Navigation Protocols in Sensor Networks

QUN LI
College of William and Mary
and
DANIELA RUS
MIT

We develop distributed algorithms for adaptive sensor networks that respond to directing a target through a region of space. We model this problem as an online distributed motion planning problem. Each sensor node senses values in its perception space and has the ability to trigger exceptions events we call "danger" and model as "obstacles". The danger/obstacle landscape changes over time. We present algorithms for computing distributed maps in perception space and for using these maps to compute adaptive paths for a mobile node that can interact with the sensor network. We give the analysis to the protocol and report on hardware experiments using a physical sensor network consisting of Mote sensors. We also show how to reduce searching space and communication cost using Voronoi diagram.

Categories and Subject Descriptors: C.2.1 [**Computer Communications Network**]: Network Architecture and Design—*Wireless communication*

General Terms: Algorithms, Design, Experimentation, Measurement, Performance

Additional Key Words and Phrases: Sensor networks, potential field, navigation, motes, robotics

1. 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 in order 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, and so forth.

In this article, 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. We model the user guidance problem as a robot motion planning problem and use the inherent feature of the sensor network to compute the robot navigation path in a distributed way. Our article contributes (1) a mobile application for sensor network; (2) an implementation and evaluation on a physical sensor network; (3) a distance computation method that does not use node positions; (4) performance analysis and hardware experimentation; (5) variation of the navigation protocols that reduces the searching space using Voronoi diagrams.

More specifically, we build on important previous work by Meguerdichian et al. [2001]; Intanagonwiwat et al. [2000]; Ye et al. [2002]; and Ganesan et al. [2002] and examine reactive sensors that can adapt to their environment by capturing a danger level map and distributing this map across the network. We consider special events called "danger" to be values detected by the sensors (e.g. temperature, biochemical agents, etc.) that are above a threshold. We consider the areas of sensor network that have triggered special events to be "obstacles". We compute the artificial potential field that corresponds to the current state in the perception space of the sensor network relative to these obstacles. 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. Finally, we discuss an implementation of our protocols on a real sensor network consisting of 50 Mote sensors [Hill et al. 2000] and present our experimental data.

2. RELATED WORK

We are inspired by previous work in sensor networks [Estrin et al. 2000] and robotics [Latombe 1992]. Our experimental work is done with the Mote

hardware [Hill et al. 2000]. Related papers, Batalin and Sukhatme [2003, 2004] address the problem of coverage and exploration of an unknown dynamic environment using a mobile robot and beacons. The beacons form a communication network that is used as a support infrastructure to aid the exploration of the mobile robot.

Our sensor network algorithms rely on scalable high-performance routing protocols such as Intanagonwiwat et al.'s [2002] direct diffusion. In this work, 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. [2002] proposed (TTDD), a Two-Tier Data Dissemination approach that provides scalable and efficient data delivery to multiple mobile sinks. The data source proactively builds a grid structure and the sink requests the data from the nodes on the grid. This approach can be applied to the general problem of sensor network data dissemination. In our article, we consider a specific application, navigation problem.

Meguerdichian et al. [2001] and Veltri et al. [2003] considered the minimal and maximal exposure path problem in a network. We consider a seemingly similar problem. We are concerned about the dangerous areas rather than the coverage of an individual sensor. Instead of calculating the information about the worst case exposure-based coverage caused by the deployment of a sensor network, we use the sensor network to compute a path that can navigate a user to the goal by avoiding the dangerous area. Furthermore, we use distributed algorithms to disseminate the data in the sensor network.

There have been many studies conducted on mote sensor networks. An empirical study on networks composed of over 150 Motes was conducted in Ganesan et al. [2002]. The paper presents the data collected in different layers and reveals that even a simple protocol can exhibit large complexity in the Mote network. Some of the observations from our experiments show the same behaviors in many scenarios. Other studies on the performance of the sensor network communication can be found in Zhao and Govindan [2003] and Woo et al. [2003]. Wan et al. [2002, 2003] proposed Pump Slowly, Fetch Quickly (PSFQ), a reliable transport protocol in wireless sensor networks. The key concept of this protocol was to distribute data from a source by pumping the data forward at a slow speed, then allowing the nodes that suffer data loss to fetch the missing data aggressively from their neighbors. Experimental verification of this algorithm was carried out on a mote network. This article addresses some problems that we encountered in our system implementation.

Zhao et al. [2002] proposed algorithms which can be used in tracking applications. An information utility measure is used to select which sensors to query and to dynamically guide data routing. The method maximizes the information gain and minimizes the latency and the bandwidth consumption. Huang et al. [2003] proposed spatiotemporal multicast for tracking applications in which a message can be delivered to a mobile zone for information collection or sensor

wake-up. Yan et al. [2003] proposed sensing schedule protocols for field coverage in surveillance application for sensor networks. Fang et al. [2004] designed algorithms for locating and bypassing routing holes in sensor networks.

We use the number of hops to evaluate the distance between sensors without relying on location information. However, one of our extensions does depend on previous work on localization such as Moore et al. [2004]; Capkun et al. [2002]; Savvides et al. [2001]; Niculescu and Badrinath [2003]; Sundaram and Ramanathan [2002]; Cheng et al. [2004].

Capkun et al. [2002] proposed a distributed, GPS-free positioning algorithm which uses the distances between the nodes to build a relative coordinate system in which the node positions are computed. Savvides et al. [2001] proposed an AHLoS system which used iterative multilateration. The approach relies on a small set of nodes initially configured as beacons to estimate node locations. Niculescu and Badrinath [2003] proposed to using the capability of an angle of arrival (the direction from which a signal is received) to determine the orientation and position of any node in an ad hoc network with the aid of some landmark nodes. Sundaram and Ramanathan [2002] proposed using the neighborhood relationships gathered by each user through message exchanges over a wireless ad hoc network to estimate the locations of hosts. It improved the accuracy of location estimation by incorporating nonneighbor constraints. Cheng et al. [2004] used the time difference of arrival of RF signals to estimate the sensor location. Their paper gave a very good statistical analysis of the performance of the proposed scheme. Moore et al. [2004] proposed a fully distributed localization method guaranteed to compute correct locations that supports mobility.

The application developed in this article 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 Latombe [1992]. Lengyel et al. [1990] proposed a robot motion planner that rasterizes configuration space obstacles into a series of bitmap slices and then uses 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. Koditschek [1989] discusses the use of an artificial potential field for robot motion planning. A robot moving in accordance with the potential will never hit obstacles, but it may get stuck in local minima. We combine these two methods to find the best path to the goal which is safe and short and then modify them to exploit the distributed nature of sensor networks.

3. A DISTRIBUTED ALGORITHM FOR GUIDING 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, and so on. We assume that each sensor can sense the presence or absence of such types of danger, for example, with temperature sensors and biochemical sensors. A danger configuration protocol runs across all the nodes of the network and 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 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 Latombe [1992]. We seek a solution that lends itself naturally to the discrete nature of sensor networks. Lengyel et al. [1990] 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. Though 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.

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 direction of this force is the current best direction of motion [Latombe 1992].

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 left). Algorithm 1 shows the potential field protocol. The potential field is computed in the following way. Each node whose sensor triggers danger[1] diffuses the information about its knowledge of danger to its neighbors

---

[1]The possibility to identify obstacles is dependent on the sensing quality of the sensors. Our assumption is that the sensors have this capability and this is not the concern of our algorithm, although it is a very important factor in applying our algorithm in real applications. In our experiments, a
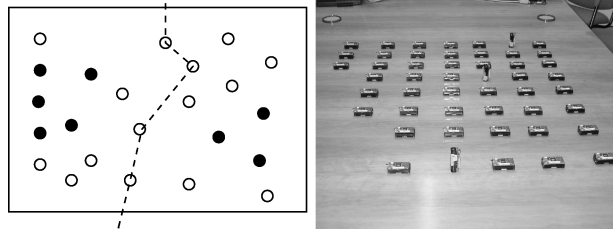
Fig. 1.   The left figure shows a typical setup for the navigation guiding task. The solid black circles correspond to sensors whose sensed value is danger. The white circles correspond to sensors that do not sense danger. The dashed line shows the guiding path across the area covered by the sensor network. 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 right picture shows some Mote sensors used for our experiments. The three sensors placed in the upright position denote two obstacles (i.e., danger areas) and one goal.

Algorithm 1. The potential field computation protocol.

1: **for** all sensors s_i in the network **do**
2:     $pot_i = 0$, $hops_j = \infty$ for any danger j
3:     **if** *sensed-value = danger* **then**
4:         $hops_i = 0$
5:         Broadcast message $(i, hops = 0)$
6:     **if** *receive(j, hops)* **then**
7:         **if** $hops_j > hops + 1$ **then**
8:             $hops_j = hops + 1$
9:             *Broadcast message* $(j, hops_j)$
10:     **for** *all received j* **do**
11:         *Compute the potential pot^j of j using* $pot^j = \frac{1}{hops_j{}^2}$
12:         *Compute the potential at s_i using all pot^j, $pot_i = pot_i + pot^j$*

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 the 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-organize adaptively to the current landscape. It updates its distributed information content by running the potential field computation protocol regularly.

light sensor becomes an obstacle when it detects a high light intensity. For Mica Motes, we found that the light sensors work well.

Algorithm 2. The safest path to goal computation protocol.
1:   Let G be a goal sensor
2:   G broadcasts $msg = (G_{id}, my_{id}(G), hops = 0, potential = 0)$
3:   **for** all sensors s_i **do**
4:       Initially $hops_g = \infty$ and $P_g = \infty$
5:       **if** *receive((g, k, hops, potential)* **then**
6:         *Compute the potential integration from*
           the goal to here:
7:         **if** $P_g > potential + pot_i$ **then**
8:           $P_g = potential + pot_i$
9:           $hops_g = hops + 1$
10:          $prior_g = k$
11:          *Broadcast* $(G_{id}, my_{id}(s_i), hops_g, P_g)$

Algorithm 3. The navigation guiding protocol.
1:   **if** s_i is a user sensor **then**
2:       **while** Not at the goal G **do**
3:           Broadcast inquiry message (G_{id})
4:           **for** all received messages $m =$
             $(G_{id}, my_{id}(s_k), hops, potential, prior)$ **do**
5:               Choose the message m with minimal potential then minimal hops
6:               Let $my_{id}(s_k)$ be the id for the sender of this message
7:               Move toward $my_{id}(s_k)$ and *prior*
8:   **if** s_i is an information sensor **then**
9:       **if** receive (G_{id}) inquiry message **then**
10:          Reply with
             $(G_{id}, my_{id}(s_i), hops_g, P_g, prior_g)$

In this way, the network can adapt to sensor failure, to the addition of new
nodes in 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 node that can talk to the network in an online fashion.
Thus, the user can be viewed as a mobile node in a sensor network. 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.

