Distributed Algorithms for Guiding Navigation across a Sensor Network

Qun Li, Michael DeRosa, and Daniela Rus
Department of Computer Science
Dartmouth College
{liqun, mdr, rus}@cs.dartmouth.edu

Dartmouth Computer Science Technical Report TR2002-435

***Abstract*— We develop distributed algorithms for self-reconfiguring sensor networks that respond to directing a target through a region. The sensor network models the danger levels sensed across its area and has the ability to adapt to changes. It represents the dangerous areas as obstacles. A protocol that combines the artificial potential field of the sensors with the goal location for the moving object guides the object incrementally across the network to the goal, while maintaining the safest distance to the danger areas. We report on hardware experiments using a physical sensor network consisting of Mote sensors.**

I. Introduction

We wish to create more versatile information systems by using adaptive distributed sensor networks: hundreds of small sensors, equipped with limited memory and multiple sensing capabilities which autonomously organize and reorganize themselves as ad-hoc networks in response to task requirements and to triggers from the environment. Distributed adaptive sensor networks are reactive computing systems, well-suited for tasks in extreme environments, especially when the environmental model and the task specifications are uncertain and the system has to adapt to them. A collection of active sensor networks can follow the movement of a source to be tracked, for example a moving vehicle. It can guide the movement of an object on the ground, for example a surveillance robot. Or it can focus attention over a specific area, for example a fire to localize its source and track its spread.

A sensor network consists of a collection of sensors distributed over some area that form an ad-hoc network. Each sensor is equipped with some limited memory and processing capabilities, multiple sensing modalities, and communication capabilities. Previous work in sensor networks has concentrated on routing protocols for sensor networks. Often the network topology is unknown and the network has to discover the best route for a packet. Optimization criteria include shortest path to destination, minimum power utilization, maximum minimum residual power in the network, etc.

In this paper we focus on a reactive task in sensor networks: guiding the movement of a user equipped with a node that can talk to the field of sensors across the field. We also discuss how sensor networks can serve as adaptive distributed repositories of information.

More specifically, we build on important previous work by [3], [11], [32], [4], [20], [19] and examine in more detail reactive sensors that can adapt to their environment by capturing a danger level map and distributing this map across the network. We represent the danger detected by the sensors as “obstacles” in the network and compute the artificial potential field that corresponds to the current state. We then develop a distributed protocol that combines this artificial potential field with information about the direction and goal of the moving object and guarantees the best safest path to the goal. By safest path we mean the path with the largest clearance of the danger zones. We also develop a protocol for distributing the information in the sensor network, such as the danger map and shortest paths. We then show how sensors equipped with limited memory can cooperate to hold and retrieve information about the network. Finally, we discuss an implementation of our protocols on a real sensor network consisting of 50 Mote sensors [10], [9], [31] and present our experimental data.

II. Related Work

We are inspired by previous work in sensor networks [27], [1], [7], [6], [2], [23], [29], [18], [16], ad-hoc networks [12], [8], [24], [25], [26], [28], [5], [22], [21], and robotics [14]. Our experimental work is done with the Mote hardware [10], [9], [31].

In [3], Cedar and Estrin propose an adaptive self-configuring sensor network topology. The nodes self-configure to establish a topology that provides communication and coverage. Each node can decide whether to join the network by considering the network condition, the loss rate, the connectivity, etc. If a node can contribute to the network, it will join the network, otherwise, it will sleep or adjust its duty cycle. In our paper, we concentrate on the application requirements, such as information distribution in the navigation guiding application.

In Intanagonwiwat et al.'s direct diffusion [11] approach, data generated by sensor nodes is named by attribute-value pairs. A node requests data by sending interests for named data; the interests will be propagated within the network to find the source of the related data. The direct diffusion method is used to reinforce the best path from the source to the sink. We propose to actively disseminate the information in the network, and consider the sensor network as an information base.

Ye et al. [32] propose a MAC protocol that aims at energy conservation and self-configuration. They use periodic listen and sleep to conserve energy. In order to keep the nodes communicating with each other, they must be synchronized. Sohrabi et al. [30] propose several algorithms for the self-organization of a sensor network, which includes the self-organizing medium access control, energy-efficient routing, and formation of ad-hoc subnetworks for cooperative signal processing functions.

In the broad area of ad-hoc networks, many routing protocols have been proposed, including [12], [8], [24], [25], [26], [28], [5], [22], [21], [17].

