Geographic Routing in Distributed Sensor Systems without Location Information

Bin Yu
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.
byu@cs.cmu.edu

Katia Sycara
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.
katia@cs.cmu.edu

Abstract - ***Geographic routing protocols have been widely used in microsensor networks, however, they cannot directly apply to distributed mobile sensor systems as mobile sensors often do not know their neighbors' exact physical locations. In this paper we consider geographic routing in distributed sensor systems without location information. We address the problem by introducing a lightweight and distributed virtual coordinate assignment protocol. We focus on the effectiveness of routing algorithms for distributed data fusion in the system and provide a detailed analysis of several routing algorithms for a sensor system with group mobility. Our simulation results show that controlled data flows significantly increase the probability of relevant data being fused.***

Keywords: distributed sensor systems, data fusion, geographic routing.

1 Introduction

Distributed sensor systems 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 [8, 12, 15]. One phenomenon in mobile sensor systems is that 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 [10].

Various statistical inference techniques have been studied for distributed data fusion in sensor systems, however, these approaches do not consider the control of data flows [11]. Uncontrolled data flows may cause large amounts of conflicting plans and have severe effects on the performance of the whole system when mobile sensors work together to conduct high-level tasks. Thus, one crucial problem for sensor systems is how to optimize routing strategies for in-network data fusion [19], so that relevant data can be fused with small communication overheads.

Geographic routing protocols have been widely used in microsensor networks [22, 23], but the effectiveness of these protocols has not been fully studied in mobile sensor systems. In existing geographic routing protocols, e.g., GPSR, forwarding decisions are made locally, based only on the node's own position, the positions of its neighbors, and the position of the destination [7]. While location awareness is of great importance for data routing, geographic routing cannot directly apply to mobile sensors as they often do not know their neighbors' *exact* geographic locations. Determining geographic location needs to consume much energy and bandwidth. Sometimes even if location information is available, it cannot represent the exact physical location of the mobile sensor.

Motivated by recent work on virtual coordinates [13, 16], we consider geographic routing for data fusion without location information. We introduce a lightweight and distributed virtual coordinate assignment protocol. Our virtual coordinate is based on the hop distance and the overall connectivity graph of the system. Unlike [16], we do not try to approximate physical coordinates, so only a relatively small overhead is needed to set up the logical coordinate. Also, different from [13], we consider the topology as a graph instead of a tree, since the hierarchical structure of a tree tends to overload the root node and other nodes at low levels of the tree.

Our work is closely related to [2], while they focus on the performance of greedy routing in the virtual coordinate and the physical coordinate, here we are particularly interested in the effectiveness of routing algorithms for distributed data fusion in the system. In our approach, instead of routing the data to the destination in a greedy manner, the data is routed to increase the probability of relevant data being fused, while minimizing communication overload. There is no querying process and each node proactively forwards the data to one of its neighbors in the virtual coordinate. Moreover, each node has to exploit the locality of sensor data and make its routing decision based on local knowledge of itself and its neighbors.

In this paper we consider scenarios such as disaster rescue and battlefield surveillance, where hundreds of sensors move with group mobility [6] and each node has a relatively fixed neighborhood in the logical coordinate space. We provide a detailed analysis of several routing algorithms for the purpose of sensor data fusion in the system. We also study the robustness of the routing algorithms to network dynamics such as node

failures and additions. The simulation results show that relevant data is efficiently fused when sensors exploit both locality and relevance of sensor data during the routing process.

The rest of this paper is organized as follows. Section 2 summarizes the relevant literature. Section 3 briefly describes the background of distributed sensor systems and the construction of the virtual coordinate space. Section 4 investigates the routing algorithms for data fusion. Section 5 presents some experimental results for the robustness and efficiency of the algorithms. Section 6 discusses the main themes and some directions for future research.

2 Related Work

Multisensor data fusion has been intensively studied for a long time, where both data fusion and control algorithms are centralized [3, 4, 20]. We will not discuss the literature on multisensor data fusion since they fall outside of the scope of this paper.

Research on distributed sensor data fusion has been focused on statistical inference techniques, such as [10, 11, 14, 17]. [14] investigates an optimum local fusion detection threshold for hypothesis testing by local sensors. [17] presents a Bayesian technique for decentralized state estimation using particle filters. [10] and [11] consider distributed data fusion as a state estimation problem using information filter (a variant of Kalman filter). These approaches study detection and tracking performance of distributed sensor systems, but there is no control on data flows in the system. Uncontrolled data flows may cause large amounts of conflicting plans and have severe effects on the performance of the whole system when mobile sensors work together to conduct high-level tasks.