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.

8   •   Q. Li and D. Rus

3.1 Analysis

3.1.1 *Correctness.*   Our protocols can correctly determine the safest path to the goal without getting stuck in the local minima which are often a limitation of the artificial potential fields methods.

THEOREM 3.1.   *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.   □

3.1.2 *The Hop Distance Model.*   One critical assumption behind Algorithm 1 is that we can represent distance in terms of numbers of hops. In general, how realistic is this model? To answer this question, we consider how the density of the sensor distribution affects the distance evaluation in our algorithms. We now address this question for the case where each node has a constant transmission range, which is an assumption consistent with our testbed hardware.

We would like to know the minimal number of hops a message has to travel from one sensor to another that is distance l away. This is hard to characterize since in our algorithms each sensor uses flooding to broadcast packets to all of its neighbors, and each sensor within the transmission range of the broadcasting sensor can forward the packets. An approximation can be obtained by allowing only the sensors at the boundary of the transmission range to forward packets. Of all those sensors, we choose the sensor that can make the most progress in the direction of the destination sensor. The number of hops computed this way is an approximation of the minimal number of hops.

Takagi and Kleinrock [1984] proposed the most forward routing and analyzed its average progress in the direction of the destination. We can use a similar analysis to approximate the distance of a single hop.

Suppose the distance between the current message holder and the destination is l, and the distance between the next message holder and the destination is l'.

Consider Figure 2. S is the current node holding the message, and D is the destination of the message. The density of the network is λ, and the sensors are distributed according to Poisson distribution. Suppose the next node along the path to D is A. Let $\angle BSD = \theta$. The area of the circular segment to the right of BC is $S_1 = \theta \cdot R^2 - R^2 \cdot \sin\theta \cdot \cos\theta$. In order for A to hold the message, two conditions must be satisfied: (1) There is no node to the right of A in that circle; (2) There must be at least one node in that square area. We now evaluate the probabilities of these two events.
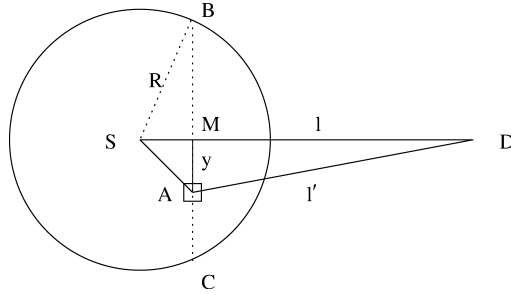
Fig. 2. This figure shows our approach to evaluating the expectation of l', the distance between the next message holder and the destination. S is the message source, and D is the message destination. S tries to find a node that can make the biggest progress in the direction to the destination. Suppose A is the next node.

The probability that there is no node in that area is

$$P_1 = e^{-\lambda S_1} = e^{-\lambda(\theta \cdot R^2 - R^2 \cdot \sin\theta \cdot \cos\theta)}.$$

The square area around A is $S_A = R \cdot \sin\theta \cdot dy \cdot d\theta$, and the probability that there is no node in that area is $e^{-\lambda R \cdot \sin\theta \cdot dy \cdot d\theta}$. It follows that the probability that there is at least one node in area around A is

$$P_2 = 1 - e^{-\lambda R \cdot \sin\theta \cdot dy \cdot d\theta} = \lambda R \cdot \sin\theta \cdot dy \cdot d\theta.$$

So the probability that the next node holding the message is in the square area around A is the product of P_1 and P_2. Thus, the probability that A holds the message is

$$P_{l'} = P_1 \cdot P_2 = e^{-\lambda(\theta \cdot R^2 - R^2 \cdot \sin\theta \cdot \cos\theta)} \cdot \lambda \cdot R \cdot \sin\theta \cdot dy \cdot d\theta.$$

The distance between A (the next node) and the destination is

$$l' = \sqrt{(l - R\cos\theta)^2 + y^2}.$$

Thus the expectation of l' is the integration upon all θ and y in the right and left half circles. The integration on the right half circle is:

$$\int l' \cdot P_{l'} = \int_0^\pi \int_{-R\sin\theta}^{R\sin\theta} \sqrt{(l - R\cos\theta)^2 + y^2}$$

$$\cdot e^{-\lambda(\theta \cdot R^2 - R^2 \cdot \sin\theta \cdot \cos\theta)} \cdot \lambda \cdot R \cdot \sin\theta \cdot dy \cdot d\theta.$$

For a sufficiently dense network, the probability of finding a node in the right half circle is large, but in a sparse network, it may not be true. We only consider sufficiently dense networks here. In order to integrate all the points on the left circle, we simply choose $l + 2 * R$ as the distance from the next node to the destination if the next node cannot be found in the right half circle. The expectation of l', $E(l')$ is the sum of the integrations in the left and right half circles.

Figure 3 shows the results of plotting the expectation for l' for different distances l. Note that all the curves with the same transmission range cluster
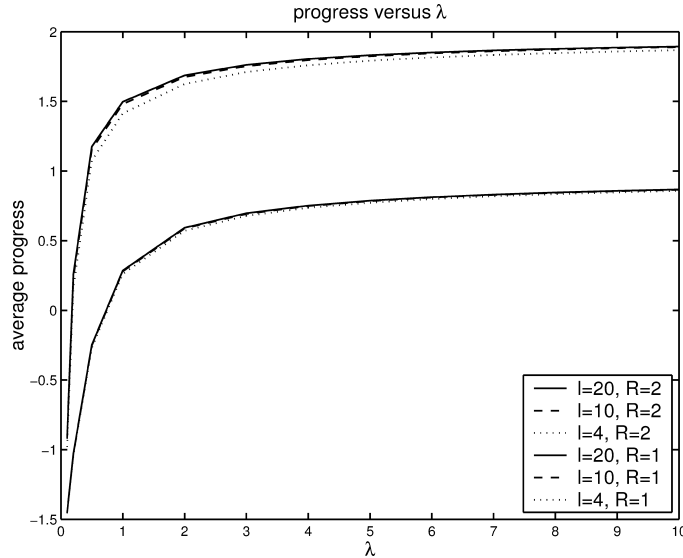
Fig. 3. Average progress by using most progress method (average progress is defined as $R' = l - E(l')$). The X axis is the density of the network, that is, how many nodes in a unit area on average. We plot the curves with different source to destination distances and transmission ranges.

together closely. In addition, because the formula we plot is a continuous function, all the values for l in between the boundaries we plot will be in the same range. Thus we conclude that the average progress is approximately the same for different distances l, with a fixed transmission range R. It follows that the average hop for a message to transmit from one node to another is just l divided by the average progress. Then the minimal ideal hop should be l/R, but the expected minimal hop in our real sensor network is l/R'. That is, the distance we evaluate is always R/R' times of the real distance.

Ganesan et al. [2002] reported the length of a hop may not be fixed as we observed in our experiments. Through experiments, we can get the expectation and the deviation of the length of a hop (call them E and d). According to the central limit theorem in probability theory, the length of n hops has the expectation of nE and the deviation of $\sqrt{n}d$, that is, the deviation (or the difference) between the real distance and the computed distance is in the order of $\sqrt{n}d$, which is small compared with the distance of the order of nE. This provides evidence that our algorithms will be robust when run on physical networks.

3.1.3 *Performance Bound of the Computed Path.*   We expect our protocols to compute the integrated potential value on the safest path, but they introduce error by approximating the distance and the potential value. We now compare the integrated potential value on the path found by our protocols and the optimal path to show how safe the found path would be. Here we assume the danger is a single point that is detected by a single sensor which is called a danger node.

THEOREM 3.2. *The computed potential integration on the computed path is upper and lower bounded with respect to the actual potential integration on the path.*

PROOF. Suppose we find a path from A to B by running our algorithms. The sum of the potential value on the sensor nodes by running our algorithm is P_1, and the nodes on the found path are $A = s_0, s_1, s_2, \cdots, s_k = B$. Let $\overline{s_0 s_1}, \overline{s_1 s_2}, \cdots, \overline{s_{k-1} s_k}$ (or $\overline{s_0 s_1 s_2 \cdots s_{k-1} s_k}$) be the path connecting all these nodes consecutively by lines. Let the integration on this path be P_2 (continuous line integration, not only on the points). We would like to compare P_1 and P_2; specifically, we would like to upper bound P_2. Take a look at $\overline{s_{i-1} s_i}$. Let the potential value of s_{i-1} be p_{i-1}, s_i be p_i. We assume we use the fixed transmission model. $|\overline{s_{i-1} s_i}| \leq R$ (R is the transmission range). For any danger source d_j, suppose the potential that s_i gets from obstacle d_j is $p_{ij} = \frac{1}{h_{ij}^2}$ where $h_{ij} = \frac{l_{ij}}{R}$, and l_{ij} is the distance between s_i and d_j. For any point t on segment $\overline{s_{i-1} s_i}$, let l_{tj} be the distance between t and d_j. Then $l_{tj} \geq l_{ij} - R$, so the potential value at t due to d_j is $p_{tj} = \frac{1}{h_{tj}^2} \leq \frac{R^2}{(l_{ij} - R)^2} = \frac{1}{(\frac{l_{ij}}{R} - 1)^2} = \frac{1}{(h_{ij} - 1)^2}$. So we have $\frac{p_{tj}}{p_{ij}} \leq \frac{h_{ij}^2}{(h_{ij} - 1)^2}$. Similarly, we have $\frac{p_{tj}}{p_{i-1j}} \leq \frac{h_{i-1j}^2}{(h_{i-1j} - 1)^2}$.

By integrating on the entire path, we have the following. $P_2 = \int_A^B \sum_j p_{tj} = \sum_{i=0}^{k-1} \int_{s_i}^{s_{i+1}} \sum_j p_{tj} \leq \sum_{i=0}^{k-1} \int_{s_i}^{s_{i+1}} \sum_j \left(\frac{h_{ij}^2}{(h_{ij} - 1)^2} \cdot p_{ij} \right) \leq \sum_{i=0}^{k-1} R \cdot \left(\sum_j \frac{h_{ij}^2}{(h_{ij} - 1)^2} \cdot p_{ij} \right)$ (since $|\overline{s_{i-1} s_i}| \leq R$)

If $\frac{h_{ij}^2}{(h_{ij} - 1)^2} \leq q_1$ for all i, j, we have $P_2 \leq R \cdot q_1 \cdot \sum_{i=0}^{k-1} \sum_j p_{ij} = R \cdot q_1 \cdot P_1$.

