Distributed Algorithms for Guiding Navigation across a Sensor Network

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

ABSTRACT

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

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Algorithms, Design, Experimentation, Measurement, Performance

Keywords

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.

MobiCom'03, September 14–19, 2003, San Diego, California, USA.

A collection of active sensor networks can follow the movement of a source to be tracked, for example a moving vehicle. It can guide the movement of an object on the ground, for example a surveillance robot. Or it can focus attention over a specific area, for example a fire to localize its source and track its spread.

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

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

Current work in reactive routing protocols (or directed diffusion) are aiming for the network communication. They cannot provide the navigation information to the user in the sensor field. We combine robotics and networking. 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 paper contributes: (1) an interesting 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.

More specifically, we build on important previous work by [17, 11, 28, 7] and examine in more detail reactive sensors that can adapt to their environment by capturing a danger level map and distributing this map across the network. We represent the danger detected by the sensors as "obstacles" in the network and compute the artificial potential field that corresponds to the current state. We then develop a distributed protocol that combines this artificial potential field with information about the direction and goal of the moving object and guarantees the best safest path to the goal. By safest path we mean the path with the largest clearance of the danger zones. We also develop a protocol for distributing the information in the sensor network, such as the danger map and shortest paths. We then show how sensors equipped with limited memory can cooperate to hold and retrieve information about the network. Finally, we discuss an implementation of our protocols on a real sensor network consisting of 50 Mote sensors [10] and present our experimental data.

2. RELATED WORK

We are inspired by previous work in sensor networks [6], ad-hoc networks [12, 13, 9, 18, 21, 5, 4, 24], and robotics [15]. Our experimental work is done with the Mote hardware [10].

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

Ye et al. [28] 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.

Meguerdichian et al. [17] considered the minimal exposure path problem in a sensor network. They developed an efficient and effective algorithm for the problem. 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, especially two recent papers that are closely related to our system implementation. An empirical study on networks composed of over 150 Motes was conducted in [7]. The paper presents the data collected in different layers and reveals that even a simple protocol can exhibit a large complexity in the mote network. It gives many very useful experimental data on a real sensor network platform. Some of the observations from our experiments show the same behaviors in many scenarios. Wan and Campbell et al. [27] proposed PSFQ (Pump slowly, Fetch Quickly), a reliable transport protocol in wireless sensor networks. This paper addresses some problems that we encountered in our system implementation.

Gupta and Kumar [8] researched the capacity bounds of a large scale ad-hoc network. Scaglione and Servetto [23] showed an approach to work around the vanishing per-node throughput problem by coupling routing and source coding in a sensor network.

We use the number of hops to evaluate the distance between sensors. The similar method was used in [19]. Papers working on location in ad-hoc networks include [3, 22, 20, 25].

The application developed in this paper uses techniques from robotics, where a key problem is how to plan the motion of moving robots. A good overview of motion planning in robotics is given by [15]. [16] proposed a robot motion planner that rasterizes configuration space obstacles into a series of bitmap slices, and then use dynamic programming to compute the distance from each point to the goal and the paths in this space. This method guarantees that the robot finds the best path to the goal. [14] discusses the use of an artificial potential field for robot motion planning. A robot moving in accordance to the potential will never hit obstacles, but it may get stuck in local minima. We combine these two methods to find the best path to the goal, which is safe and short, and modify them to exploit the distributed nature of sensor networks. Another related work by Batalin and Sukhatme [2] is to address the problem of coverage and exploration of an unknown dynamic environment using a mobile robot by using beacons. The beacons (markers) that form a communication network are used a support infrastructure to aid exploration of the mobile robot.

3. A DISTRIBUTED ALGORITHM FOR GUIDING THE NAVIGATION OF A USER

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

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

There are numerous solutions to motion planning in the presence of obstacles and uncertainty. For a good survey of the techniques see [15]. We seek a solution that lends itself naturally to the discrete nature of sensor networks. In [16], Donald et al. describe an optimal solution for motion planning when the map of the world is given. The first step of the solution is to rasterize the configuration space obstacles into a series of bitmap slices. Dynamic programming is then