Many geographic routing protocols have been proposed for wireless microsensor networks, e.g., GPSR [7]. Geographic routing is scalable, but it requires nodes know their precise locations. Recently, [13] and [16] studied virtual coordinates for geographic routing without location information. The idea is to assign logical coordinates to each node and then use greedy forwarding in the virtual coordinate space. These two protocols work well even in the face of physical obstacles, but both of them require relatively large setup overhead in the forms of several rounds of floods or iterations of relaxation.

Our work is closely related to [2], but we focus on the effectiveness of routing algorithms for distributed data fusion. We present a lightweight and distributed virtual coordinate assignment protocol based on the hop distance and the overall connectivity graph of the system. Unlike [16], we do not try to approximate physical coordinates. Instead, we need a relatively small overhead to set up the logical coordinate. Also, different from [13], we consider the topology as a graph instead of a tree. The hierarchical structure tends to overload root node and other nodes at low levels of the tree. Moreover, the setup overhead in our protocol is equivalent to that of determining the order of children in the tree.

3 Preliminaries

In this section, we give some background of sensor data fusion in distributed sensor systems and the construction of virtual coordinate space.

3.1 Background

Mobile sensors such as robots or UAVs (Unmanned Aerial Vehicles) are frequently used in disaster search and rescue, or battlefield surveillance, where hundreds of robots or UAVs move together to search for trapped victims or targets. For example, in military applications, hundreds of UAVs are deployed as a team in the battlefield for target detection, tracking, and classification [18]. These UAVs collaboratively explore unknown regions and search moving, possibly evading targets in the complex and dynamic environment [21].

A target T_K may be detected by multiple UAVs, but each UAV cannot initialize its plans based on its own raw sensor data about the target, e.g., engagement of its missile with the target. [1] The reason is that the raw sensor data from a single 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 routed to other nodes for fusion in the system [5].

Formally, the connectivity graph of 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. Also, each node knows its immediate one-hop neighbors, but it does not know their physical locations.

In this paper, we only consider a distributed sensor system with group mobility [6]. This often happens in the scenarios of disaster search or battlefield surveillance, where large numbers of robots or UAVs maintain a desired shape (formation) while following a desired trajectory.

In order to efficiently route the data for fusion, we have to exploit two properties of sensor data in the system.

- *Relevance*: We need to consider the relevance of events when we route the data in the system. Here two events are relevant if they are referring to the same target.

- *Locality*: We also need to consider the locality of data during the routing process. Locality means sensors that detect relevant data are likely to be close to each other in the system.

[1] We assume that each UAV only has the chance to scan the target once due to its kinematic constraints.

3.2 Coordinate Construction and Assignment

In this section we introduce a lightweight and distributed virtual coordinate assignment protocol. It includes constructing a virtual coordinate space for the system and assigning a virtual coordinate (x, y, z) to each node during network initialization. The basic idea is to find three boundary nodes as anchor nodes for X, Y, and Z in a distributed manner, so we can assign the coordinate to each node through triangulation. While this is not part of our research contribution, we present it here for completeness.

In [2] the authors give some heuristics on how to choose the three anchor nodes X, Y, and Z, such that they have high probability on the boundary of the network and they are not too close to each other. The whole process can be summarized as follows,

- One randomly selected node generates a W message and its hop counter is initialized as *zero*. The node is only used to select the other three anchors X, Y, and Z. The counter for the W message is increased by *one* by the forwarding nodes. If a node gets more than one message, it will only forward the W message with the smaller hop counter.

- After all nodes have received the W messages, the nodes that have maximal hop counter of W messages within a two-hop neighborhood consider they are in the boundary of the network and generate X message with initial hop counter *zero*. Similarly, the counter of X messages is increased by *one* each time by forwarding nodes. If a node receives more than one message, it will choose the one with smaller hop counter of W message. [2]

- The propagation Y and Z messages is similar to X, but selection of the anchor of Y and Z needs to satisfy some given rules. For example, we have to choose Y anchor with maximal hop counter for X message and the hop counter of W is greater than a parameter γ. In this case, we can ensure that Y is not aligned with X and W. Otherwise, if only the distance from X is considered to select Y, it would be possible to have Y very close to W.