On the other hand, we have the following. First we have $|\overline{s_{i-1} s_{i+1}}| \geq R$, so $|\overline{s_{i-1} s_i}| + |\overline{s_i s_{i+1}}| \geq R$. Let's find s_i' on $\overline{s_i s_{i+1}}$ such that $|\overline{s_{i-1} s_i}| + |\overline{s_i s_i'}| = R/2$, or find s_i' on $\overline{s_{i-1} s_i}$ such that $|\overline{s_i' s_i}| + |\overline{s_i s_{i+1}}| = R/2$. Without loss of generality, we assume s_i' is on $\overline{s_i s_{i+1}}$. The distance from any point on $\overline{s_{i-1} s_i}$ or $\overline{s_i s_i'}$ to s_{i-1} is no greater than $R/2$ (also less than R), so for any point t on these two segments, we have $p_{tj} = \frac{1}{h_{tj}^2} = \frac{R^2}{(l_{tj})^2} \geq \frac{R^2}{(l_{i-1j} + R)^2} = \frac{1}{(h_{i-1j} + 1)^2}$. Similarly, the distance from any point on $\overline{s_i' s_{i+1}}$ to s_i is no greater than R, so for any point t on this segment, we have $p_{tj} = \frac{1}{h_{tj}^2} = \frac{R^2}{(l_{tj})^2} \geq \frac{R^2}{(l_{ij} + R)^2} = \frac{1}{(h_{ij} + 1)^2}$. Let $s_0, s_2, \cdots, s_{2i}$ be $s_0', s_2', \cdots, s_{2i}'$, and we then create $s_1', s_2', \cdots, s_{2i+1}'$ by the previous procedure. It follows that $P_2 = \int_A^B \sum_j p_{tj} = \sum_{i=0}^{k-1} \int_{s_i'}^{s_{i+1}'} \sum_j p_{tj} \geq \sum_{i=0}^{k-2} \int_{s_i'}^{s_{i+1}'} \sum_j \left(\frac{h_{ij}^2}{(h_{ij} + 1)^2} \cdot p_{ij} \right) \geq \sum_{i=0}^{k-2} \frac{R}{2} \cdot \left(\sum_j \frac{h_{ij}^2}{(h_{ij} + 1)^2} \cdot p_{ij} \right)$.

If $\frac{h_{ij}^2}{(h_{ij} + 1)^2} \geq q_2$ for all i, j, and $\sum_j \frac{h_{k-1j}^2}{(h_{k-1j} + 1)^2}$ is very small compared to P_1, we have $P_2 \geq \frac{R}{2} \cdot q_2 \cdot \sum_{i=0}^{k-1} \sum_j p_{ij} = \frac{R}{2} \cdot q_2 \cdot P_1$.

Combining the preceding analysis, we have $\frac{R}{2} \cdot q_2 \cdot P_1 \leq P_2 \leq R \cdot q_1 \cdot P_1$. This tells us that the real potential integration on the computed path is relatively close to the computed potential integration of the sensor nodes on that path. □

Theoretically, there is an optimal path that has the minimal potential integration and may not traverse any sensor node, but this path is not feasible in our system since a user can only go from one sensor to another by listening to the reply from the next sensor in our navigation protocol. Therefore, instead of defining an optimal path, we define an optimal sensor path as one

that is composed of a series of sensor nodes that are connected consecutively by straight line segments (the connected nodes are within the transmission range of each other) which we expect to characterize the motion of a user. Assume the optimal sensor path is a series of segments $\overline{u_0 u_1 \cdots u_l}$, where $u_0, u_1, \cdots, u_l$ are the sensor points and the potential integration along all these segments is P_0. We now compare the potential integration of this optimal sensor path (P_0) with that of our computed path (P_2).

THEOREM 3.3. *The potential integration on the computed path is upper bounded with respect to the potential integration on the optimal sensor path.*

PROOF. Starting from $u_0 = s_0$, we want to choose some nodes from $u_0, u_1, \cdots, u_l$ in that order. Suppose we have chosen $s_0 = u_0, s_1 = u_{l_1}, \cdots, s_{2i-3} = u_{l_{2i-3}}, s_{2i-2} = u_{l_{2i-2}}$. Let's choose the next two points $s_{2i-1} = u_j, s_{2i} = u_{j+1}$ with the least j such that $j > l_{2i-2}$ and $|\overline{s_{2i-2}u_{j+1}}| = |\overline{u_{l_{2i-2}}u_{j+1}}| > R$. The process continues until there is no point left, and we let the last point be $s_k = u_l$. Let the potential sum on all those points s_i ($0 \le i \le k - 1$) be P_0' (by adding up the potential values on all the node points), and we will compare P_0 and P_0'. For any $1 \le x \le k$, we have $|\overline{s_{x-1}s_x}| \le R$, and for any $0 \le y \le \lfloor k/2 \rfloor$, we have $|\overline{s_{2y-2}s_{2y-1}}| + |\overline{s_{2y-1}s_{2y}}| > R$. Consider segments $\overline{s_{2y-2}u_e \cdots u_f s_{2y-1}s_{2y}}$ on the optimal sensor path. If the sum of all the segments of $\overline{s_{2y-2}u_e \cdots u_f s_{2y-1}}$ is no less than $R/2$, we find s_{2y-1}' on the segments of $\overline{s_{2y-2}u_e \cdots u_f s_{2y-1}}$ such that $|s_{2y-2}u_e \cdots u_p s_{2y-1}'| \ge R/2$, and all the points on segments $s_{2y-1}'u_r \cdots s_{2y-1}$ are within R distance from s_{2y-1}, and $|s_{2y-1}'u_r \cdots s_{2y-1}| + |\overline{s_{2y-1}s_{2y}}| \ge R/2$. (The argument is as follows. Draw a circle with radius R centered at s_{2y-1}. If the circle intersects segments $\overline{s_{2y-2}u_e \cdots u_f s_{2y-1}}$, let the last intersection point be t_y. We have $|\overline{s_{2y-2}t_y}| + |\overline{t_y s_{2y-1}}| + |\overline{s_{2y-1}s_{2y}}| \ge |\overline{s_{2y-2}s_{2y-1}}| + |\overline{s_{2y-1}s_{2y}}| \ge R$, and $|\overline{t_y s_{2y-1}}| = R$. There must be a point s_{2y-1}' on segments $\overline{t_y u_a \cdots u_b s_{2y-1}}$ such that $|\overline{s_{2y-2}u_e \cdots u_p s_{2y-1}'}| \ge R/2$, $|s_{2y-1}'u_r \cdots s_{2y-1}s_{2y}| \ge R/2$, and all points on $\overline{s_{2y-2}u_e \cdots u_p s_{2y-1}'}$ is within R from s_{2y-2}, and all points on $\overline{s_{2y-1}'u_r \cdots s_{2y-1}s_{2y}}$ is within R from s_{2y-1}). If the sum of all the segments of $\overline{s_{2y-2}u_e \cdots u_f s_{2y-1}}$ is less than $R/2$, we find s_{2y-1}' on the segment of $\overline{s_{2y-1}s_{2y}}$ such that $|s_{2y-2}u_e \cdots u_f s_{2y-1}s_{2y-1}'| = R/2$. In either case, any point t on segments $\overline{s_{2y-2}u_e \cdots s_{2y-1}'}$ has potential value $p_{tj} \ge \frac{1}{(h_{2y-2j}+1)^2}$, and any point t on segments $\overline{s_{2y-1}' \cdots s_{2y}}$ has potential value $p_{tj} \ge \frac{1}{(h_{2y-1j}+1)^2}$. Both $|\overline{s_{2y-2}u_e \cdots s_{2y-1}'}|$ and $|\overline{s_{2y-1}' \cdots s_{2y}}|$ are no less than $R/2$. $P_0 = \sum_{i=0}^{l-1} \int_{u_i}^{u_{i+1}} \sum_j p_{tj} \ge \sum_{i=0}^{\lfloor k/2 \rfloor - 1}(\int_{s_{2i-2}}^{s_{2i-1}'} \sum_j (\frac{h_{2i-2j}^2}{(h_{2i-2j}+1)^2} \cdot p_{2i-2j}) + \int_{s_{2i-1}'}^{s_{2i}} \sum_j (\frac{h_{2i-1j}^2}{(h_{2i-1j}+1)^2} \cdot p_{2i-1j})) \ge \sum_{i=0}^{k-2} \frac{R}{2} \cdot (\sum_j \frac{h_{ij}^2}{(h_{ij}+1)^2} \cdot p_{ij})$.

If $\frac{h_{ij}^2}{(h_{ij}+1)^2} \ge q_0$ for all i, j, and $\sum_j \frac{h_{k-1j}^2}{(h_{k-1j}+1)^2}$ is very small compared to P_0', we have $P_0 \ge \frac{R}{2} \cdot q_0 \cdot \sum_{i=0}^{k-1} \sum_j p_{ij} = \frac{R}{2} \cdot q_0 \cdot P_0'$. Since $P_0' \ge P_1 \ge \frac{P_2}{Rq_1}$, we have $P_2 \le \frac{2q_1}{q_0}P_0$, that is, our computed path has bounded potential integration. □

3.1.4 *Propagation and Communication Capability.* Two natural questions arise about the protocols we described previously: (1) How much time does it

take to propagate the obstacle and goal information? and (2) Is the network capable of transmitting all the information? In this section, we answer the two questions in the context of our current implementation in which we use one packet for propagating the information of each obstacle or goal for every broadcast. To optimize the bandwidth usage by reducing the information transmission, we can combine the information about two or more obstacles and the goal into a packet or use information encoding to reduce the information redundancy among the neighboring nodes.

We assume that each node has fixed transmission range and its neighbors (say k nodes) should be silent to avoid contention when that node broadcasts. For the obstacle information propagation, assume the number of the concerned obstacles is o; that is, on average, each node has to process the information of o obstacles. Let the transmission rate for each node be b packets/s. Then the time for the obstacle information propagating to a node is okl/b where $l = min(L, l_0)$, L is the distance for the potential value to become 0, and l_0 is the distance between the node and the obstacle, both in number of hops. The formula is for the case in which we add waiting time for each broadcast; that is, each node only broadcasts once for each obstacle information propagation. In this case, each node needs to wait for k/b time before broadcasting the best value. This waiting time allows enough time for each of the node's neighbors to broadcast the packet if they hold the same value as the current node so that they do not collide. For the case without explicit waiting time, the MAC protocol enforces this delay to make sure all the packets go through smoothly. On the other hand, suppose we do not have the waiting time scheme, each node may then broadcast multiple times because the least number of hops is unlikely to be obtained by the first received message so that the node needs to broadcast several packets before the best value is propagated. In this case, we must multiply the propagation time by another parameter m, which is the average messages broadcast for each node. Similarly, we can evaluate the propagation time for the goal information.