The application developed in this paper uses techniques from robotics, where a key problem is how to plan the motion of moving robots. A good overview of motion planning in robotics is given by [14]. [15] proposes a robot motion planner that rasterizes configuration space obstacles into a series of bitmap slices, and then use dynamic programming to compute the distance from each point to the goal and the paths in this space. This method guarantees that the robot finds the best path to the goal. [13] discusses the use of artificial potential field for robot motion planning. A robot moving in accordance to the potential will never hit obstacles, but it may get stuck in local minima. We combine the two methods to find the best path to the goal, which is safe and short, and modify them to adapted them to the distributed nature of sensor networks.

III. A Distributed Algorithm for Guiding the Navigation of a User

Sensors detect information about the area they cover. They can store this information locally or forward it to a base station for further analysis and use. Sensors can also use communication to integrate their sensed values with the rest of the sensor landscape. In this section we explore using sensor networks as distributed information repositories. We describe a method to distribute the information about the environment redundantly across the entire network. Users of the network (people, robots, unmanned planes, etc.) can use this information as they traverse the network. We illustrate this property of a reactive sensor network in the context of a guiding task, where a moving object is guided across the network along a safe path, away from the type of danger that can be detected by the sensors.

The guiding application can be formulated as a robotics motion planning problem in the presence of obstacles. The dangerous areas of the sensor network are represented as obstacles. Danger may include excessive heat (volcanoes, fire, etc), people, etc. We assume that each sensor can sense the presence or absence of such types of danger. A danger configuration protocol run across all the nodes of the network creates the danger map. We do not envision that the network will create an accurate geometric map, distributed across all the nodes. Instead, we wish for the nodes in the network to provide some information about how far from danger each node is. If the sensors are uniformly distributed, *the smallest number of communication hops to a sensor that triggers "yes" to danger* is a measure of the distance to danger. The goal is to find a path for the moving object that avoids the dangerous areas. We envision having the user ask the network regularly for where to go next. The nodes within broadcasting range from the user supply the next best step.

There are numerous solutions to motion planning in the presence of obstacles and uncertainty. For a good survey of the techniques see [14]. We seek a solution that lends itself naturally to the discrete nature of sensor networks. In [15], Donald et al. describe an optimal solution for motion planning when the map of the world is given. The first step of the solution is to rasterize the configuration space obstacles into a series of bitmap slices. Dynamic programming is then used to calculate the optimal path in this space. Although this method can not be applied directly, it can be adapted for sensor networks. Although the map is not immediately available, the motion planning algorithm fits a sensor network well in two ways. First, the sensors can be regarded as the bitmap pixels. Second, the dynamic programming component of the algorithm can be implemented by using the sensor communications.

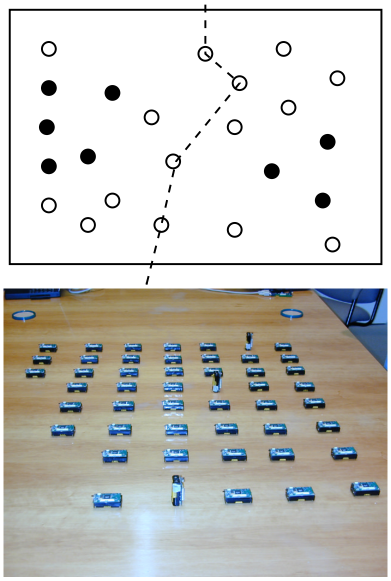


Fig. 1. The top figure shows a typical setup for the navigation guiding task. The solid black circles correspond to sensors whose sensed value is "danger". The white circles correspond to sensors that do not sense danger. The dashed line shows the guiding path across the area covered by the sensor network. Note that the path travels from sensor to sensor and preserves a maximal distance from the danger areas, while progressing to the exit area. The bottom picture shows some Mote sensors used for our experiments. The three sensors placed in the upright position denote 2 obstacles (that is, danger areas) and one goal.