At the end of the process each node is assigned with a virtual coordinate (x, y, z) based on its hop distance to the anchors. Figure 1 shows an example of coordinate assignment through triangulation, where the three anchor nodes are with coordinates $(0, 1, 1)$, $(1, 0, 1)$ and $(1, 1, 0)$.

Note that we measure the distance as hops, some nodes are likely to have the same coordinates. We can see that the hop distance for nodes with same coordinates lies within a limited number of hops from each other. In Section 5, we show that the coordinate system can effectively support geographic routing for data fusion in the system.

[2]Note that each X message is generated after a random time in order to reduce the number of messages in the system.

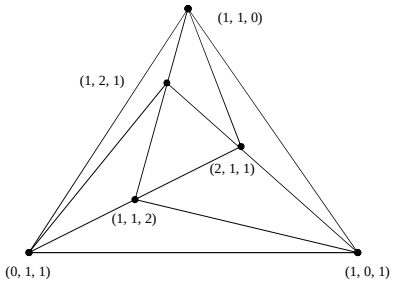

Figure 1: An example of coordinate assignment through triangulation.

3.3 Sensor Data

If sensor a detects a target T_K, it will generate an event e_i about target T_K. Here sensor a is also called a *source* node. A set of events $\langle e_1, e_2, e_3, \ldots e_F \rangle$ are relevant if they are referring to the same target, where F is the total number of events for target T_K. Formally, event e_i can be denoted as a tuple $\langle sender, identity, p\_location, v\_location, TTL, pedigree \rangle$, where

- *sender* is the ID of the sensor that detects the target. The ID could be predefined or a random number chosen on a large range.

- *identity* is the decision about the target made by the sensor.

- *p-location* is the physical location of the target T_K, denoted as (x_i, y_i, z_i). [3]

- *v-location* is the location of the sensor that detects the target T_K in the virtual coordinate space, denoted as $(\hat{x_i}, \hat{y_i}, \hat{z_i})$.

- *TTL (time-to-live)* is the maximal number of hops allowed for the event propagation in the system.

- *pedigree* is the list of nodes event e_i has visited, denoted as L. Note that pedigree is used to avoid cycles during event propagation.

Note that fusing all relevant events is infeasible since, sometimes, we may only know 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 f if the number of relevant events F is big. The value of f depends on the target type and the sensor, but for simplicity, we just consider the case of $f = 3$ in this paper.

Moreover, we need to stop the propagation of the rest of relevant events in the system. These redundant events are harmful for high-level team coordination since they may induce conflicting plans. For example, the same target could be engaged by multiple sensors, which will waste the valuable resources and

[3]The physical location of a target is mainly used in the data association process to determine if two events are referring to the same target.

degrade the performance of the system. In Section 4 we will discuss how to control the redundant events after f out of F events are successfully fused in the system.

4 Algorithms

This section describes routing algorithms for data fusion in sensor systems. Instead of routing the data to the destination in a greedy manner, the data is routed to some nodes to increase the probability of relevant data being fused, while minimizing communication overload.

Note that there are two properties of sensor data in the system: locality and relevance. Next we describe three routing algorithms with increasing effectiveness.

- random walks, which are the naive routing algorithm and serve as the baseline for our study [9].
- path reinforcement algorithm, where sensors only exploit the relevance of sensor data;
- path reinforcement algorithm with geographic routing, where sensors exploit both locality and relevance of sensor data.

In random walks, sensor node a with event e_i first decides if it will stop the propagation of e_i, where $1 \leq i \leq F$. If not, sensor node a just forwards 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.

Random walks are simple and easy to implement, but they are not efficient for data fusion as they do not exploit any relationships or properties of events and just pass events randomly in the system. Hence, we design other two algorithms — path reinforcement algorithm and path reinforcement algorithm with geographic routing.

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 and only if e_i and e_j are relevant.

Figure 2 shows an example of data flows in the system 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_j. 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 2, 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

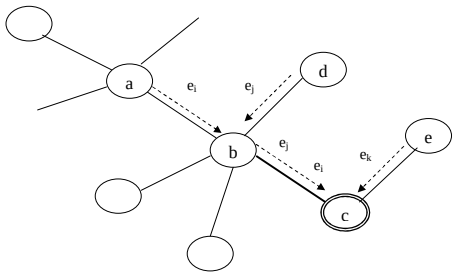

Figure 2: An example of data flows in the system using path reinforcement algorithm, where e_i, e_j, and e_k are three relevant events.

e_j meet at node b and they continue to walk randomly in the system, there will be a smaller chance that they can meet again and be fused with event e_k at node c.