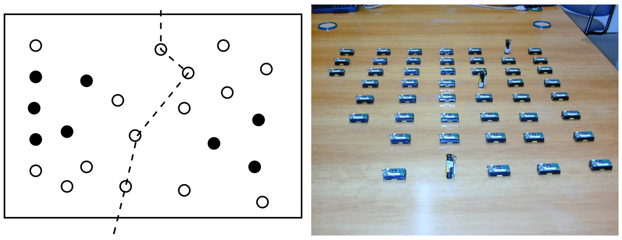


Figure 1: The left figure shows a typical setup for the navigation guiding task. The solid black circles correspond to sensors whose sensed value is “danger”. The white circles correspond to sensors that do not sense danger. The dashed line shows the guiding path across the area covered by the sensor 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 2 obstacles (that is, danger areas) and one goal.

used to calculate the optimal path in this space. Although this method can not be applied directly, it can be adapted for sensor networks. Although the map is not immediately available, the motion planning algorithm fits a sensor network well in two ways. First, the sensors can be regarded as the bitmap pixels. Second, the dynamic programming component of the algorithm can be implemented by using the sensor communications.

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 [15].

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 the danger to its neighbors in a message that includes its source node id, the potential value, and the number of hops from the source of the message to the current node. When a node receives multiple messages from the same source node, it keeps only the message with the smallest number of hops. (The message with the least hops is kept because that message is likely to travel along the shortest path.) The current node computes the new potential value from this source node. The node then broadcasts a message with its potential value and number of hops to its neighbors.

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

Algorithm 1 The potential field computation protocol.

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

Algorithm 2 The safest path to goal computation protocol.

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

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

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

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

A user of the sensor network can rely on the informa-

Algorithm 3 The navigation guiding protocol.

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

tion 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.

3.1 Implementation Issues

Our navigation algorithms have an implicit assumption that the communication paths in the network are bi-directional. 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 4.4) has taught us that not all links in sensor networks are bi-directional. For example see Figure 2 that shows the distribution of symmetric and asymmetric links in an experiment with a 7x7 grid of Mote sensors. This is consistent with data from [7]. We propose the following method for identifying the bi-directional links in the network. The computation can be thought of as an additional protocol run by each node.

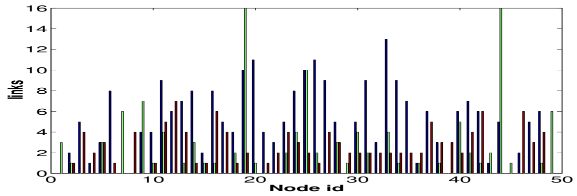


Figure 2: The distribution of symmetric and asymmetric links in one experiment. The x-axis shows the node id and the y-axis the number of links. For each node we have three bars: the first shows the number of symmetric links, the second is the number of unidirectional outgoing links and the third the number of unidirectional incoming links.

Each node does neighbor profiling to find all its stable one-hop neighbors bi-directionally; 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. $1/5$ is a parameter we chose for our experiments.

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 bi-directional links. Our hop distance can also be close to average instead of too abnormal.

Algorithms 1 and 2 ask each sensor to broadcast upon receiving a message with fewer hops to the dangerous area or a smaller potential integration to the goal. Many broadcasts may not be necessary since only the message with the least hops to the danger 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 it broadcasts. The waiting time for sensor s_i is proportional to one unit in Algorithm 1 and the value pot_i in Algorithm 2. 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 non-optimal 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 [1]. 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 congestions due to the simultaneous transmissions.

In order to desynchronize the nodes in a proximity that would, upon the reception of a packet, simultaneously broadcast the packet, we also add random variable waiting time to each node to reduce the contention.

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.

3.2 Analysis

3.2.1 *Correctness*

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

THEOREM 1. *Algorithm 3 will always give the user sensor a path to the goal.*

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

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

3.2.2 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 when that each node has a constant transmission range, which is an assumption consistent with our testbed hardware.

The key metric is the minimal number of hops between any two sensors that are l distance apart. 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, it is very hard to characterize this metric. 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.

In [26], Takagi and Kleinrock proposed the most forward routing and analyzed its average progress in the direction of the destination. We can use the same analysis to approximate the distance of a single hop.

Suppose the average progress be R' by using the analysis in [26] and the transmission range be R. Then the minimal ideal hops should be l/R, but the expected minimal hops in our real sensor network is l/R'. That is, the distance we evaluate is always R/R' times of the real distance.

In [7], Ganesan et al. reported the length of a hop may not be fixed, as we observed in our experiments. By experiments, we can get the expectation and the deviation of the length of a hop (call them E and d). According to 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 actually shows that our algorithms are robust in the real network scenario.