In order to supply obstacle information to the planning algorithm we use artificial potential fields. In an artificial potential field, objects move under the actuation of artificial forces. Usually, the goal generates an attractive potential which pulls the object to the goal. The obstacles generate a repulsive potential which push the object away from the goal. The (negated) gradient of the total potential is the artificial force acting on the object. The

Algorithm 1 The potential field computation protocol.

```
for all sensors $s_i$ in the network do
    $pot_i = 0$
    if sensed-value = danger then
        $hops_i = 0$
        Broadcast message $(i, hops = 0)$
    if receive($j, hops$) then
        $hops_j$ = min($hops_j, hops + 1$)
        Broadcast message $(j, hops_j)$
    for all received $j$ do
        Compute the potential $pot^j$ of $j$ using $pot^j = \frac{1}{{hops_j}^2}$
        Compute the potential at $s_i$ using all $pot^j$, $pot_i = pot_i + pot^j$
```

Algorithm 2 The safest path to goal computation protocol.

```
Let $G$ be a goal sensor
$G$ broadcasts $msg = (G_{id}, my_{id}(G), hops = 0, potential = 0)$
for all sensors $s_i$ do
    Initially $hops_g = \infty$
    if receive($(g, k, hops, potential)$) then
        Compute the potential integration from the goal to here:
        $P_g = min(P_g, potential + pot_i)$
        $hops_g = min(hops_g, hops + 1)$
        if $P_g == P_g + pot_i$ and $hops_g == hops + 1$ then
            $prior_g = k$
            Broadcast $(G_{id}, my_{id}(s_i), hops_g, P_g)$
```

direction of this force is the current best direction of motion [14].

The "obstacles" (recall they correspond to the dangerous areas) will have repulsing values and the goal will have an attracting value according to some metric (see Figure 1(Top)). Algorithm 1 shows the potential field protocol. The potential field is computed in the following way. Each node whose sensor triggers "danger" diffuses the information about the danger to its neighbors in a message that includes its source node id, the potential value, and the number of hops from the source of the message to the current node. When a node receives multiple messages from the same source node, it keeps only the message with the smallest number of hops. (The message with the least hops is kept because that message is likely to travel along the shortest path.) The current node computes the new potential value from this source node. The node then broadcasts a message with its potential value and number of hops to its neighbors.

After this configuration procedure, nodes may have several potentials from multiple sources. To compute its current danger level information, each node adds all the potentials.

Note that the potential field protocol provides a distributed repository of information about the area covered by the sensor network. It can be run in an initialization phase, continuously, or intermittently. The sensor network can self-reconfigure adaptively to the current landscape. It updates its distributed information content by running the potential field computation protocol regularly. In this way, the network can adapt to sensor failure, to the addition of new nodes into the network and to dynamic danger sources that can move across the network.

The potential field information stored at each node can be used to guide an object equipped with a sensor that can talk to the network in an on-line fashion. The safest path to the goal can be computed using Algorithm 2. The goal node initiates a dynamic programming computation of this path using broadcasting. The goal node broadcasts a message with the danger degree of the path, which is zero for the goal. When a sensor node receives a message, it adds its own potential value to the potential value provided in the message, and broadcasts a message updated with this new potential to its neighbors. If the node receives multiple messages, it selects the message with the smallest potential (corresponding to the least danger) and remembers the sender of the message.

Algorithm 3 The navigation guiding protocol.

```
if $s_i$ is a user sensor then
  while Not at the goal $G$ do
    Broadcast inquiry message $(G_{id})$
    for all received messages $m = (G_{id}, my_{id}(s_k), hops, potential, prior)$ do
      Choose the message $m$ with minimal potential than minimal hops
      Let $my_{id}(s_k)$ be the id for the sender of this message
      Move toward $my_{id}(s_k)$ and $prior$
if $s_i$ is an information sensor then
  if receive $(G_{id})$ inquiry message then
    Reply with $(G_{id}, my_{id}(s_i), hops_g, P_g, prior_g)$
```

A user of the sensor network can rely on the information computed using Algorithms 1 and 2 to get continuous feedback from the network on how to traverse the area. Algorithm 3 shows the navigation guiding protocol. The user asks the network for where to go next. The neighboring nodes reply with their current values. The user's sensor chooses the best possibility from the returned values. Note that this algorithm requires the "integrated" potential computed by Algorithms 1 and 2 in order to avoid getting stuck in local minima.

We can prove that our protocols will correctly determine the safest path to the goal without getting stuck in the local minima that are often an issue with artificial potential fields methods.

Theorem 1: Algorithm 3 will always give the user sensor a path to the goal.

Proof: In Algorithm 2, the $prior$ link of a node points to a node that has $potential$ value less than that of the current node. So for each node other than the goal, there must be a neighboring node that has a smaller $potential$ value. This proves that there is no local minima in the network.

