Replacing Failed Sensor Nodes by Mobile Robots

Yongguo Mei, Changjiu Xian, Saumitra Das, Y. Charlie Hu and Yung-Hsiang Lu
Purdue University, West Lafayette
{ymei, cjx, smdas, ychu, yunglu}@purdue.edu

Abstract

Sensor replacement is important for sensor networks to provide continuous sensing services. Existing approaches relocate redundant nodes to fill the holes created by failed sensors and require all or most sensor nodes to have mobility. However, mobility equipment is expensive while technology trends are scaling sensors to be smaller and cheaper. In this paper, we propose to use a small number of mobile robots to replace failed sensors in a large-scale static sensor network. We study algorithms for detecting and reporting sensor failures and coordinating the movement of robots that minimize the motion energy of mobile robots and the messaging overhead incurred to the sensor network. A manager receives failure reports and determines which robot to handle a failure. We study three algorithms: a centralized manager algorithm, a fixed distributed manager algorithm, and a dynamic distributed manager algorithm. Our simulations show that: (a) The centralized and the dynamic distributed algorithms have lower motion overhead than the fixed distributed algorithm. (b) The centralized algorithm is less scalable than the two distributed manager algorithms. (c) The two distributed algorithms have higher messaging cost than the centralized algorithm. Hence, the optimal choice of the coordination algorithm depends on the specific scenarios and objectives being optimized.

1 Introduction

Sensor networks have been intensively studied. Many studies focus on how to effectively collect and transfer data through energy-aware and fault-tolerant routing techniques [1, 7, 10, 11]. Sensor networks are assumed unattended in various environments such as disaster areas, hazard fields, or battle fields. Sensor nodes usually have simple designs, and their components are prone to failures. This can be especially serious in a hazardous environment, such as high temperatures or humidity. After a sensor network is deployed, some nodes may fail and leave holes in coverage. It is desirable to maintain the sensor network autonomously and keep the coverage. One way of maintaining the coverage is to replace failed nodes with functional ones. This is called *sensor replacement*. Wang et al. [13] propose using some redundant mobile sensors relocating themselves to fill the holes. They propose a cascading movement method to balance the energy cost and the response time of sensor replacement. However, mobility is an expensive feature, as mobile sensors need to have motors, motion control, and even GPS. Adding mobility to a large number of sensor nodes is expensive.

In this paper, we propose using only a few robots to assist sensor replacement. All robots are mobile and can pick, carry, and unload sensor nodes. When nodes fail, the robots move to the locations of the failed nodes and unload functional nodes. In our algorithms, a *manager* is a robot that receives failure reports and determines which robot to handle a specific failure. A *maintainer* is the robot that moves and replaces failed nodes. A robot can be both a manager and a maintainer. Senor nodes serve each other as *guardians* and *guardees* for failure detection.

In our proposed approach, since there are far fewer robots than sensors, the cost is expected to be lower than requiring most of the sensors to have mobility. Since sensors do not move, the routing algorithm can be more efficient for delivering sensing data than in mobile sensor networks. We present three different robot coordination algorithms: a centralized manager algorithm, a fixed distributed manager algorithm, and a dynamic distributed manager algorithm. The centralized algorithm has a central manager that receives failure reports from sensors and forwards them to individual robots. In the two distributed algorithms, the management responsibility is distributed over the robots, and each robot func-

[1]This work is supported in part by the National Science Foundation IIS-0329061. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Proceedings of the 26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'06)
0-7695-2541-5 /06 $20.00 © 2006 IEEE

tions as both a manager and a maintainer. The three algorithms are compared in respect to motion overhead and messaging overhead. We perform simulations of the three algorithms using Glomosim [14], and the experimental results show that (a) the centralized or the dynamic algorithm can achieve lower motion overhead and (b) both the fixed and the dynamic algorithms are more scalable but also have higher messaging overhead.