The transmission rate of the Mote sensors we are using is approximately $b = 40$ packets/s, so for $k = 8$, the added waiting time to each node is $8/40 = 0.2$s. Regardless of how many obstacles there are in this system, if each node is in the proximity of only one obstacle, it takes $0.2 * 10 = 2$ seconds to propagate the information up to 10 hops away.

When the obstacles are static and we do not care about the time, the network is capable of transmitting these bits. If there are time constraints (e.g., there are moving obstacles and the location of an obstacle must be known to the network within a distance resolution d), the network may not be able to carry all the information. Suppose the maximal speed of the obstacle is v. In the worst case, an obstacle generates v/d packets per time unit, so each node needs to process ov/d packets which should be less than b/k, that is, $ov/d < b/k$. If we do not have the waiting time, we expect more packets will be generated and the precision about the vehicle represented by the network will be low.

Suppose an obstacle is moving at a speed of $1m/s$, the maximal transmission rate for a node is 40 packets/s, the number of concerned obstacles is 1, and the number of the concerned neighbors of a node is 8. The network can sustain updates at a resolution of 0.2 meters. If we have the same network, but the

14   •   Q. Li and D. Rus

moving object is a vehicle moving at a speed of 30 miles, the vehicle updates can happen every 2.7 meters.

3.2 Discussion

3.2.1 *Node Failure.*   In a real sensor network deployment, node failure may not be an exception. Sensors may be destroyed by wind, rain, snow, temperature, or treading of vehicles. They may also malfunction due to component failure or battery depletion. Moreover, the terrain on which sensors are deployed contributes to the communication failure. It is natural to ask how the node failure affects the path found or the correctness of the protocol. We consider the problem in several periods of the protocol run.

If node failure occurs before the protocol initiation, it only affects the density of the sensor network. When only a very small number of nodes fail, the randomness of the sensor distribution is not harmed so the protocol will not be affected. However, failure of multiple nodes may lead to two cases: (1) some part of the network has lower density because of the node failure; (2) holes may appear in some part of the network because node failure may be epidemic. These two cases are not inherent to the node failure problem but should be addressed to complete the performance of the protocol under different network topologies.

When the density of the network is low, the number of hops between two points becomes larger. Since we use the number of hops between two points to approximate the distance between them, the uneven distribution of the sensors in the network may generate error, for example, for the same distance, the number of hops in one area may be different from that in another. This gives the protocol biased distance estimation. However, the theoretical performance bound (Theorem 3.3) still holds if we assume the transmission range is a circle with a constant radius because our theoretical analysis does not depend on any assumptions about the node density. A biased distance estimation affects the performance dramatically when a point is close to the danger area because a slight variation of the distance estimation will change the performance greatly. When a point is relatively distant from the danger, the estimated distance variation due to the uneven density is negligible because the potential value is not sensitive relative to the long distance.

When holes are present in the network (e.g., sensors are destroyed by external factors), the number of hops does not capture the distance since the hops are counted by going around a hole which shows a longer distance than the actual distance. Our protocols cannot deal with this case correctly since we assume the nodes are randomly distributed.

When nodes fail in the middle of the distance estimation protocol and path computation protocol, the protocols still run well since they do not depend on specific nodes. As long as the network is connected, the protocols can compute the safe path, and so the performance will be degraded.

After the path has been computed, a user can query the potential integration from its neighbors to know where to go. Therefore, the protocol can tolerate

node failure as long as a node within the user's neighborhood has a smaller potential integration. The one-hop neighboring node that has a smaller potential integration may fail, but the user can always find a node with smaller potential integration among its multiple-hop neighbors.

3.2.2 *Assumptions of the Protocols.*   We assume that a user can always go anywhere in the sensor field without being blocked. It is possible that some barricade in the field, e.g., a wall, may block a user but not communication. Our protocol works in this case provided the obstacle can be detected by sensors and can be regarded as a dangerous spot.

Our protocols work under the assumption that the sensors can discover the events of interest. It is possible that the sensors may be destroyed and this requires the network to detect the void area. The nodes on the edge of the void area can play the role of nodes detecting danger in our protocols and run our protocols to compute the safest path around the hole.

3.2.3 *Does Navigation Protocol Always Find a Path?.*   The protocol always finds a path if the network is connected. In the ideal case, the protocols can compute: (1) the distance between any point and the danger, and (2) the exact potential integration of a path. Since the sensor network is discrete, our protocols compute approximations for the distance and potential integration. As long as the network is connected, the protocols will, however, compute on the connected component and always find a path if the user and the goal are in the same connected component. How well the computed path matches the ideal path is related to the density of the network, the connection graph (e.g., two nearby nodes may have a communication path that is long by taking a detour, will substantially affect the performance since this communication pattern does not capture the physical spatial distance), and the transmission range.

3.2.4 *How Do Long Links Affect Computation?.*   The transmission range of a node may not be a perfect circle. Instead, it may be irregular shape, which may introduce long links. We have analyzed this situation in the hop/distance model part. The random analysis assumes that the effect of the long link will decrease with increasing number of hops.

A long link creates the illusion that two far away nodes are close to each other. The long link enforces the protocols to sum up only two potential values on its two ends to approximate the potential integration over this link, which in the normal case, should be the sum of multiple potential values of nodes on that link (that is, if transmission links are short, more nodes are required to hop from one end to another). This affects the distance estimation and safe path computation.

Our implementation uses a neighbor profile technique to eliminate infrequent and asymmetric links, which has the effect of eliminating long links.

3.2.5 *How Does Density Affect Performance?.*   We discussed the relationship between the density and the quality of the computed path in Section 3.1.2, Theorem 3.2 and 3.3. The higher the node density is, the better we can estimate

the real distance and the potential integration of a path, then the better path we can get. This is also true for path computation.

4. EXPERIMENTS

4.1 Experiment Setup

We have implemented the algorithms described in Section 3 using the Mote MOT300 sensors [Hill et al. 2000]. The goal is represented with one Mote. The obstacles are represented by one Mote each. The user traversing the sensor network is also represented by one Mote. In our experiments, we asked both the goal nodes and the obstacle nodes to generate the potential field and propagate it to the entire network periodically. This demonstrates experimentally the adaptation of our protocols to changes in the goal nodes and obstacle nodes.

A first experiment was designed to show empirically that the protocols are correct. In this first experiment, we used a grid of 12 first generation Motes. The Motes were placed approximately in line. Several nodes were placed around the obstacles in order to test if the safest computed path is a detour around the obstacle. All neighbors were within communication range. The application was run by iterating a request for the next step by the user, a response by the network, and a move to the direction of the network response. To implement this last part, we assumed that the nodes know their location and could transmit it to the moving object/user. This can be done by augmenting Motes with a GPS location, or a localization algorithm such as Corke et al. [2003]. Since we have not done this augmentation of the hardware yet, we simulated location knowledge by placing the Motes in a grid pattern and supplying coordinates. The potential field and goal path protocols were 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 the oldest entry is erased to save space.

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 the node's neighbors.

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

In the second experiment, we used a testbed of 50 Motes MOT300. We arranged the nodes in the given topology and gave each node position information (which could be obtained as previously described.) We ran a suite of different network topologies and measured the network stabilization time when obstacles and goals are injected online in the network. Table I summarizes 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

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

Exp. Config.	danger prop.	Shortest distance	goal prop.	safest path	Exp. Config.	danger prop	Shortest distance	goal prop.	safest path
	0.23	1.13	0.17	9.23		1.23	22.33	2.27	19.27
	1.20	20.23	2.13	3.13		4.20	9.10	1.37	22.63
	0.20	2.17	0.13	4.23		5.20	20.30	1.17	9.40
	1.16	3.13	2.13	10.03		1.30	16.23	1.10	2.17
	0.17	6.17	0.04	19.37		0.20	7.17	1.13	3.1
	0.10	3.10	0.10	1.07		9.37	17.37	0.33	8.43
	0.23	1.33	0.13	1.07		1.10	1.10	3.13	5.10
	7.27	10.13	0.04	33.80		4.20	4.20	0.30	12.80
	1.43	2.10	0.17	2.17		7.27	12.23	0.20	8.40
	1.10	27.17	4.27	9.10		1.10	27.17	4.27	9.10
	4.20	4.20	0.30	12.80		7.27	10.13	0.04	33.80
	0.20	7.17	1.13	3.13		0.17	6.17	0.04	19.37

experiments has been to determine 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(right). For each experiment, we set the transmission range to be very small (9″). In all these experiments, we focused on measuring the performance of the network as a whole and did not use a base station. Thus, we did not collect data in a central place. To collect timing data, we used two procedures: videotaping and data logging.

We used the videotaping procedure to capture the global behavior of the sensor field. The Mote LEDs were programmed to capture the state of the Mote. 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
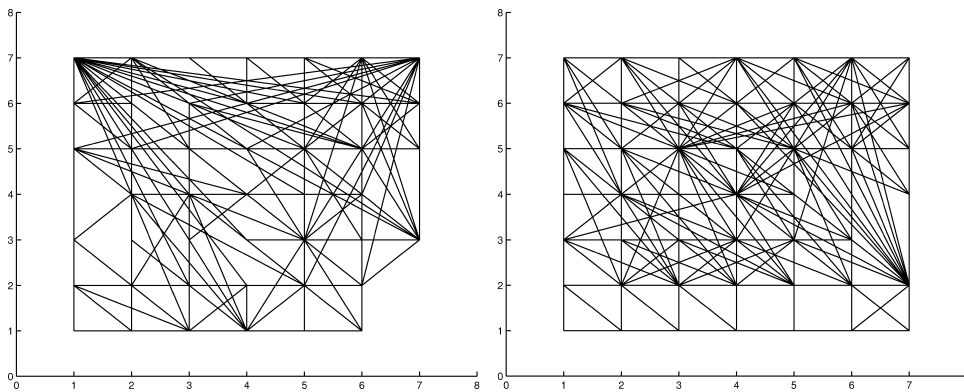
Fig. 4.   Measured communication graph of the experimental 7 × 7 grid network. The left is the one in our second experiment. Notice the absence of many point to point links we expected to be available (from example from (1, 1) to (2, 1)) and the presence of long links we did not expect to have, for example from (1, 7) to (7, 6). The right one is the measured communication graph in performance improvement.

measurements which gave us a resolution of 1/30th 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 for the navigation algorithms are on the order of seconds, we believe our methodology is accurate enough.