The user's sensor can always find a node among its neighbors that leads to a smaller $potential$ value. If the process continues, the node will end up with the goal that has the smallest $potential$ value 0. Therefore, Algorithm 3 can always give the user sensor a path to the goal. ■

Algorithms 1 and 2 ask each sensor to broadcast upon receiving a message with fewer hops to the dangerous area or a smaller potential integration to the goal. Many broadcasts may not be necessary since only the message with the least hops to the danger or the minimal potential integration to the goal is useful. To reduce the message broadcasts, we can let each sensor wait for some time before it broadcasts. The waiting time for sensor s_i is proportional to one unit in Algorithm 1 and the value pot_i in Algorithm 2. We can prove that the number of message broadcasts for each sensor is 1 in each algorithm. The detailed analysis can be found in [18].

IV. Using Sensor Networks to Distribute Information

Section III provided a simple example for how to use a sensor network as a distributed information repository about the environment in the context of a navigation guiding application. In this section we examine in more detail how to use a sensor network as a distributed information repository.

Consider again the navigation guiding application formulated as a motion planning problem. Suppose multiple goal are installed in the network. It is possible that each sensor has enough memory to store all the pertinent information about these goals. However, the current sensor hardware has very limited memory which restricts the amount of information that can be stored.

We argue that sensors do not have to store all the information about the goals. Instead, all the necessary information should be stored somewhere, but not everywhere, in the network. The important thing is being able to retrieve the information any time it is needed.

Many sensors can cooperate to store information by having each sensor locally store only part of it. If the density of the network is such that multiple sensors cover the same area, the local information is the same for the sensors in some neighborhood. Thus, it does not matter who stores what. We propose that when a node receives a piece of information about the network, it randomly chooses to either keep it or to discard the information. To make this work, we must address (1) how to quantify the probability of discarding the information with respect to the information amount, the message size, and the density of the nodes; (2) how to retrieve the information from this sensor proximity, and (3) what are the trade-offs between the memory utilization and broadcasting amount.

In order to address the information storage question, consider the proximity area S covered by a group of sensors. All local (environmental) information about S is the same for all these sensors. To use Algorithm 3, at least one of the sensors in S must store information about the goals. Let $\lambda \cdot S$ be the number of sensors in the area where λ is the density of the sensor distribution and S is the area of the field in question. Suppose each sensor has memory m. Then $m\lambda S$ is the total memory across all sensors. Let the amount of information to be recorded be $\sum m_i$ where m_i is the size of information i. If $m\lambda S \geq \sum m_i$, then it is possible that in the proximity area S, all the required information can be found locally using Algorithm 3. To achieve this information distribution when the amount of information is too large for a node's memory (that is, $m < \sum m_i$), we can use a random, independent and distributed method to store the information on each sensor. Each sensor node randomly keeps a piece of information with probability $p = \frac{m}{\sum m_i}$. When it receives a piece of information, the probability that the information can be found in this area is $1 - (1-p)^{\lambda S}$ (see Figure 2). Currently, we are also exploring some other approaches to cooperative caching data among proximity sensors.

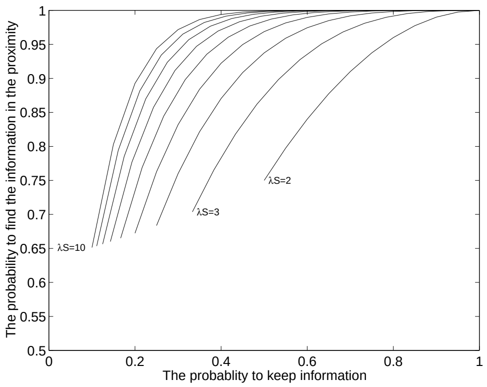


Fig. 2. This figure shows the probability (Y-axis) that a piece of information can be found in S, some neighborhood of a sensor. The X-axis is the probability that the sensor keeps a piece of information. We plot for various numbers of sensors in the area from $\lambda S = 2$ to $\lambda S = 10$ where λS is the number of sensors in that area. As the number of the sensors increases, the probability to find some information in that area is close to 1 even though the probability that a sensor keeps the information is small.

Algorithm 4 summarizes the protocol for locating a piece of information in a sensor network. If the information cannot be found in the proximity area S, the sensor must try to retrieve information beyond the area in the sensor network. Intuitively, the request for information is broadcast to all the sensors in the area S. The sensors who have the information reply to the request. If there is no reply in the transmission range, the request must be broadcast again to a larger area, by making larger and larger concentric communication bands. More specifically, the user sends out the information request; the sensors in the broadcast range hear the request and reply if they have the information. Otherwise no sensor replies to the request. After some period of silence with no reply (Δ, the transmission time for the request and reply message), the user's requesting node sends out an information request for two hops. Each node receiving this message will broadcast the request out. If the information is found, it is sent back to the requesting node. Otherwise after some time of silence time with no reply (2Δ here), the requesting node sends out an information request for three hops, and so on, until finally the information gets back to the requesting node.