4.2 Path reinforcement algorithm with geographic routing

In path reinforcement algorithm each node only considers the relevance of sensor data and routes the data in the connectivity graph. One way to exploit locality of sensor data is to limit the distance that the data is routed from the source node, so that the data can meet other relevant data in the neighborhood with high probability.

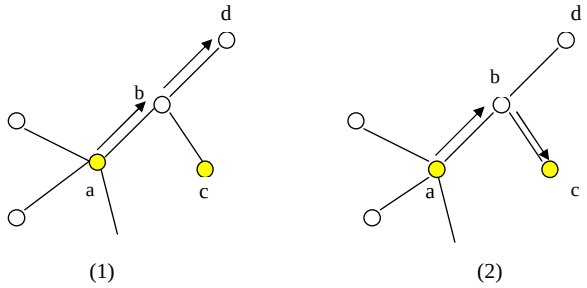

Figure 3: An example of data flows in the system with and without geographic information (1) path reinforcement algorithm in the connectivity graph; (2) geographical routing in the virtual coordinate.

Figure 3 describes an example of data routing, where node b needs to consider the (virtual) geographic locations of its neighbors when it makes the decision. In Figure 3 two nodes a and c have relevant data and a is a neighbor of c. Node b may forward the data to either c or d in path reinforcement algorithm, but forwarding to c will increase the probability of fusing the data generated from node c.

Hence, in each step, sensor node a needs to consider both relevance and locality of sensor data when it makes the decision of routing. A sensor will forward event e_i to one of its neighbors k according to path reinforcement algorithm if node a routes one of relevant events to k before. Otherwise, node a chooses k according to the the minimal distance to the location of source node $(\hat{x}_i, \hat{y}_i, \hat{z}_i)$. The distance d in the virtual

space is defined as follows,

$$d(k, e_i) = \sqrt{(x_k - \hat{x_i})^2 + (y_k - \hat{y_i})^2 + (z_k - \hat{z_i})^2}$$

where (x_k, y_k, z_k) and $(\hat{x_i}, \hat{y_i}, \hat{z_i})$ are the coordinates of the sensor k and the source node of event e_i, respectively. A sensor will choose one of two neighbors randomly if a tie happens.

Obviously, the relevant data will be more efficiently fused when sensors exploit both locality and relevance of sensor data during the routing process. Another issue is how to control redundant sensor data, so we can minimize the communication overhead and conflicting plans during the routing process.

The communication overhead for a set of events $\langle e_1, e_2, e_3, \ldots e_F \rangle$ is defined as follows,

$$c = \sum_{i=1}^{F} |L(e_i)|$$

where $L(e_i)$ is the pedigree of event e_i and $|L(e_i)|$ is the length of path for event e_i.

One way to reduce communication overhead is to allow some nodes to filter out redundant events after they successfully fuse a subset of F events. As we show in Section 5, the mechanism is quite effective and can filter out up to $5-10\%$ redundant messages, depending on the value of F and the routing algorithm.

4.3 Network Dynamics

The topology of sensor systems may change over time when some nodes fail and new nodes are added. When either of them happens, we have to modify the coordinates of some nodes in the virtual coordinate space. The overhead here mainly includes detecting and propagating changes in the logical coordinate space.

Node Failures Individual sensor tends to fail due to either energy (fuel) depletion or hostile environment. In either case, we need to be able to deal with failures with minimal overhead. Obviously, we do not want the anchors to flood the system, as these anchors may not know when the network topology changes. The key is how to detect the topology change locally, such that each node can efficiently propagate the change in the coordinate space.

Next we introduce the notion of upstream and downstream nodes, with which we can restrict the propagation to certain directions.

Definition 1 For any two nodes a and b, (x_a, y_a, z_a) and (x_b, y_b, z_b) are their coordinates. Node a is defined as an upstream node of b if any of the following conditions holds,

- $x_a < x_b$, or
- $y_a < y_b$, or
- $z_a < z_b$

Node b is also called a downstream node of node a.

When a node a detects that one of its neighbors k has failed, it first needs to adjust its coordinate if k is an upstream node of a. In this case, the failed node tends to increase the coordinate of node a since node k may connect a to one of anchor nodes. Node a will set its X coordinate as follows,

$$x_a = x_c + 1$$

where x_c is minimal X coordinate of the rest of a's neighbors except for k. The update of Y and Z is similar to X.