2 Problem Statement

We make the following assumptions in this study: (a) Sensors and robots are randomly uniformly distributed in a 2-dimensional field. Their locations are known by themselves. This can be enabled in the initial deployment process. The lifetime of a node is limited, and follows an exponential distribution with an expected value of T. (b) The robots travel at a constant speed v. Robots can localize themselves. The number of robots is much smaller than the number of sensors. (c) All the sensors and the mobile robots communicate via wireless links and form a multihop wireless network. (d) Replacement nodes are at the same locations as the corresponding failed nodes.

Using mobile robots for sensor replacement is essentially a coordination problem: how to coordinate multiple robots to maintain a large sensor network, in other words, to determine which robot should receive the failure report and which robot should replace the failed node. The problem can be divided into the following three stages.

(a) *Initialization* in three steps: 1) Setting up the roles of robots. A robot may be assigned to be a manager or a maintainer or both. 2) Setting up the initial relationship between the sensors and the robots. Every sensor node needs to know which robot is its manager and the location of the manager. 3) Setting up the guardian-guardee relationship among sensor nodes.

(b) *Failure detection and reporting*: When a guardian node detects a failure of its guardee node, the guardian reports the failure to a manager.

(c) *Failure handling*: Once a failure is reported, the manager dispatches a maintainer to the failure location to replace the failed node with a functional one. After replacing a failed node, the maintainer robot may need to update the manager or some sensors with its new location.

The goal of this study is to minimize the motion and the messaging overhead. The motion overhead is measured as the robots' traveling distance which reflects the energy consumed and the delay in repairing the failed sensors. The messaging overhead is measured as the number of wireless transmissions incurred for failure detection, reporting, and coordination.

3 Coordination Algorithms

The coordination algorithms address two issues: how to report a failure, and which robot should handle the failure. In the following, we present three algorithms: a centralized manager algorithm, a fixed manager algorithm, and a dynamic manager algorithm.

3.1 Centralized Manager Algorithm

The centralized algorithm has a robot functioning as the central manager. All failures are reported to the central manager. It then forwards failures to different maintenance robots. To simplify the design, we assume the manager does not move and is located at the center of the area to balance failure reports from all directions.

Initialization: In this stage, there are three types of messages. First, the manager broadcasts its location to all the sensor nodes and all the maintenance robots. Second, each maintenance robot sends a message with its current location to the manager, and broadcasts its location to their one-hop neighbor sensors, i.e., those sensors that are within transmission range of the robot. Third, the sensors broadcast their locations for establishing guardian-guardee relationship for failure detection. Every sensor receives one message from each of its neighbors, picks its nearest neighbor as its *guardian*, and then sends a confirmation message to the guardian to establish the relationship. After initialization, all the sensors and robots know the manager's location, the manager knows all robots' locations, and the guardian-guardee relationship is established.

We use geographic routing in the routing layer because it utilizes location information to reduce the routing overhead. The location service needed by geographic routing is provided as part of the coordination algorithms in the application layer. After the initialization stage, a robot needs to inform the manager and some sensors of its new location if the robot has moved.

Failure Detection and Reporting: After initialization, each sensor node periodically sends beacon messages to its one-hop neighbor nodes. If a guardian has not received any beacon from a guardee for a certain amount of time (three beaconing periods in our study), the guardian conceives that the guardee has failed and sends a failure message with the failed node's location to the manager. Similarly, if a guardee has not received

any beacon from a guardian for a certain interval, it assumes the guardian has failed and selects a new guardian from its one-hop neighbors. We assume that the probability of both a guardian and a corresponding guardee fail close in time is small and negligible, and hence all the failures can be detected and reported to the manager.

Failure Handling: Upon a failure report, the manager selects a robot based on the *current location* of each robot, specifically, the manager selects the robot whose current location is the closest to the failure. Upon receiving the request to replace a failed node, a robot moves to the failed node's location and replaces it by a functional node. Along the way, whenever it moves a certain threshold distance, it updates its location to the manager node via geographic routing and to the one-hop neighbor sensors via a one-hop broadcast. This allows the manager to keep track of the current location of each robot, and to continuously send replacement requests to the moving robot. A robot queues such requests and handles the failures in a first-come-first-serve fashion.