Algorithm 4 Sensor information query algorithm

```
if I am the query sensor $s$ then
    $depth1 = depth2 = 1$
    while true do
        Broadcast $(s, query, depth1, depth2)$
        Wait for time $depth1 * \Delta$
        if some reply arrives then
            stop
        else
            $depth1 + +$, $depth2 + +$
if I am not the query sensor then
    receive$(s, query, depth1, depth2)$
    if I have already received a message with prefix $(s, query, depth1, *)$ then
        discard the message
    if I have the information to $query$ then
        send the information to $s$, stop
    if $depth2 - 1 == 0$ then
        stop
    else
        broadcast$(s, query, depth1, depth2 - 1)$
```

V. Experiments

We have implemented the algorithms described in Section III using the Mote MOT300 sensors [10], [9], [31].

A. The Mote Hardware

The Mote hardware configuration includes a microcontroller and devices such as LEDs, a low-power radio transceiver, a photo-sensor, a serial port, and a co-processor unit (see Figure 1(Bottom)). We have augmented this platform with a power sensor and a sound sensor. We are currently using the second generation MICA Motes MOT300 with an Atmel ATMEGA103 processor (4MHz, 128KB Instruction Memory, 4KB RAM), and 4Mbit flash. The radio communication consists of an RF Monolithics 916.50 MHz transceiver (TR1000), antenna, and some components to adjust the physical layer characteristics such as signal strength and sensitivity.

The operating system support for the Motes is provided by TinyOS, an event-based operating system. TinyOS consists of a very small scheduler and components that abstract the Mote functions: clocks for timing, RFM for radio raw data communication, Radio Byte, Radio Packet and Active Message for different abstractions of the mote communication similar to the network stack. Each component is triggered by explicit commands from the upper level, or by events from other components, such as clock interrupts, message arrivals, message send completions, etc.

B. Correctness Validation

We have implemented the protocols in Algorithms 1, 2, and 3. In our experiment, we asked both the goals and the obstacles to generate the potential field and propagate it to the entire network periodically. So the goals and the obstacles can be added to the network at any time.

The goal is represented with one Mote. The obstacles are represented by one Mote each. The user or robot traversing the sensor network is also represented by one Mote.

A first experiment was designed to show empirically that the protocols work and are correct. In this first experiment we used a grid of 12 first generation Motes. All neighbors are within communication range. The application is run by iterating a request for the next step by the robot, a response by the network, and a move to the direction of the network response. To implement this last part we assume that the nodes know their location and that it can be transmitted to the moving object/robot. This can be done by augmenting Motes with a GPS location, or via triangulation. Since we have not done this augmentation of the hardware yet, we simulate location knowledge by placing the Motes in a grid pattern and supplying coordinates. The potential field and goal path computations are run by the network continuously.

When an obstacle or goal broadcasts, the receiving network node checks its list of known goals, and replaces the old data with the new broadcast if the new broadcast has a lower hop count. If the obstacle or goal is unknown, then an entry is created, and it erases the oldest entry if there is no room.

When a node receives a broadcast, it degrades the value of the broadcast based either on a linear function on the number of hops (for goals) or by the number of hops squared (for obstacles). If the new value is not below a cutoff threshold, the packet is transmitted to its neighbors.

When a robot requests potential estimates, all nodes that can hear it respond. The robot chooses the node with the lowest value (that is lower than the value of the current node). The robot moves toward this node.

This first experiment proved that a robot with a sensor node actually went around the obstacles and got to the goal, via the correct path. We also observed that the network adapted to the introduction of new obstacle nodes quickly and robustly.

When a new obstacle is inserted in the network, the obstacle starts broadcasting its danger information which affects the information held by each node. At this point Algorithms 1 and 2 cause the local information to change. We call the total time for the network to identify the new distances from danger and to the goal for each node the *time for the network to stabilize.* In other words, the time for the network to stabilize is the information propagation time in the network, which depends on the maximal hops from the goals or the obstacles to any node in the network. In our experiment, the maximal hop count was eight, so the network stabilization time was $8 * 0.07 = 0.56$ seconds where 0.07 is the measured packet transmission time for one hop. When an obstacle is added to the system online, it takes an identical amount of time to diffuse the information to the whole network. Since each node only propagates the message with the least hops, if we assume each node gets the least hop messages earlier than the messages with more hops, the number of messages each node transmits is approximately the number of obstacles and goals.