3.2.3 Performance Bound of the Computed Path

We expect our protocols to compute the integrated potential value on the safest path, but the implementation introduces error. 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.

THEOREM 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 $\overline{s_0s_1}, \overline{s_1s_2}, \cdots, \overline{s_{k-1}s_k}$ (or $\overline{s_0s_1s_2\cdots s_{k-1}s_k}$) 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 upon the entire path, we have the following. $P_2 = \int_A^B \sum_j p_{tj} = \sum_{i=0}^{k-1} \int_{s_i}^{s_{i+1}} \sum_j p_{tj} \leq \sum_{i=0}^{k-1} \int_{s_i}^{s_{i+1}} \sum_j (\frac{h_{ij}^2}{(h_{ij}-1)^2} \cdot p_{ij}) \leq \sum_{i=0}^{k-1} R \cdot (\sum_j \frac{h_{ij}^2}{(h_{ij}-1)^2} \cdot p_{ij})$ (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 $|\overline{s_{i-1}s_i}| + |\overline{s_is_{i+1}}| \geq R$. Let's find s_i' on $\overline{s_is_{i+1}}$ such that $|\overline{s_{i-1}s_i}| + |\overline{s_is_i'}| = R/2$ or find s_i' on $\overline{s_{i-1}s_i}$ such that $|\overline{s_i's_i}| + |\overline{s_is_{i+1}}| = R/2$. Without loss of generality, we assume s_i' is on $\overline{s_is_{i+1}}$. The distance from any point on $\overline{s_{i-1}s_i}$ or $\overline{s_is_i'}$ 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 above procedure. It follows that $P_2 = \int_A^B \sum_j p_{tj} = \sum_{i=0}^{k-1} \int_{s_i'}^{s_{i+1}'} \sum_j p_{tj} \geq \sum_{i=0}^{k-2} \int_{s_i'}^{s_{i+1}'} \sum_j (\frac{h_{ij}^2}{(h_{ij}+1)^2} \cdot p_{ij}) \geq \sum_{i=0}^{k-2} \frac{R}{2} \cdot (\sum_j \frac{h_{ij}^2}{(h_{ij}+1)^2} \cdot p_{ij})$.

If $\frac{h_{ij}^2}{(h_{ij}+1)^2} \geq q_2$ 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 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 $\overline{u_0u_1\cdots u_l}$ 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. *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 $|\overline{s_{2y-2}u_e \cdots u_p s'_{2y-1}}| \ge R/2$, and all the points on segments $\overline{s'_{2y-1}u_r \cdots s_{2y-1}}$ are within R distance from s_{2y-1}, and $|\overline{s'_{2y-1}u_r \cdots s_{2y-1}}| + |\overline{s_{2y-1}s_{2y}}| \ge R/2$ (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 $|\overline{s_{2y-2}u_e \cdots u_p s'_{2y-1}}| \ge R/2$, $|\overline{s'_{2y-1}u_r \cdots s_{2y-1}s_{2y}}| \ge R/2$, 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 $|\overline{s_{2y-2}u_e \cdots u_f s_{2y-1}s'_{2y-1}}| = R/2$. 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 $R/2$. $P_0 = \sum_{i=0}^{l-1} \int_{u_i}^{u_{i+1}} \sum_j p_{tj} \ge \sum_{i=0}^{\lfloor k/2 \rfloor - 1} (\int_{s_{2i-2}}^{s'_{2i-1}} \sum_j (\frac{h^2_{2i-2j}}{(h_{2i-2j}+1)^2} \cdot p_{2i-2j}) + \int_{s'_{2i-1}}^{s_{2i}} \sum_j (\frac{h^2_{2i-1j}}{(h_{2i-1j}+1)^2} \cdot p_{2i-1j})) \ge \sum_{i=0}^{k-2} \frac{R}{2} \cdot (\sum_j \frac{h^2_{ij}}{(h_{ij}+1)^2} \cdot p_{ij})$.

If $\frac{h^2_{ij}}{(h_{ij}+1)^2} \ge q_0$ for all i, j, and $\sum_j \frac{h^2_{k-1j}}{(h_{k-1j}+1)^2}$ 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 $P_2 \le \frac{2q_1}{q_0} P_0$, i.e., our computed path has bounded potential integration. □

3.2.4 Propagation and Communication Capability

Two natural questions arise about the protocols we described previously: How much time does it take to propagate the obstacle and goal information? 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. It is no surprising they can provide performance gain to our system.