Although the central manager can always forward a failure to the closest robot, it can become a performance bottleneck in a large senor network where the distance traveled by the failure reports and replacement requests become too long.

3.2 Fixed Distributed Manager Algorithm

In the fixed algorithm, the area is partitioned into equal-size subareas. The number of subareas is the same as the number of robots, and each robot is assigned with an equal-size subarea. A robot is both the manager and the maintainer for its subarea. In the initialization stage, the robots first move to the centers of their corresponding subareas, and then broadcast their locations to all the sensors within their subareas. Each sensor node then selects the closest robot *myrobot* for failure report. The setup of guardian-guardee relationships is the same as in the centralized algorithm, with the restriction that they belong to the same subarea. Once a sensor detects a failure, it reports the failure to its "myrobot". In this way, each robot independently handles failures within its own subarea.

Similarly to the centralized algorithm, sensors broadcast their location information to their one-hop neighbors in initialization. However, the location update of a moving robot is slightly different. Since all the sensors within the robot's subarea can potentially report a failure to the robot, all the sensors in a subarea need to receive the robot's location update. To achieve this, all the sensors relay the robot's location update. During this process, a sensor may receive the same update

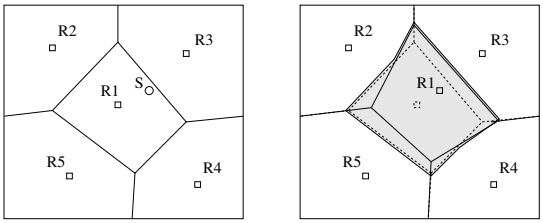

Figure 1. Voronoi graphs. (a) original Voronoi graph; a failure happens at 'S' inside R1's subarea. (b) After R1 moves to 'S', the Voronoi graph changes; the original graph is shown by dashed lines. The shading area shows the area that the robot needs to update its location.

message multiple times, but it relays the message to its neighbors only once. This is achieved by remembering the sequence number of the robot location updates it has relayed before. The fixed algorithm avoids the single manager bottleneck problem. However, because the area partitions are fixed, when a failure happened in one subarea, some robots in neighbor subareas may be closer to this failure than the robot in this subarea. Thus, the traveling distance tends to be longer. The dynamic algorithm described in next subsection improves on this by allowing dynamic area partition.

3.3 Dynamic Distributed Manager Algorithm

As the name indicates, there are no fixed boundaries between two robots in the dynamic algorithm. The boundaries between robots are constructed dynamically as Voronoi graph. Voronoi graphs have been used in wireless multihop networks to study coverage-boundary detection by Carbunar et al. [3], and to study the coverage problem by Meguerdichian et al. [8]. Figure 1 shows a Voronoi graph of five robots. The Voronoi graph partitions the area into five convex polygons with one robot in each subarea. Any sensor within each polygon is closer to the robot in that subarea than to any other robot. When a robot moves to replace failed nodes, the Voronoi graph needs to be adjusted.

The dynamic algorithm dynamically maintains the Voronoi graph. This is not realized directly by exchange locations between neighbor robots. Instead, robots do not know the border or the sensors they are responsible for. Each sensor node decides which robot to report a failure to and sets the robot as the sensor's "myrobot". This is achieved by robots' dynamic broadcasting their locations to sensors and sensors' dynamic adjusting their "myrobot"s. Different from the fixed algorithm, the sensors that receive and relay the broadcasting message are

not restricted to the nodes currently having the robot as their "myrobot"s. Some nodes currently in the neighbor subareas may need to switch their "myrobot"s to the moving robot, and such nodes may also need to relay the location update messages. In this way, the nodes update their "myrobot"s dynamically to be the closest robot.

In the dynamic algorithm, since a sensor node reports a failure to the closest robot, the robot achieves similar traveling distance as in the centralized algorithm without suffering the scalability problem. Note this is achieved at the cost of high messaging overhead, as the new location of a moving robot needs to be updated to many sensors. Compared with the fixed algorithm, the dynamic algorithm can have higher messaging overhead since some nodes in the neighbor subareas may also need to relay the location update messages. We will compare the messaging overhead in the next section.