We also used the logging procedure to collect data about the message flow in the system. In the logging procedure, information about incoming and outgoing messages as well as internal events of interest were logged to the 4Mbit flash chip on the Mote sensors with a resolution of 1/128 of a second. After each experiment, the data was read out over the radio link and then postprocessed using custom C programs. There are some limitations to this approach since data can be lost if a write to the flash chip is already pending. This was minimized by adding buffers so that at least one message or event of each type could be queued for writing if a write was already pending.

Figure 4 shows the connectivity between Motes. A line between two Motes indicates that they communicated directly at least once in the experiment. We can see how irregular the connection graph is. Note the Motes in the right corner which are completely disconnected.

In the third experiment, we implemented the performance improvement methods in Section 4.2 to eliminate the asymmetric and transient links and to reduce the network congestion. The experiment set-up is the same as the second experiment and Figure 4 (right) shows the communication graph.

4.2 Implementation Issues

Our navigation algorithms have an implicit assumption that the communication paths in the network are bidirectional. Since the safest path is computed backward from the goal, messages have to be able to flow in the opposite direction to lead the user to the goal. Our experience (see Section 4.6) has taught us that not all links in sensor networks are bidirectional. For example, Figure 5 (left) shows the distribution of symmetric and asymmetric links in
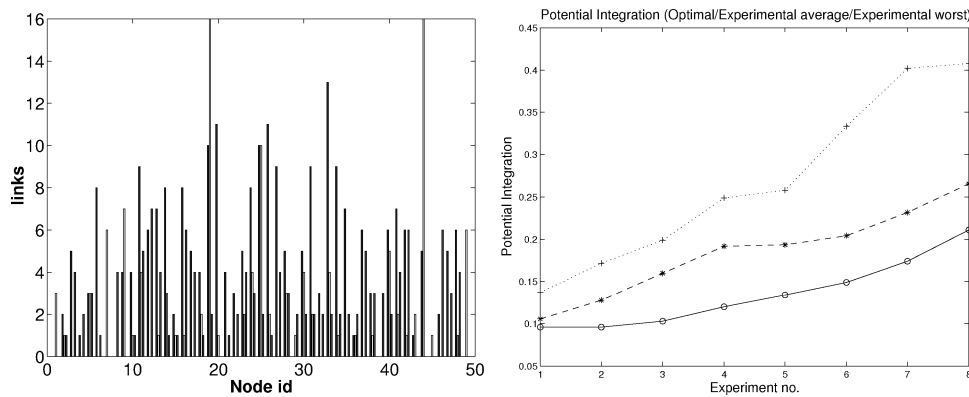
Fig. 5.   The left figure shows the distribution of symmetric and asymmetric links during one experiment. The x-axis shows the node id and the y-axis the number of links. For each node, we have three bars: the first shows the number of symmetric links, the second is the number of unidirectional outgoing links, and the third the number of unidirectional incoming links. The experiment was conducted on a 7×7 grid network running our navigation protocol for approximately one minute. The right figure shows the potential integrations in 54 experiments with eight different network topologies. The solid, dashed, and dotted lines are the potential integrations on optimal path, average over all computed paths, and the worst computed, respectively.

an experiment with a 7×7 grid of Mote sensors. This is consistent with data from Ganesan et al. [2002]. We propose the following method for identifying the bidirectional links in the network. The computation can be thought of as an additional protocol run by each node.

Each node does neighbor profiling to find all of its stable one-hop neighbors bidirectionally; that is, these neighbors should be reachable to and from the node with high probability. In this way, we may ward off the unidirectional link nodes that may lead to long distance hops. Each node only uses the received packets from its stable neighbors after profiling. In our current implementation, we perform the neighbor profiling on the fly. Every time a node receives a packet it increases the frequency of the sender of the packet which measures the stability of that link. A link is used only if its frequency is higher than some threshold value which is one fifth of the maximal frequency of all the links in our implementation. A parameter we chose for our experiments is 1/5.

A side effect of neighbor profiling is the removal of many of the transient links that are active for a very short time. By exchanging the information about the frequency of two neighbors, the system ends up using the most stable bidirectional links. Our hop distance can also be close to average instead of to abnormal.

Algorithms 1 and 2 require each sensor to send a broadcast upon receiving a message with fewer hops to danger 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 node location or the minimal potential integration to the goal is useful. To reduce the message broadcasts, we let each sensor wait for some time before broadcasting. The waiting time for sensor s_i is proportional to one unit in Algorithm 1 and to the value pot_i in Algorithm 2. The main idea is

to let the message traveling time be proportional to the hops from the danger or the potential integration along the path traveled. Then the messages that carry the nonoptimal value will be suppressed, and only the messages that carry the optimal value will get broadcast. We can prove that the number of message broadcasts for each sensor is 1 in each algorithm using this technique [Aslam et al. 2003]. In our current implementation, we let each sensor wait for one unit time plus a small random number to reduce the message broadcasts and traffic congestion due to the simultaneous transmissions.

In order to desynchronize the nodes so that they would not simultaneously broadcast the same packet, we also add random variable waiting time.

Packet loss is common in our Mote network because of the network congestion or the inability of the Mote to handle the incoming packets. Thus it is important that we design protocols that repeat the packet transmission.

Most of the information stored at a node can be inferred by reading the protocols. To adapt the network topology (goal and obstacles) change, each sensor periodically flushes its route cache (route to obstacles and goal) with all the other information unchanged. Currently we have not included the capability to tune the cache expiration timer. Instead, we fix the expiration time to flush the caches.

4.3 Correctness Validation

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

Figure 6 shows the comparison between the measured real distance and the hops counted using our algorithm. The data was collected in our 7×7 grid network. We can see the measured real distance is approximately linear in the number of hops. Figure 5 (right) shows the potential integrations (line integration instead of the sum of the point potentials) in 54 experiments with eight different network topologies. For each network topology, we computed the optimal path by using dynamic programming and recorded the computed paths in several experiments. The solid line is the potential integration on the optimal path. The dashed line is the average potential integration over the computed paths. The dotted line is the worst potential integration among all the computed paths in experiments. Note that the potential integration bears no linear relationship to the distance. Compared with a dangerous path, which has a potential integration of 3–5, the computed path is quite close to the safest path.

4.4 Measuring Adaptation

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
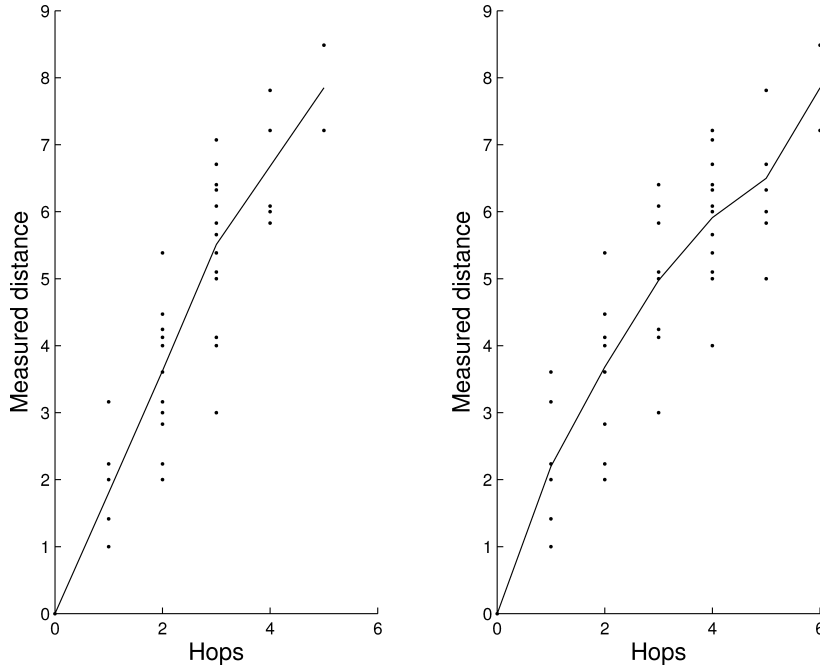
Fig. 6.   This figure shows the comparison between the measured real distance and the hops counted using our algorithm.

which depends on the maximal hops from the goals or the obstacles to any node in the network. When an obstacle is added to the system online, it takes an identical amount of time to diffuse the information to the whole network.

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 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. Table I shows 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 II 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 user finds the path to the goal. The route cache on each mote is 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.

Nodes were configured to log records of the packets sent and received, corresponding time, and related internal events. The network was a 7×7 grid with 49 Motes evenly placed on the grid. The neighboring Motes were spaced

Table II. 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. 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.)

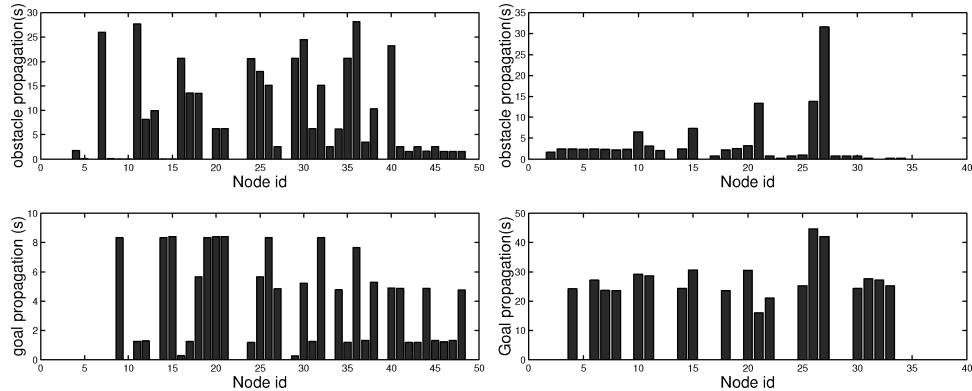| Exp. No.	Exp. Config.	Response Time	Exp. No.	Exp. Config.	Response Time
0		0	8		4.43
1		3.63	9		5.33
2		6.83	10		4.43
3		3.60	11		9.63
4		4.93	12		6.27
5		2.13	13		4.50
5		1.73	14		4.87
7		2.23	15		6.70

Fig. 7. The obstacle and goal propagation time distribution. In our experiment, the nodes were arranged in a 7×7 grid with obstacles placed at $(1, 1)$ and $(7, 7)$. The nodes on x-axis are sorted according to the Manhattan distance to $(1, 1)$, while the y-axis shows propagation time (in seconds). The top two figures show the data collected before performance optimization, while the bottom two figures refer to a network after performance optimization.

apart from each other at a distance slightly less than the transmission range in the appropriate direction. The two obstacles were placed at $(1, 1)$ and $(7, 7)$; the goal was put at position $(1, 7)$. Starting from $(1, 1)$, we numbered the Motes along the lines parallel to the line $(1, 7)$–$(7, 1)$ so that 1 and 49 were obstacles and 22 was the goal. The number of each mote gives a sense of the distance to the two obstacles. The obstacles and the goal periodically broadcast beacons. Each mote rebroadcasts a packet only if the received packet has a value that is as good, or better than, the current optimal value.