C. *Measuring Adaptation*

We have implemented the protocols in Algorithms 1, 2, and 3 on the second generation Motes MOT300. In this second experiment, we used a 50 Mote MOT300 testbed. We arranged the nodes in the given topology and gave each node position information (which could be obtained using a GPS extension of the hardware.) We ran a suite of different network topologies and measured the network stabilization time when obstacles and goals are injected on-line in the network. Tables I and II summarize our data.

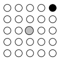
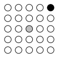
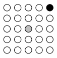
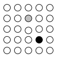
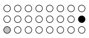
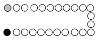
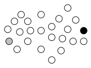
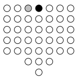
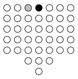
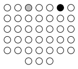
The layouts include grids with various numbers of Motes, randomly dispersed Motes, and circles. In each network we inserted obstacle sensors (assumed to have detected danger) and goal sensors. The focus of these experiments has been to determine on how quickly the network responds to the environmental change, specifically new danger sources and goal changes.

We ran all the experiments on a large table in our lab, as shown in Figure 1(Bottom). For each experiment, we set the transmission range to be very small (9”). In all these second round experiments we focused on the network as a whole and did not use a base station (thus, we did not collect data in a central place.) Because Mote sensors do not provide an easy way to synchronize clocks and to do local time measurements, we used the following procedure to capture timing data. Each time a node sent a broadcast, one of its LEDs flashed. We recorded the experiment with a Sony video camera at a rate of 30 frames per second. We then analyzed the resulting video to capture the timing measurements—which gave us a resolution of $1/30^{th}$ of a second. We looked at the video sequence frame by frame and kept track of when and which LED triggered. Since the overall timings are on the order of seconds, we believe our methodology is accurate enough.

We analyzed four metrics for each experiment: the time for the danger information to propagate from the danger/obstacle sensor to the whole network, the time for all the nodes in the network

TABLE I

THE DATA THAT SUMMARIZES TIMING MEASUREMENTS FOR SEVERAL EXPERIMENTS WITH A SENSOR NETWORK CONSISTING OF MOTE SENSORS. ALL NETWORK TOPOLOGIES ARE SUMMARIZED AS GEOMETRIC ICONS AND ALL MEASUREMENTS ARE IN SECONDS.

Exp. Config.	danger propagation	Shortest distance	goal propagation	safest path
	0.23	1.13	0.17	9.23
	0.20	2.17	0.13	4.23
	1.16	3.13	2.13	10.03
	0.10	3.10	0.10	1.07
	0.23	1.33	0.13	1.07
	1.43	2.10	0.17	2.17
	1.10	27.17	4.27	9.10
	7.27	10.13	0.04	33.80
	0.17	6.17	0.04	19.37
	4.20	9.10	1.37	22.63

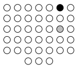
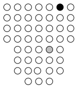
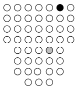
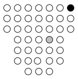
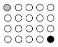
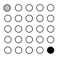
For each experiment, the goal is at the black disk and the danger is at the shaded disk.

to obtain their shortest distance to the dangerous areas, the time for the goal information to propagate to the whole network, and the time for all the nodes in the network to obtain their safest path to the goal. Tables I and II show the time distribution of the four metrics.

We also did experiments to measure the response time of the sensor network after changing the topology of the network. Starting from the initial topology (No. 0), we changed the locations of the obstacles and recorded the response time in each experiment. Table III shows the data of 15 consecutive experiments. The response time is defined as the period from the time when the topology change occurs to the time when the robot finds the path to the goal. The route cache on each mote is

TABLE II

TABLE I CONTINUED

Exp. Config.	danger propagation	Shortest distance	goal propagation	safest path
	1.23	22.33	2.27	19.27
	5.20	20.30	1.17	9.40
	1.30	16.23	1.10	2.17
	9.37	17.37	0.33	8.43
	1.10	1.10	3.13	5.10
	7.27	12.23	0.20	8.40
	4.20	4.20	0.30	12.80
	0.20	7.17	1.13	3.13
	1.20	20.23	2.13	3.13

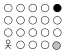
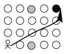
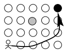
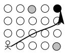
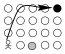
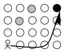
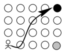
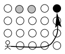
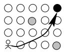
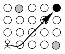
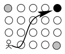
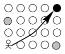
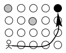
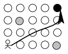
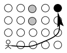
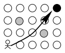
refreshed every 10 seconds. The route information incurred by the topology change is updated only after flushing the cache. Without taking into account the information propagation time, the average response time is 5 seconds. The information propagation adds extra time after the cache is flushed.