4 Experiments

In this section, we compare the relative performance of the three algorithms using simulations.

4.1 Experimental Setup

We use Glomosim to simulate using robots to replace failed nodes. Glomosim [14] is a packet-level simulator for ad hoc networks with an accurate radio model. The link layer uses IEEE 802.11, and the radio model has a nominal bit-rate of 11Mbps. We differentiate three types of nodes: sensors, robots, and manager (for the centralized algorithm). The manager and the maintenance robots have the same transmission range of $250m$, while the sensors have a transmission range of $63m$ to save the sensor's power. In our simulations, the sensors are static, and robots move around to replace sensors. We have implemented an on-demand mobility model in which robots move on demand after receiving a failure report. The failure detection, the report and the replacement request are implemented in the application level.

In realistic scenarios, the failure happening rate is expected to be low and robots spend most of the time waiting for failure replacement requests. This conserves the robots' energy consumption. We selected the following simulation parameters: (1) The average area per robot is $200 \times 200m^2$. (2) The robots' speed is 1m/s based on the specification of Pioneer 3DX robots [9]. (3) The sensor node density is $\frac{50}{200\times200m^2}$, i.e., each robot is in charge of 50 nodes on average. (4) The number of maintenance robots varies from 1 to 16. The sensor area and the number of sensors change with different number of robots. For example, with 16 robots, the sensor area is $800 \times 800m^2$ with total 800 sensors. (5) In the fixed algorithm, the area is partitioned into squares. (6) The sensors' expected lifetime is 16000 seconds. (7) The simulation time is 64000 seconds. (8) The sensors' beaconing period for failure detection is 10 seconds. We choose a relatively short expected lifetime so that the simulation can finish in a reasonable amount of time, while still showing the relative maintenance overhead for different algorithms.

4.2 Geographic Routing and Robot Location Update

Routing in our system is implemented by geographic forwarding. As we discussed in section 2, each node knows its own geographic location and each robot can localize itself. Our implementation of geographic forwarding is based on face-routing [2] and our implementation parameters are the same as in GPSR [7]. During initialization, all the sensors broadcast their locations to their one-hop neighbors (within the radio transmission range) and the robots also broadcast their locations to some sensors and some other robots depending on the algorithms. Subsequently, a packet can be routed to a destination node using the local state at each node. Each packet contains the destination address in the IP header and the destination's location (x- and y-coordinates) in an IP option header. To forward a packet, a node searches its neighbor table and forwards the packet to its neighbor closest in geographic distance to the destination's location. Under the above greedy geographic forwarding, forwarded packets may potentially reach a node that does not know any other node closer to the destination than itself. This indicates a hole in the geographical distribution of nodes. Recovering from holes is possible using approaches such as GFG [2] or GPSR [7], using planar subgraphs to route around holes.

After initialization, sensors periodically send out beacons to their one-hop neighbors for failure detection. In addition, when one of the following two events occurs, extra updates of location information are required. (a) When a node detects a neighbor sensor node's failure, it deletes the failed neighbor from its neighbor table. After a failed node is replaced, the new node broadcasts its location to its one-hop neighbors. The neighbors send beacons containing their own locations. This enables the new node to set up its own neighbor table. (b) When a robot moves, the robot needs to send updates containing its new location to the manager and some sensors depending on which algorithm is being used as discussed in Section 3. In all the three algorithms, the robot updates its location whenever it moves away from the last

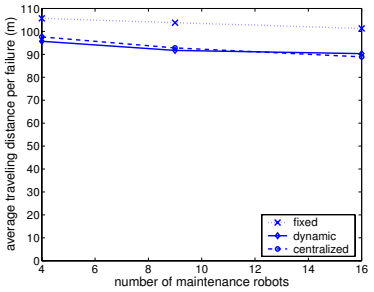

Figure 2. The average robot traveling distance as a function of the number of robots.

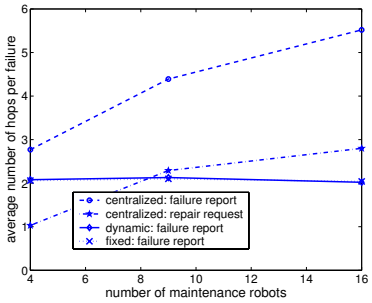

Figure 3. The average message passing hops per failure.