In order to distinguish the information propagation of the obstacle and goal, we first turned on the obstacles for approximately 30 seconds, and turned them off, then turned on the goal mote for more than 30 seconds.

Figure 7 (top left) shows the obstacle information propagation time, that is, the time when a mote receives a stable potential value. Some Motes get the
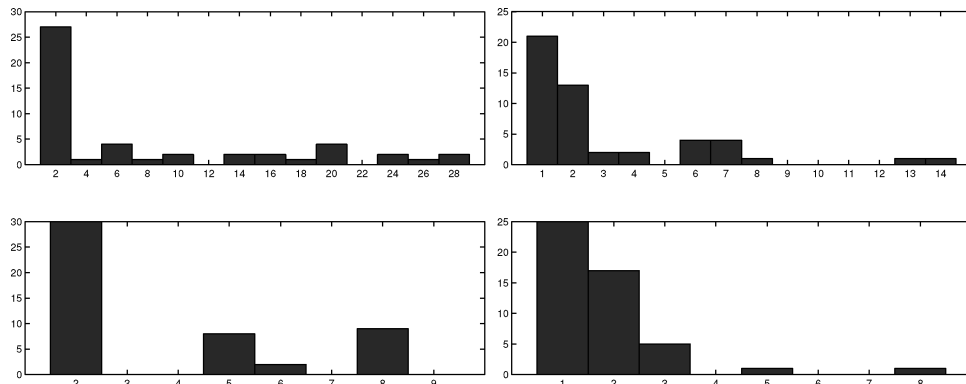
Fig. 8.   The histograms of the obstacle and goal propagation time corresponding to Figure 7. The y-axis is the number of nodes that have the propagation time corresponding to the value on x-axis. The left two figures show the data collected before performance optimization, while the right two figures refer to a network after performance optimization.
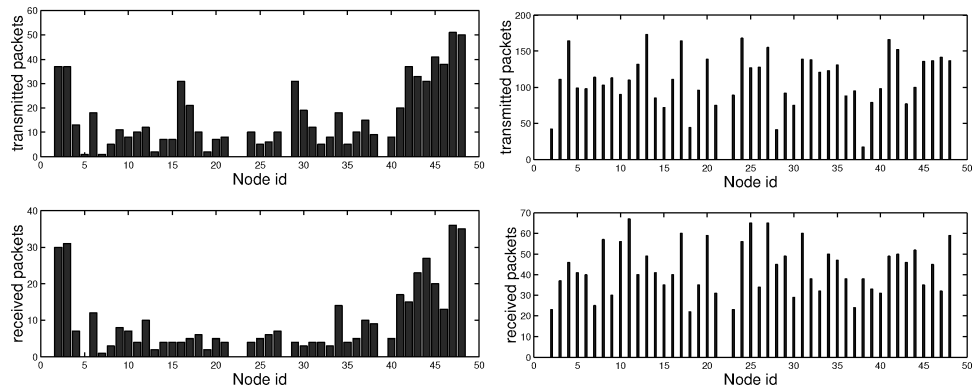

Fig. 9.   Transmission count distribution. The nodes on x-axis are sorted according to the Manhattan distance to (1, 1), while the y-axis represents the number of packets (send/receive) for that node. The left two figures show the data collected before performance optimization, while the right two figures refer to a network after performance optimization.

stabilized potential value very quickly, but it takes a long time for a fraction of Motes to finally get the potential. The same observation can be made in Figure 7 (bottom left), which shows the goal propagation time, defined as the time for a mote to get a stabilized integration value to the goal. The left two figures in Figure 8 show the histogram of the propagation time.

Figure 9 (left two figures) presents the number of transmitted packets and received packets at each mote. The Motes closer to the obstacles transmit and receive more data. In the middle of the x-axis, the heightened activity represents Motes that are close to the goal. The left two figures in Figure 10 show the histogram of the transmission counts.

Figure 11 (top figure) plots the send/receive activity of each node over time, which gives more detail about the packets sent and received. The Motes close to the obstacles or the goal receive and rebroadcast more packets than other Motes.
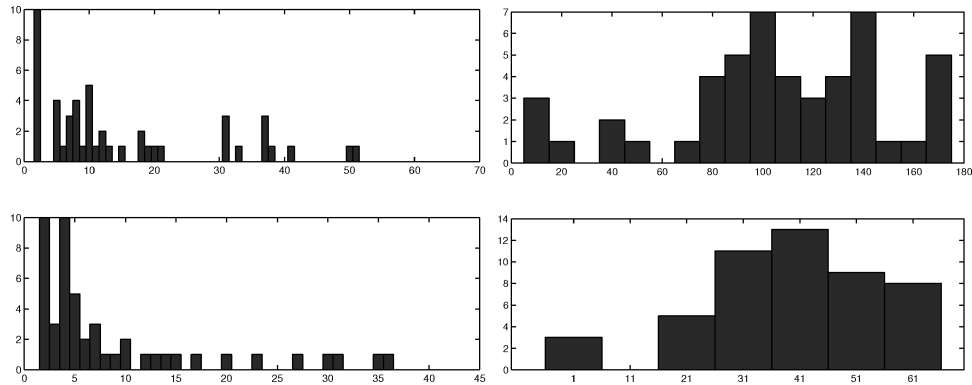
Fig. 10.   The histograms of the message transmission and reception counts corresponding to Figure 9. The y-axis is the number of nodes that transmit and receive the number of messages corresponding to the value on x-axis. The left two figures show the data collected before performance optimization, while the right two figures refer to a network after performance optimization.

In this figure, we observe some void areas where no mote sends or receives any packet. This is because many Motes do not reliably rebroadcast packets to their neighbors. We believe this is caused by two factors. One is that the rate of packet reception at these key nodes is too high, and thus they are unable to process all incoming messages. Another is that the packets these nodes forward to their neighbors are corrupted because of network congestion. This also explains why the obstacle and goal propagation times are uneven as much more traffic is generated by two obstacles than by one goal.

4.5 Performance Optimization

We optimized the message broadcasts using the methods described in Section 4.2 and performed several experiments with this implementation. The goal was to eliminate the asymmetric and transient links and to reduce the network congestion. The experiments were conducted on the same 7×7 grid as the ones in Section 4.4. The following figures were plotted in the same fashion as the related figures in the previous experiments. We observed the following (as compared to the initial suite of experiments).

(1) The obstacles and goal propagation time (Figure 7, right two figures) and their histogram figures (Figure 8, right two figures). The obstacle propagation was done very quickly and evenly for each node because the network had less congestion. Our current waiting time scheme gave the priority to the packets that traveled with less hops.

(2) Packet send/receive count (Figure 9, right two figures) and their histogram figures (Figure 10, right two figures). Compared to our previous scheme, we see a much more balanced packet transmission on all the nodes. Most of the nodes showed the increase in the transmitted packet both for sending and receiving which suggests that fewer packets were suppressed because of the congestion, and all the nodes had quite a large probability to broadcast their best computed value to the network.
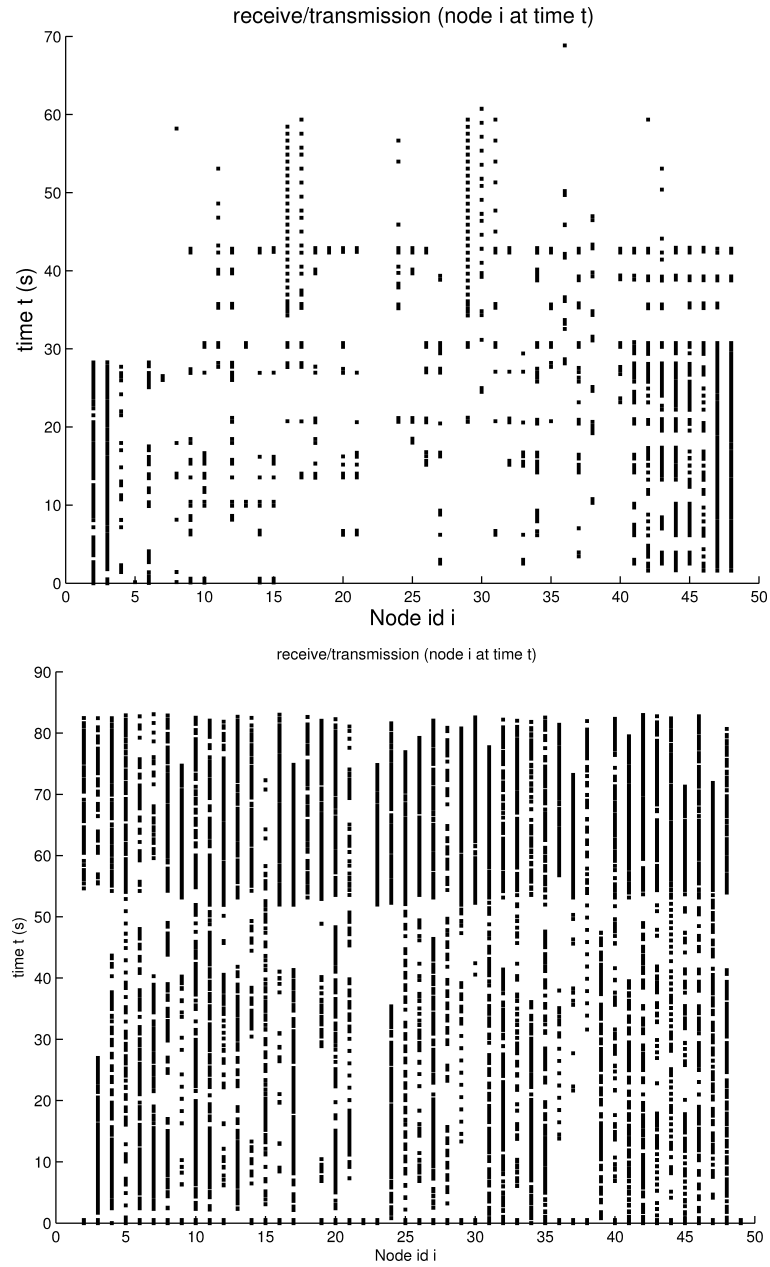
Fig. 11. Transmission/reception of packets by individual nodes over time. The bottom part signifies the propagation of the information from the two obstacles (1 and 49) in the first period of the experiment when only the obstacles were turned on. The top part is the goal information propagation in the second half of the experiment when only the goal was turned on. The top figure shows the data collected before performance optimization, while the bottom figure refers to a network after performance optimization.