When node a changes its coordinate, the consistency of other nodes' coordinate can also be violated. To fix this, node a needs to propagate its coordinate to any of its neighbors who are downstream node of a. The process continues until there is no further downstream node.

Sometimes the anchor nodes may fail. In this case, we need to select one from the neighbors of the anchor node as the new anchor node. Assume that anchor node X fails, then a neighbor of anchor X with minimal distance to X will be selected as the new anchor node for X, e.g., node a. The coordinate of a is updated as $x_a = x_a - 1$. Moreover, node a propagates its new coordinate to all of its downstream nodes.

New Nodes For many applications, it is desirable to add new nodes into the system. The new nodes may replace failed nodes, or to improve the sensor coverage area.

To join the system, a new node needs to first find a small number of nodes in the system as its neighbors (these neighbors are in the radio range of the new node). The question is how to assign the coordinate to the new node after it gets the coordinates of its neighbors. One simple way is to allow the new node to share the same coordinate with one of the neighbors. This may degrade the performance of geographic routing sometimes, but this minimizes the overhead of propagating the changes in the virtual space. The update of all downstream nodes of the new added node can be delayed when a node failure happens in the system.

However, when more nodes are added into the system, the anchor nodes may not be on the boundary of the network. This requires to initialize the anchor nodes again in the system. Time delay and communication overhead are two issues we need to consider. Unfortunately, it is difficult to achieve both goals. One way is to trade time for communication overhead, so we can reduce the number of broadcasts for selecting anchors for X, Y, and Z. The idea is that we can delay the generation of X messages for a random time, so some of them do not have to broadcast to other nodes in the system.

5 Experiments

In this section, we empirically study the routing algorithms for data fusion in a UAVs network. The UAVs are used to search for moving, and possibly evading targets in a hazardous environment. We do not consider packet loss in our simulation environments. The

topology is a random network of 100 nodes, each of which has, on average, six neighbors, if not specified.

In each iteration of the simulation, a random spanning tree is chosen in the system. The nodes on the tree have a set of relevant events and they propagate the events according to different algorithms.

Besides communication overhead, another main metric in our study is the probability of relevant data being fused, or *probability of fusion*. Formally, for a set of relevant events $\langle e_1, e_2, e_3, \ldots e_F \rangle$, the probability of fusion is defined as follows

$$p = \frac{\sum_{i=1}^{H} s_i}{H}$$

where s_i is one if f events are successfully fused at iteration i. Otherwise $s_i = 0$. H is the total number of iterations.

We first assume there is a a given set of relevant events, e.g., $F = 5$, and we study the probability of fusion for random walks, path reinforcement algorithm and path reinforcement algorithm with geographic routing. And then we look at some practical scenarios such as a UAVs network, where the total number of events may change during the search process. The results presented below are averaged over 1000 iterations.

5.1 Probability of Fusion

In the first experiment we study the relationship between probability of fusion and the hop number.

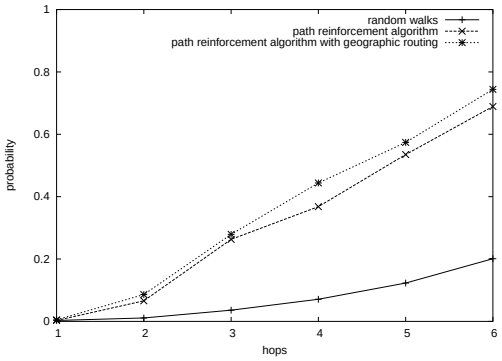

Figure 4: Probability of fusion for random walks, path reinforcement algorithm and path reinforcement algorithm with geographic routing, where the hops number is between one to six.

Figure 4 compares the probabilities of fusion for three routing algorithms, where the hops number is between one to six. We can find that random walks are quite easy to beat since they do not consider either locality or relevance between events. Path reinforcement algorithm performs significantly better than random walks, where path reinforcement algorithm at least triples the probability of fusion compared with random walks.

We also study the effects of geographic routing for data fusion in the system. The results indicate that geographic routing is quite effective for sensor data fusion. The scheme increases the probability of fusion for path reinforcement algorithm by 5% when the hop of events is greater than three. This is very impressive as totally the data is only allowed to propagate six hops and the first two hops are in the neighborhood of the source node.