Several interesting aspects of these experiments can be observed. The time for network stabilization (that is, the time for all the nodes to get the shortest distance to the danger source and the time for all the nodes to get the safest path to the goal) takes much longer than we expected. In our algorithms we made two typical assumptions: (1) a node broadcasts the message received immediately and (2) each node gets the packet traveling through the shortest path. We observed that on the hardware testbed neither of these assumptions held. The network stabilization takes a long time because of network congestion and transitory link status. Often, nodes seemingly out of range hear each other for brief moments of time.

Our observations of these hardware experiments have taught us some lessons about the assumptions used by most distributed sensor network protocols examined theoretically or in simulation.

1) *Data loss.* Data loss is not rare in sensor net-

TABLE III
The data of the response time for several experiments with a sensor network consisting of Mote sensors. All network topologies are summarized as geometric icons and all measurements are in seconds.

Exp. No.	Exp. Config.	Response Time	Exp. No.	Exp. Config.	Response Time
0		0	8		4.43
1		3.63	9		5.33
2		6.83	10		4.43
3		3.60	11		9.63
4		4.93	12		6.27
5		2.13	13		4.50
5		1.73	14		4.87
7		2.23	15		6.70

For each experiment, the goal is at the black disk and the danger is at the shaded disk. The black line and arrow signify the safe path found in each network topology.

works. This is due to network congestion, transmission interference, and garbled messages.

2) *Asymmetric connection.* We observed that the transmission range in one direction may be quite different from that in the opposite direction. Thus, the assumption that if a node receives a packet from another node, it can send back a packet is too idealistic. In routing algorithm design, the existence of a route that can carry a packet from the source to a node does not guarantee a reverse route from that node to the source.
3) *Congestion.* Network congestion is very likely when the message rate is high. This is aggravated when the nodes in proximity of each other try to send packets at the same time. For a sensor network, because of its small memory and simplified protocol stack, congestion is a big problem.
4) *Other unpredictable network conditions.* In our sensor networks nodes that should be several hops away from each other occasionally come in direct communication range. We expect many transitory links (on and off) in a unstable network due to the impact of the unpredictable conditions.

We conclude that the uncertainty introduced by data loss, asymmetry, congestion, and transient links is fundamental in sensor networks. We need new models, algorithms, and simulations that take this kind of uncertainty into account. Guided by these lessons, we are currently conducting experiments to characterize better the likelihood of these uncertainty conditions.

VI. Conclusion

We have discussed self-reconfiguring sensor networks that can react to their environment and adapt to changes. We have described a novel application: using the sensor network to guide the movement of a user (human or robot, equipped with a sensor that can talk to the network) across the area of the network along a safest path. Safety is measured as the distance to the sensors that detect danger. We described several protocols for solving this problem. Our protocols implement a distributed repository of information that can be stored and retrieved efficiently when needed. We have used ideas from robotics to provide a correct solution to the navigation guiding task. We have implemented these protocols on a network of 50 Mote sensors. The key metric used in our experimental evaluations is the time it takes the network to adapt to a new situation (detecting a moving vehicle, detecting a new obstacle, adding a new sensor in the network, removing a sensor from the network, etc.). Our experimental work has taught us a number of lessons about some typical assumptions for designing protocols and have pointed out some important new directions of research.

Acknowledgments

We are very grateful to Deborah Estrin for introducing us to the Motes hardware. We thank the TinyOS group for providing great support. We are also grateful to Ron Peterson for his invaluable help and discussions.

This work bas been supported in part by Department of Defense contract MURI F49620-97-1-0382 and DARPA contract F30602-98-2-0107, ONR grant N00014-01-1-0675, DARPA contract F30602-00-2-0585, NSF awards 0225446, E1A-0202789, I1S-9912193, Honda corporation, and the Sloan foundation; we are grateful for this support.

References

[1] Jon Agre and Loren Clare. An integrated architeture for cooperative sensing networks. *Computer*, pages 106 – 108, May 2000.

[2] B. Badrinath, M. Srivastava, K. Mills, J. Scholtz, and K. Sollins, Eds. Special issue on smart spaces and environments. *IEEE Personal Communications*, Oct. 2000.