(3) The packet send/receive analysis (Figure 11, bottom). In this figure, we have more packets for goal propagation because each node actively broadcasts (broadcasts once for every second) to test the network congestion. We see that although there is heavy network traffic, the sends and receives on each node are balanced. For all the nodes, the transmitted packets are balanced among all the nodes. We can reduce the transmitted packets in goal propagation by changing the program.

4.6 Lessons Learned

Several interesting aspects of these experiments can be observed. The time for network stabilization 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 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 networks. 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 to 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 an 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 better characterize the likelihood of these uncertainty conditions.

4.7 Conclusion

Our experiment results proved the correctness of the protocols, showed that the protocols can adapt to the network topology change well, and demonstrated that

a number of performance improvement methods can be used to reduced the adverse communication quality (e.g., asymmetric links, network congestion, and transient links). Our protocols are very robust and adapt to network topology change, node failure or introduction, and irregular communication. Our data shows that the navigation protocols adapt to new obstacles and goals and generate the next node in time on the order of seconds. We believe that this is adequate for guiding mobile robotic nodes or people across a network field because the space we expect these nodes to travel in a few seconds is small. Our experiments also show that, in practice, a fairly complicated task (e.g., navigation using sensor networks) can be accomplished by using simple sensor network operations. The protocols work well without relying on localization, complex computations, and coordinated communications. The protocols are scalable when measures to regulate the communication are added. However, our protocols generate a lot of network traffic to compute the best path. This is due to the flooding scheme we proposed. However, we can improve upon the original scheme by developing a scalable algorithm using a Voronoi diagram method to direct message flow as proposed in Section 5. An alternative approach would be to enforce broadcast by only one node in a small area of the network.

5. OPTIMIZING COMMUNICATION WITH VORONOI DIAGRAMS

The navigation protocols discussed previously compute the paths from all the nodes in the network to a single goal by broadcasting to the entire network. They are distributed algorithms with no centralized control. However, there are some drawbacks: (1) every time the event sites change, the protocols must recompute the entire map; (2) the searching space for the safest path to the goal is the entire network. These issues lead to inefficient communication and translate into energy consumption. In order to conserve the battery power, we would like to reduce the number of broadcasts and the scope of the broadcasts.

In this section, we develop a different path computation algorithm that relies on Voronoi diagrams [Guibas and Stolfi 1983]. The Voronoi diagram of a set of points is a partition of space into cells. Each cell consists of the points that are closer to one particular point from a given set than to any others. For example, the Voronoi diagram of two points in the plane is their perpendicular bisector. Voronoi diagrams have been used in robotics for path planning. The edges of the Voronoi diagram define the channels that maximize the clearance to the obstacles. We consider computing such Voronoi paths in the sensor network which will keep the mobile user maximally distant from danger.

Voronoi diagrams can be built in the sensor field in a distributed way (Algorithm 4). Every time an event site changes (appears, disappears, or moves), the network can adapt to the change easily by updating the distance information from every sensor to the event. This distance can be computed as previously, by a broadcast initiated by the event sensor. Each node decides whether it is on a Voronoi edge by comparing its two smallest distances to events. If the two distances are approximately equal to each other, it is on a Voronoi diagram edge defined by the two corresponding event sites. Because we consider approximate distance equality, we have to ensure Voronoi edge connectivity. We can

28   •   Q. Li and D. Rus

Algorithm 4. The Voronoi diagram computation protocol.
1:   **for** all sensors s_i in the network **do**
2:     **if** *sensed-value = danger* **then**
3:       $distance_i = 0$
4:        Broadcast message $(i, distance = 0)$
5:     **if** *receive*$(j, distance)$ *from node k* **then**
6:       $distance_j = min(distance_j, distance + dist(i, k))$ $(dist(i, k)$ *is the distance*
        between s_i and s_k)
7:        *Broadcast message* $(j, distance_j)$
8:     **for** *all received j* **do**
9:        *Find the least two distances* $(j_1, distance_{j_1})$ *and* $(j_2, distance_{j_2})$
        $(distance_{j_1} \leq distance_{j_2})$
10:       **if** $|distance_{j_1} - distance_{j_2}| \leq \epsilon$ **then**
11:         *This node is a Voronoi node and close-danger$_i = j_1$*

Algorithm 5. The safest path from a Voronoi node to goal computation protocol.
1:   Let G be a goal sensor
2:   G broadcasts $message = (G_{id}, my_{id}(G), potential = 0)$
3:   **for** any sensor s_i **do**
4:     **if** *receive*(g, k, pot) **then**
5:       **if** I am not a Voronoi node and *close-danger$_i \neq$ close-danger$_{G_{id}}$* **then**
6:         do nothing
7:       **if** *I am a Voronoi node or close-danger$_i =$ close-danger$_{G_{id}}$* **then**
8:         *Compute the potential integration from the goal to here:*
9:         **if** $P_g > pot + pot(i, k)$ **then**
10:          $pot(i, k)$ is the potential integration from s_i to s_k
11:          $P_g = pot + pot(i, k)$
12:          $prior_g = k$
13:          Broadcast $(G_{id} = g, my_{id}(s_i), P_g)$

tune the threshold for the difference of the two smallest distances to introduce
more nodes on the Voronoi edge. Another method is to use broadcast to connect
the disconnected nodes on the Voronoi edge. Each node that finds itself on a
Voronoi edge initiates a broadcast restricted within a limited number of hops
and finds the paths that connected to other nodes on Voronoi edges with minimal
number of hops. In the following, we look at all the nodes on the Voronoi edges
and on the paths connecting them without discrimination and call them Voronoi
nodes for ease of explanation. A sensor can determine if a neighbor is in the
same cell by comparing their closest event locations. The closest event of a
sensor is the site that has the smallest distance value. If two sensors have the
same closest sites, they are in the same cell. Otherwise, they are not.

   Given a Voronoi diagram embedded as edges in a sensor network, Algorithm 5
and 6 compute the safe path to a goal. The computation is similar to our previ-
ous algorithms except that the broadcast is now constrained to the cells where
the goal and the user reside and to the nodes along Voronoi edges. The net-
work first computes the best path from any node on the Voronoi edges to the
goal (Algorithm 5). The nodes in the goal's cell (including the nodes on the cell
boundary) are searched for the best path from the goal to any node on the cell's
boundary. The protocol keeps running only on the nodes that are on the Voronoi
edges. Eventually all the nodes on the Voronoi edges get the safe path to the
goal.

Algorithm 6. The safest path from the user to goal computation protocol.

1: Let U be a user sensor
2: U broadcasts $message = (U_{id}, pathquery)$
3: **for** any sensor s_i **then**
4:      **if** *receive(U_{id}, path query)* **then**
5:          **if** *close-danger$_i \neq$ close-danger$_{U_{id}}$* **then**
6:              do nothing
7:          **if** *close-danger$_i =$ close-danger$_{U_{id}}$ and I am not a Voronoi edge node* **then**
8:              *broadcast* ($U_{id}, pathquery$)
9:          **if** *close-danger$_i =$ close-danger$_{U_{id}}$ and I am a Voronoi edge node* **then**
10:             broadcast ($g, i, potential$)
11:     **if** *receive(g, k, pot)* **then**
12:         **if** *close-danger$_i \neq$ close-danger$_{U_{id}}$* **then**
13:             do nothing
14:         **if** I am a Voronoi edge node **then**
15:             do nothing
16:         **if** *close-danger$_i =$ close-danger$_{U_{id}}$ and I am not a Voronoi edge node* **then**
17:             *Compute the potential integration from the goal to here:*
18:             **if** $P_g > pot + pot(i, k)$ **then**
19:                 $P_g = pot + pot(i, k)$
20:                 $prior_g = k$
21:                 Broadcast ($G_{id} = g, my_{id}(s_i), P_g$)

When the mobile node and the goal are in different cells, the mobile node first broadcasts a path query message to all the nodes on Voronoi edges of its cell boundary (Algorithm 6). Those nodes keep broadcasting the potential integrations (from goal to the node in question) within the cell. Eventually the user knows the best path to the goal by choosing the path with the smallest potential integration. Therefore, the path from the mobile node to the goal is composed as a path to a node on the Voronoi edge, a path following the Voronoi edges to the goal cell, and a path within that goal cell.

The query protocol is the same as in the previous section. The mobile node pings the nearby sensors for the next location. The correctness of this Voronoi protocol relies on the fact that the mobile node will not get stuck in a local minimum. This is true because (1) the Voronoi diagram edges are connected, (2) all cells have some adjacent Voronoi edges, (3) the path computation is done using dynamic programming on the connected component consisting of the goal cell, Voronoi diagram edges, and the user cell.

The Voronoi protocol has two properties worth noting. Once computed, the Voronoi infrastructure can be used repeatedly to identify paths to any goal without additional network broadcast. Additionally, the searching space for the path is restricted to Voronoi edges and the nodes in the start and goal cells. When the sensor field detects a large number of events, the number of Voronoi nodes increases and so does the search scope. When two event nodes are close to each other, the Voronoi edge between them is not safe. These unsafe edges will be searched but discarded in favor of safer edges by Algorithm 6.

