An introduction to multisensor data fusion. In *Proceedings of the IEEE*, pages 6–23, 1997.

[5] G. Hoblos, M. Staroswiecki, and A. Aitouche. Optimal design of fault tolerant sensor networks. In *Proceedings of the IEEE International Conference on Control Applications*, pages 467–472, 2000.

[6] Andrew Howard, Maja J. Mataric, and Gaurav S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots Special Issue on Intelligent Embedded Systems*, 13(2):113–126, 2002.

[7] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MobiCom*, pages 56–67, 2000.

[8] Gal A. Kaminka and Milind Tambe. What is wrong with us? improving robustness through social diagnosis. In *AAAI*, 1998.

[9] Victor Lesser, Charles Ortiz, and Miland Tambe, editors. *Distributed Sensor Networks: A Multiagent Perspective*. Kluwer Academic Publishers, 2003.

[10] Ning Li, Jennifer C. Hou, and Lui Sha. Design and analysis of an mst-based topology control algorithm. In *InfoCom*, pages 1702–1712, 2003.

[11] Alexei Makarenko and Hugh Durrant-Whyte. Decentralized data fusion and control in active sensor networks. In *Proceedings of the Seventh International Conference on Information Fusion*, 2004.

[12] James Manyika and Hugh Durrant-Whyte. *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*. Ellis Horwood, 1994.

[13] Robin R. Murphy. National science foundation summer field institute for rescue robots for research and response. *AI Magazine*, 25(2):133–136, 2004.

[14] Ruixin Niu, Pramod K. Varshney, M. Moore, and Dale Klamer. Decision fusion in a wireless sensor network with a large number of sensors. In *Fusion*, 2004.

[15] H. Van Dyke Parunak, Sven A. Brueckner, and James J. Odell. Swarming coordination of multiple UAV's for collaborative sensing. In *Proceedings of the Second AIAA Unmanned Unlimited Systems Technologies and Operations Aerospace Land and Sea Conference and Workshop*, 2003.

[16] Matt Rosencrantz, Geoffrey Gordon, and Sebastian Thrun. Decentralized sensor fusion with distributed particle filters. In *UAI*, 2003.

[17] Allison Ryan, Marco Zennaro, Adam Howell, Raja Sengupta, and J. Karl Hedrick. An overview of emerging results in cooperative UAV control. In *Proceedings of 43rd IEEE Conference on Decision and Control*, 2004.

[18] Paul Scerri, Yang Xu, Elizabeth Liao, Justin Lai, and Katia Sycara. Scaling teamwork to very large teams. In *AAMAS*, pages 888–895, 2004.

[19] Devika Subramanian, Peter Druschel, and Johnny Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. In *IJCAI*, 1997.

[20] Stelios C.A. Thomopoulos, R. Viswanathan, and D. C. Bougoulias. Optimal decision fusion in multiple sensor systems. *IEEE Transactions on Aerospace and Electronic Systems*, 23(5):644–653, 1987.

[21] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, June 1998.

[22] Feng Zhao and Leonidas Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann Publishers, 2004.