[3] Alberto Cerpa and Deborah Estrin. Ascent: Adaptive self-configuring sensor networks topologies. In *INFOCOM*, New York, NY, June 2002.

[4] M. Chu, H. Haussecker, and F. Zhao. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. *International Journal of High Performance Computing Applications*, 2002.

[5] Samir Das, Charles Perkins, and Elizabeth Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *INFOCOM*, 2000.

[6] Deborah Estrin, Ramesh Govindan, and John Heidemann. Embedding the internet. *Communications of ACM*, 43(5):39–41, May 2000.

[7] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *ACM MobiCom 99*, Seattle, USA, August 1999.

[8] Z. J. Haas. A new routing protocol for the reconfigurable wireless network. In *Proceedings of the 1997 IEEE 6th International Conference on Universal Personal Communications, ICUPC'97*, pages 562 –566, San Diego, CA, October 1997.

[9] Jason Hill, Philip Bounadonna, and David Culler. Active message communication for tiny network sensors. In *INFOCOM*, 2001.

[10] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for network sensors. In *ASPLOS*, 2000.

[11] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. of the Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM 2000)*, Boston, Massachusetts, August 2000.

[12] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad-hoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, pages 153 –181. Kluwer Academic Publishers, 1996.

[13] D. E. Koditschek. Planning and control via potential fuctions. *Robotics Review I (Lozano-Perez and Khatib, editors)*, pages 349–367, 1989.

[14] J.-C Latombe. *Robot Motion Planning*. Kluwer, New York, 1992.

[15] J. Lengyel, M. Reichert, B. Donald, and D. Greenberg. Real-time robot motion planning using rasterizing computer graphics hardware. In *Proc. SIGGRAPH*, pages 327–336, Dallas, TX, 1990.

[16] Qun Li, Javed Aslam, and Daniela Rus. Hierarchical power-aware routing in sensor networks. In *DIMACS Workshop on Pervasive Networking*, Rutgers University, May 2001.

[17] Qun Li, Javed Aslam, and Daniela Rus. Online power-aware routing in wireless ad-hoc networks. In *MOBICOM*, pages 97–107, Rome, July 2001.

[18] Qun Li, Javed Aslam, and Daniela Rus. Distributed energy-conserving routing protocols for sensor networks. In *Hawaii International Conference on System Science*, Hawaii, Jan. 2003.

[19] Qun Li, Ron Peterson, Michael DeRosa, and Daniela Ru. Reactive behavior in self-reconfiguring sensor network. *ACM Mobile Computing and Communications Review*, 2002.

[20] Qun Li, Ron Peterson, Michael DeRosa, and Daniela Rus. Reactive behavior in self-reconfiguring sensor networks. In *ACM Mobicom Poster*, Atlanta, GA, Sep. 2002.

[21] Qun Li and Daniela Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *MOBICOM*, pages 44–55, Boston, August 2000.

[22] Qun Li and Daniela Rus. Communication in disconnected ad-hoc networks using message relay. *Journal of Parallel and Distributed Computing*, to appear.

[23] Seapahn Meguerdichian, Farinaz Koushanfar, Gang Qu, and Miodrag Potkonjak. Exposure in wireless ad hoc sensor networks. In *MOBICOM*, pages 139–150, Rome, July 2001.

[24] S. Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM/Baltzer Journal on Mobile Networks and Applications*, MANET(1,2):183 –197, October 1996.

[25] V. Park and M. S. Corson. A highly adaptive distributed algorithm for mobile wireless networks. In *Proceedings of INFOCOM'97*, Kobe, Japan, April 1997.

[26] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *Computer Communication review*, 24(4):234 –244, October 1994.

[27] G. J. Pottie and W. J. Kaiser. Wireless integrated newtork sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[28] Elizabeth Royer and C-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. In *IEEE Personal Communication*, volume 6, pages 46 – 55, April 1999.

[29] Gabriel T. Sibley, Mohammad H. Rahimi, and Gaurav S. Sukhatme. Robomote: A tiny mobile robot platform for large-scale sensor networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[30] Katayoun Sohrabi, Hay Gao, Vishal Ailawadhi, and Gregory J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, pages 16–27, Oct. 2000.

[31] Alec Woo and David Culler. A transmission control scheme for media access in sensor networks. In *Mobicom*, 2001.

[32] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient mac protocol for wireless sensor networks. In *INFOCOM*, New York, NY, June 2002.