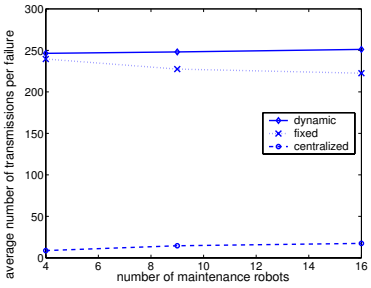

Figure 4. The average number of transmissions for location update per failure.

updated location by a distance threshold. The threshold depends on sensors' transmission range. In the simulations, we choose this threshold to be $20m$, less than $\frac{1}{3}$ of the sensors' transmission range ($63m$) to ensure that the robots can receive failure messages all the time.

4.3 Simulation Results

4.3.1 Motion Overhead

We run experiments with three different numbers of maintenance robots: 4, 9, and 16. We choose square numbers to make area partition easy. We skip one robot experiments, since there is little difference among the three algorithms. For a simulation with k^2 maintenance robots, the total area size is set to be $200 \times 200 \times k^2 m^2$, and there are a total of $50k^2$ sensors. For the fixed algorithm, we only show the results for the square partition method, as other partition methods (e.g., hexagon partition) show negligible difference in the overheads.

Figure 2 presents the average traveling distance per failure with different numbers of maintenance robots. The dynamic algorithm has similar motion overhead as the centralized algorithm, while the fixed algorithm has higher motion overhead. For example, with 16 maintenance robots, the dynamic algorithm can save 10.8% traveling distance compared with the fixed algorithm. In the fixed algorithm, a sensor's failure is reported to the robot in that subarea. There is no cooperation between robots and a sensor's failure is not necessarily reported to the closest robot. In the centralized or the dynamic algorithm, a failure is reported to the closest robot and the robots coordinate with each other indirectly. For this reason, the two algorithms have lower motion overhead than the fixed algorithm.

4.3.2 Messaging Overhead

The messaging overhead can be divided into four parts: initialization, failure detection, failure report and robot location update. Since all three algorithms are similar in initialization and failure detection, we focus on the overhead from failure report and location update.

Figure 3 shows the average number of hops for failure reports or replacement requests. Since we use geographic routing, the number of hops is roughly proportional to the distance between the report node (sender) and the manager (receiver). The failure reports and the replacement requests are delivered with 100% delivery ratio due to the high density of sensor nodes and low traffic load in the network. In the dynamic and the fixed algorithm, the failure is reported from a sensor node to a robot, and the distance between them is about 100m on average. This distance has the same average value as the robot traveling distance. Since the sensors' transmission range is 63m, passing a message cross $100m$ requires at least two hops. From the figure, the average number of hops traveled by the failure reports in the dynamic or the fixed algorithm is stable at about 2. This validates the effectiveness of the geographic routing. In the centralized algorithm, we consider both failure report and failure forwarding. Statistically, the distances traveled by failure reports and replacement requests have the same distribution, and they increase as the area increases because the central manager is always at the area center. However, the average number of hops traveled by failure reports and replacement requests differ. The reason is that sensors and robots have different transmission ranges. Failure reports have larger numbers of hops. The two curves in Figure 3 for centralized algorithm show the increasing trend and their difference. When the area increases, the number of maintenance robots increases accordingly, and the number of hops in the centralized algorithm also increases. Therefore, compared with the dynamic or the fixed algorithm, the centralized algorithm is less scalable.

Figure 4 shows the messaging overhead of location

updates in the three algorithms. We use the number of transmissions as the metric. The dynamic and the fixed algorithms result in much higher average number of transmissions per failure. Also, the dynamic algorithm has slightly higher number of transmissions than the fixed algorithm, as we discussed in the last paragraph of section 3.3. The high messaging overhead in the two distributed algorithms can be reduced by using more efficient broadcast schemes (e.g. [12]) which require only a subset of the sensors in each subarea to relay the location update messages.