5.2 Network Density

In this section we study whether the density of the network affects the probability of fusion. For example, each node of the network may have 4, 6, and 8 neighbors on average. Apparently, density has limited affect on the probability of fusion for random walk and path reinforcement algorithm as both of them work independently of the network density, as long as the network is connected.

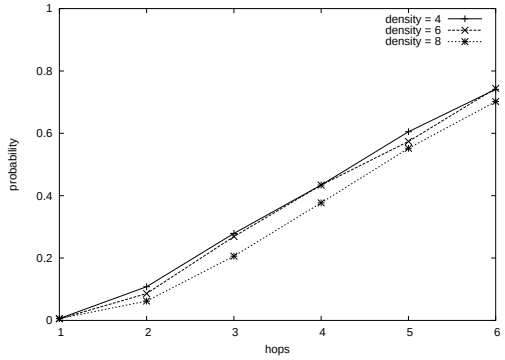

Figure 5: Probability of fusion for a network with different densities (only for path reinforcement algorithm with geographic routing)

One question is whether the density of the network will affect the probability of fusion for path reinforcement algorithm with geographic routing. Figure 5 shows that there are some slight differences for the probability of fusion when the density of the network is different. We can find that the probability of fusion tends to decrease as we increase the network density from six to eight. The reason is that geographic routing becomes less effective for high network density, as many nodes share same coordinates in the virtual space. Hence, we choose average degree six as network density for the rest of our experiments.

5.3 Dynamics of the Number F

In the above two experiments we evaluate the probability of fusion for a fixed number of relevant events. In practice, the total number of relevant events F may change when many UAVs cooperate to search targets in the environment. However, the number cannot be arbitrarily large as the target may be destroyed. Moreover, each target may be detected by different sensors at different timestamps. This is because there is some distance among UAVs to avoid collisions and a UAV needs some time to reach the same target after another UAV flies over the target [1].

For simplicity, we assume the following parameters,

- Once a target is detected by a sensor, the target is detected by one additional sensor after each simulation round.
- An event will be propagated three hops in the system during each simulation round.
- After the events are fused, it will take another two rounds to destroy the target.

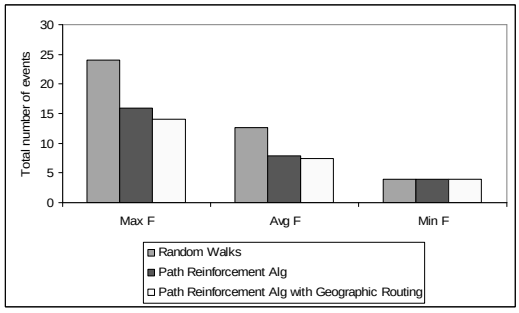

Figure 6: Numbers of relevant events for a target, averaged over 1000 iterations.

Figure 6 describes the number of relevant events for a target, where each event is propagated at most six hops. We measure the maximal, minimal, and average numbers of events for three routing algorithms. Note that the number here is equal to the number of sensors detected the target. From the figure it is clear to see that, for path reinforcement algorithm with geographic routing, the target is destroyed after it was detected by about seven sensors. As expected, the average number for random walks is almost doubled, where the target needs to be detected by about 13 sensors. Intuitively, this is because that sensors using path reinforcement algorithm with geographic routing can effectively fuse the data together and initialize the plan of target engagement earlier than using random walks.

5.4 Messages Filtering

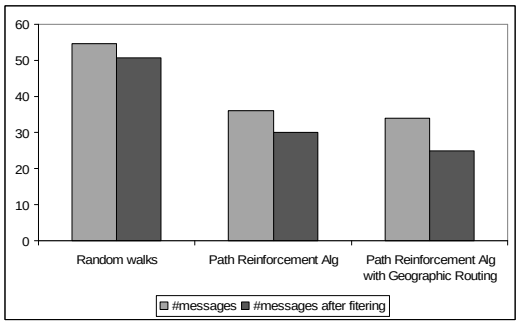

Figure 7: Number of messages during the data fusion process, averaged over 1000 iterations.

After an event is generated by a sensor, it is allowed to propagate by a maximal hop number, e.g., $TTL = 6$. However, when any three relevant events are fused by any node in the system, we would like to stop the propagation of the rest of events. This would reduce the traffic and the number of conflicting plans. One way to address this problem is to allow a node to filter other relevant events after the node fuses a subset of events, e.g., $f = 3$. Figure 7 shows the number of messages during the data fusion process. Note that any node k may stop routing event e_j if it has fused other three relevant events. Because of the role of message filtering, the number of messages is reduced up to five to ten percent without any additional communication overhead.