To show the quality of the path computed using this method, we ran simulation experiments. Figure 12 shows a typical result. Figure 13 shows simulation data from a network with 2000 nodes, transmission range 5, and network field
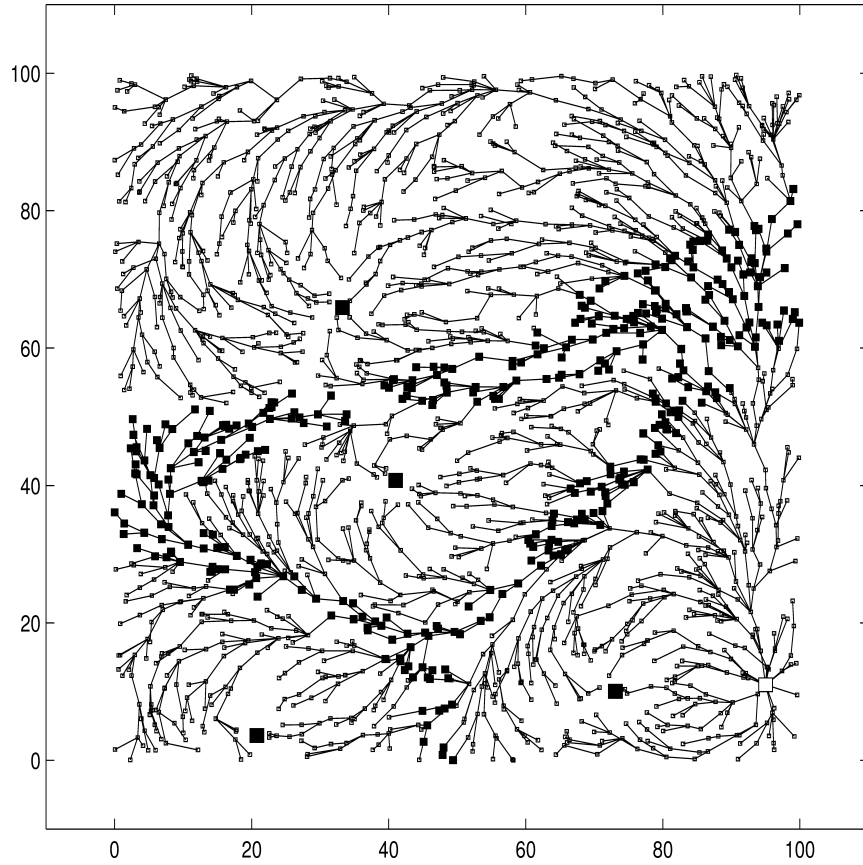
Fig. 12.   The figure shows a typical network configuration and the computed Voronoi diagram and the path from all the nodes to the goal. In the simulation, the network parameters are: number of nodes 2000, transmission range 5, network field size 100×100. The big black points are four dangerous sites and one goal node. The small black points are the nodes on the Voronoi diagram edges. The tiny nodes are the normal nodes. The lines depict the paths to the goal.

size 100×100. We randomly generated 4 event sites and one goal node, and computed the path from any node to the goal. We computed three paths for each node: (1) the optimal path computed by knowing all the positions of the nodes, (2) the path computed by Algorithms 1 and 2, and (3) the path computed by a Voronoi algorithm. Let the potential integrations on the three paths be P_o, P_b, P_v. Figure 13 plots the CDF of the distribution of $\frac{P_v - P_o}{P_o}$ (the solid curve) and $\frac{P_b - P_o}{P_o}$ (the dotted curve). We see that the paths computed using Algorithms 1 and 2 are close to optimal. The performance degradation of the Voronoi scheme is at most 60%. Only 5% of the nodes have degradation of more than 30%, and 20% have more than 4%. 80% of the nodes have less than 4% degradation. We do not intend to compare the exact communication cost here since it is dependent on the communication pattern and packet scheduling scheme. We only look at the search scope because it is a metric that shows how many nodes are involved and sheds light on the communication cost. As to the danger information
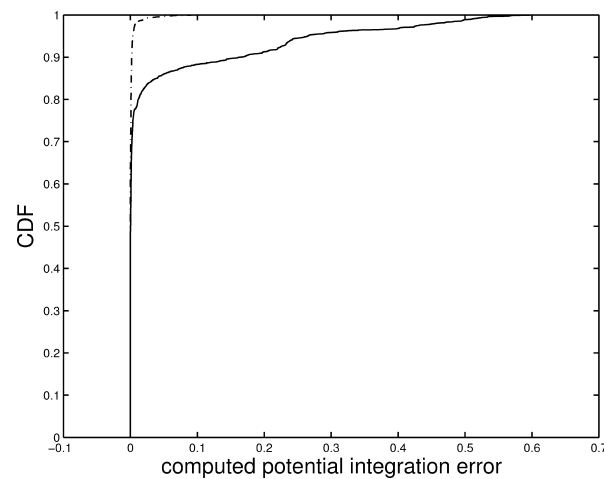
Fig. 13.   The figure shows the CDF (cumulative distribution function) of the error distribution of the potential integration of the path computed using algorithms in Section 3 (dotted curve) and the path computed using Voronoi algorithm (solid curve). In the simulation, the network parameters are: number of nodes 2000, transmission range 5, network field size 100×100.

propagation part, the communication cost of the Voronoi diagram-based scheme is the same as the previous methods.

6. SUMMARY

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 has pointed out some important new directions for research.

We also looked at the variations of our generic navigation protocols. We show that a Voronoi diagram computed in a distributed and localized way can help to reduce the searching space and thus the communication cost.

In the future, we hope to use spanners to approximate the network field and further reduce the search space.

REFERENCES

ASLAM, J., LI, Q., AND RUS, D. 2003. Three power-aware routing algorithms for sensor networks. *Wirel. Comm. Mobile Comput. 3*, 2 (Mar.), 187–208.

BATALIN, M. AND SUKHATME, G. 2003. Efficient exploration without localization. In *the International Conference on Robotics and Automation (ICRA'03)*. Taipei,Taiwan.

BATALIN, M. A. AND SUKHATME, G. S. 2004. Coverage, exploration and deployment by a mobile robot and communication network. *Telecomm. Syst. J. 26*, 2–4.

CAPKUN, S., HAMDI, M., AND HUBAUX, J. P. 2002. GPS-free positioning in mobile ad-hoc networks. *J. Cluster Comput*.

CHENG, X., THAELER, A., XUE, G., AND CHEN, D. 2004. Tps: A time-based positioning scheme for outdoor sensor networks. In *the 23rd Conference of the IEEE Communications Society (INFOCOM '04)*. Hong Kong.

CORKE, P., PETERSON, R., AND RUS, D. 2003. Networked robots: Flying robot navigation using a sensor net. In *the 11th International Symposium of Robotics Research*. Sienna, Italy.

ESTRIN, D., GOVINDAN, R., AND HEIDEMANN, J. 2000. Embedding the internet. *Comm. ACM 43*, 5 (May), 39–41.

FANG, Q., GAO, J., AND GUIBAS, L. 2004. Locating and bypassing routing holes in sensor networks. In *the 23rd Conference of the IEEE Communications Society (INFOCOM'04)*. Hong Kong.

GANESAN, D., KRISHNAMACHARI, B., WOO, A., CULLER, D., ESTRIN, D., AND WICKER, S. 2002. Complex behavior at scale: An experimental study of low-power wireless sensor networks. UCLA Computer Science Tech. Rep. UCLA/CSD-TR 02-0013.

GUIBAS, L. J. AND STOLFI, J. 1983. Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. In *ACM Symposium on the Theory of Computing (STOC'83)*. 221–234.

HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER, D., AND PISTER, K. 2000. System architecture directions for network sensors. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS'00)*.

HUANG, Q., LU, C., AND ROMAN, C. 2003. Spatiotemporal multicast for sensor networks. In *ACM SenSys '03*. Los Angeles, CA.

INTANAGONWIWAT, C., GOVINDAN, R., AND ESTRIN, D. 2000. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the Mobicom'00*. Boston, MA.

KODITSCHEK, D. E. 1989. Planning and control via potential fuctions. *Robotics Rev. I* , 349–367.

LATOMBE, J.-C. 1992. *Robot Motion Planning*. Kluwer, New York.

LENGYEL, J., REICHERT, M., DONALD, B., AND GREENBERG, D. 1990. Real-time robot motion planning using rasterizing computer graphics hardware. In *Proceedings of SIGGRAPH*. Dallas, TX, 327–336.

MEGUERDICHIAN, S., KOUSHANFAR, F., QU, G., AND POTKONJAK, M. 2001. Exposure in wireless ad hoc sensor networks. In *MOBICOM '01*. Rome, Italy. 139–150.

MOORE, D., LEONARD, J., RUS, D., AND TELLER, S. 2004. Robust distributed network localization with noisy range measurements. In *ACM SenSys'04*. Baltimore, MD.

NICULESCU, D. AND BADRINATH, B. R. 2003. Ad hoc positioning system (APS) using AOA. In *the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)*. San Francisco, CA.

SAVVIDES, A., HAN, C.-C., AND STRIVASTAVA, M. B. 2001. Dynamic fine-grained localization in ad-hoc networks of sensors. In *MOBICOM '01*. Rome, Italy, 166–179.

SUNDARAM, N. AND RAMANATHAN, P. 2002. Connectivity based location estimation scheme for wireless ad hoc networks. In *Proceedings of Globecom '02*. Taipei, Taiwan.

TAKAGI, H. AND KLEINROCK, L. 1984. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Trans. Comm. 32*, 3 (Mar.).

VELTRI, G., HUANG, Q., QU, G., AND POTKONJAK, M. 2003. Minimal and maximal exposure path algorithms for wireless embedded sensor networks. In *ACM SenSys '03*. Los Angeles, CA.

WAN, C.-Y., CAMPBELL, A., AND KRISHNAMURTHY, L. 2002. PSFQ: A reliable transport protocol for wireless sensor networks. In *the 1st ACM International Workshop on Wireless Sensor Networks and Applications*. Atlanta, GA.

WAN, C.-Y., EISENMAN, S. B., AND CAMPBELL, A. T. 2003. Coda: Congestion detection and avoidance in sensor networks. In *ACM SenSys '03*. Los Angeles, CA.

WOO, A., TONG, T., AND CULLER, D. 2003. Taming the underlying challenges of reliable multihop routing in sensor networks. In *ACM SenSys '03*. Los Angeles, CA.

YAN, T., HE, T., AND STANKOVIC, J. A. 2003. Differentiated surveillance for sensor networks. In *ACM SenSys '03*. Los Angeles, CA.

YE, F., LUO, H., CHENG, J., LU, S., AND ZHANG, L. 2002. A two-tier data dissemination model for large-scale wireless sensor networks. In *ACM Mobicom '02*. Atlanta, GA.

ZHAO, F., SHIN, J., AND REICH, J. 2002. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Process. Mag. 19*, 2 (Mar.), 61–72.

ZHAO, J. AND GOVINDAN, R. 2003. Understanding packet delivery performance in dense wireless sensor networks. In *ACM SenSys '03*. Los Angeles, CA.