5 Related Work

Wang et al. [13] propose relocating redundant mobile sensor nodes to fill the coverage holes and using a cascading movement method to balance the energy cost and the repairing time. Ganeriwal et al. [4] use low energy nodes, on predicted pending failure, broadcast a Panic request message. Nodes with high energy level respond with the Panic reply message if they can move without losing existing coverage. Howard et al. [5] treat sensor nodes as virtual particles and the virtual force between two nodes is larger if the two nodes are closer. As the virtual forces repel the nodes from each other, the sensor nodes tend to form a uniform distribution in the sensor area without coverage holes. These methods require the sensor nodes equipped with motors, steering devices, and GPS. To the best of our knowledge, we are the first to propose using a small number of mobile robots to replace failed nodes in a large scale sensor network, eliminating the mobility requirement for sensors. Ingelrest et al. [6] describe several broadcasting strategies for hybrid sensor networks in a more general context. However, their strategies primarily optimize broadcasting for fixed infrastructure points. Our solutions also require multi-hop broadcasting but are targeted at mobile robots.

6 Conclusion

In this paper, we have proposed using mobile robots for sensor replacement. Sensors detect failures of their neighbors and report to robots. Robots move and replace failed nodes with functional ones. We have presented three different algorithms and analyzed their performance by simulations. Simulation results show that the centralized algorithm is not scalable as the message passing distance increases with the sensor network area. The dynamic and the centralized algorithms have lower motion overhead. The fixed and the dynamic algorithms have higher messaging overhead. In general, the three algorithms have different properties, and the optimal choice of the algorithm depends on specific scenarios and objectives being optimized. In our future work, we will study more efficient location update mechanisms to reduce the messaging overhead in the dynamic and the fixed algorithms.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A Survey on Sensor Networks. *IEEE Communications Magazine*, 40(8):102–114, August 2002.

[2] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with Guaranteed Delivery in Ad hoc Wireless Networks. In *Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 48–55, 1999.

[3] B. Carbunar, A. Grama, and J. Vitek. Distributed and Dynamic Voronoi Overlay Maintenance for Coverage Detection and Distributed Hash Tables in Ad Hoc Networks. In *IEEE International Conference on Parallel and Distributed Systems*, pages 549–556, 2004.

[4] S. Ganeriwal, A. Kansal, and M. B. Srivastava. Self Aware Actuation for Fault Repair in Sensor Networks. In *the IEEE International Conference on Robotics and Automation*, pages 5244–5249, May 2004.

[5] A. Howard, M. J. Mataric, and G. S. Sukhatme. Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem. In *International Symposium on Distributed Autonomous Robotics Systems*, pages 299–308, June 2002.

[6] F. Ingelrest, D. Simplot-Ryl, and I. Stojmenovic. *Routing and broadcasting in hybrid ad hoc and sensor networks, Chapter in Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Networks, Jie Wu, editor*. Auerbach Publications, 2005.

[7] B. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *International Conference on Mobile Computing and Networking*, pages 243–254, 2000.

[8] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage Problems in Wireless Ad-Hoc Sensor Networks. In *INFOCOM*, pages 1380–1387, April 2001.

[9] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee. A Case Study of Mobile Robot's Energy Consumption and Conservation Techniques. In *International Conference on Advanced Robotics*, pages 492–497, 2005.

[10] G. J. Pottie and W. J. Kaiser. Wireless Integrated Network Sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[11] I. Stojmenovic and X. Lin. Power-Aware Localized Routing in Wireless Networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(11):1122–1133, November 2001.

[12] I. Stojmenovic and J. Wu. Broadcasting and Activity-Scheduling in Ad Hoc Networks. In S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, editors, *Ad Hoc Networking*. IEEE/Wiley, 2004.

[13] G. Wang, G. Cao, T. L. Porta, and W. Zhang. Sensor Relation in Mobile Sensor Networks. In *INFOCOM*, volume 4, pages 2302–2312, 2005.

[14] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks . In *Workshop on Parallel and Distributed Simulation*, pages 154–161, 1998.