5.5 Effects of Node Failures

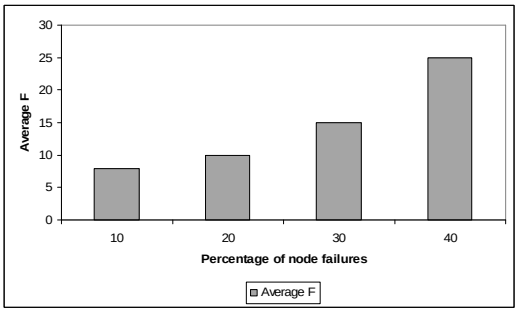

Figure 8: The effects of node failures on the total number of events for path reinforcement algorithms with geographic routing

In the last experiment we test the effects of node failures on the total number of events for path reinforcement algorithms with geographic routing, where about ten to forty percent nodes fail. Figure 8 suggests the average number F could become large when many nodes fail. This is due to the network partition problem. Some data is isolated somewhere in the network and they cannot be routed to other nodes. In future work, we plan to address this problem by allowing the system to self-organize when the network density drops.

6 Conclusion

In this paper we present a virtual coordinate-based routing algorithm for data fusion in distributed sensor systems. Our algorithm only requires each node to keep state only about its neighbors and does not require geographic location information. The empirical results show that our algorithm minimizes communication overheads, while still guarantees the high probability of data fusion.

In future work, we intend to continue the study of our algorithm by using more realistic link layer models and UAV kinematics such as message delay and flying speed. Additionally, we plan to study other mobility models for sensor systems and their effects on sensor data fusion. For example, most nodes have relatively stable neighborhood and a few of them move out

of their neighbors' radio range. Moreover, we would like to explore the effectiveness of our routing algorithms for high-level team coordination, e.g., task and resource allocation. Both problems have been intensively studied in the area of multiagent systems, but very few of them consider location awareness in team coordination [19].

Acknowledgements

This research has been sponsored in part by AFRL/MNK Grant F08630-03-1-0005 and AFOSR Grant F49620-01-1-0542.

References

[1] John Bellingham, Michael Tillerson, M. Alighanbari, and Jonathan P. How. Cooperative path planning for multiple UAVs in dynamic and uncertain environments. In *Proceedings of the 41st IEEE Conference on Decision and Control*, 2002.

[2] Antonio Caruso, Stefano Chessa, Swades De, and Alessandro Urpi. GPS free coordinate assignment and routing in wireless sensor networks. In *InfoCom*, 2005.

[3] Zeineddine Chair and Pramod K. Varshney. Optimal data fusion in multiple sensor detection systems. *IEEE Transactions on Aerospace and Electronic Systems*, 22(1):98–101, 1986.

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

[5] David L. Hall and Sonya A. H. McMullen. *Mathematical Techniques in Multisensor Data Fusion.* Artech House Publishers, second edition, 2004.

[6] Xiaoyan Hong, Taek Jin Kwon, Mario Gerla, Lihui Gu, and Guangyu Pei. A mobility framework for ad hoc wireless networks. In *Proceedings of the Second International Conference on Mobile Data Management*, pages 185–198, 2001.

[7] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *MobiCom*, 2000.

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

[9] Laszlo Lovasz. Random walks on graphs: A survey. In T. Szonyi D. Miklos, V. T. Sos, editor, *Combinatorics, Paul Erdos is Eighty.* Janos Bolyai Mathematical Society, 1993.

[10] 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.

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

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

[13] James Newsome and Dawn Song. GEM: graph embedding for routing and data-centric storage in sensor networks without geographic information. In *SenSys*, 2003.

[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 *Proceedings of the Seventh International Conference on Information 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] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *MobiCom*, 2003.

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

[18] 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.

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

[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] Patrick Vincent and Izhak Rubin. A framework and analysis for cooperative search using UAV swarms. In *Proceedings of ACM Symposium on Applied Computing*, 2004.

[22] Guoliang Xing, Chenyang Lu, Robert Pless, and Qingfeng Huang. On greedy geographic routing algorithms in sensing-covering networks. In *MobiHoc*, pages 31–42, 2004.

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