We assume that each node has fixed transmission range and the nodes in a node's neighborhood (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; i.e., 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 when we add waiting time for each broadcast; i.e., 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 this node, so that they do not collide. For the case without explicit waiting time scheme, 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 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 amount of bits. If we have some constraints on the time, say, we have 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 that b/k, i.e., $ov/d < b/k$. 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 moving object is a vehicle moving at a speed of 30 miles, the vehicle updates can happen every 2.7 meters.

4. EXPERIMENTS

We have implemented the algorithms described in Section 3 using the Mote MOT300 sensors [10].

4.1 Correctness Validation

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

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

A first experiment was designed to show empirically that the protocols work and are correct. In this first experiment we used a grid of 12 first generation Motes. The Motes were approximately on a line with several nodes around the obstacles in order to test if the safest path is a detour around the obstacle. All neighbors are within communication range. The application is 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 assume that the nodes know their location and that it can be transmitted to the moving object/user. This can be done by augmenting Motes with a GPS location, or via triangulation. Since we have not done this augmentation of the hardware yet, we simulate location knowledge by placing the Motes in a grid pattern and supplying coordinates. The potential field and goal path computations are run by the network continuously.

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

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

When a 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.

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.

When a new obstacle is inserted in the network, the obstacle starts broadcasting its danger information which affects the information held by each node. At this point Algorithms 1 and 2 cause the local information to change. We call the total time for the network to identify the new distances from danger and to the goal for each node the *time for the network to stabilize.* In other words, the time for the network to stabilize is the information propagation time in the network, which depends on the maximal hops from the goals or the obstacles to any node in the network. When an obstacle is added to the system online, it takes an identical amount of time to diffuse the information to the whole network.

Fig. 3 shows the comparison between the measured real distance and the hops counted using our algorithm. The data was collected in our 7x7 grid network. We can see the measured real distance is approximately linear in the number of hops. Fig. 4 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 with the distance. Compared with a dangerous path, which has potential integration of 3-5, the computed path is quite close to the safest path.

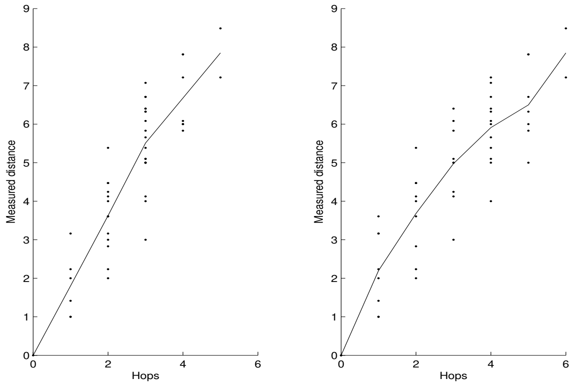


Figure 3: This figure shows the comparison between the measured real distance and the hops counted using our algorithm.

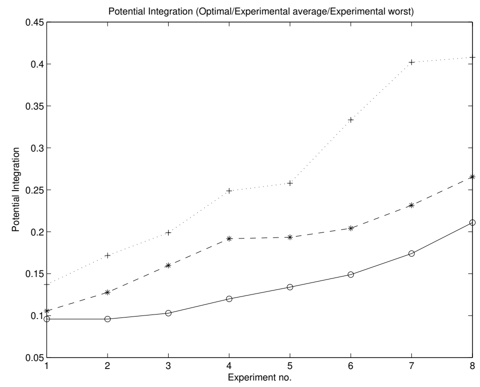
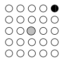
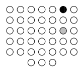
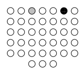
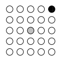
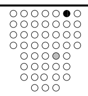
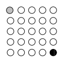
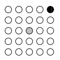
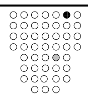
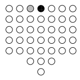
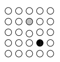
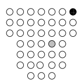
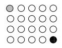
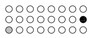
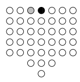
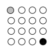
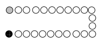
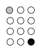
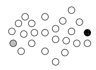


Figure 4: This figure shows the potential integrations in 54 experiments with eight different network topologies. The solid, dashed, and dotted lines are the potential integrations on optimal path, average over all computed paths, and the worst computed respectively.

4.2 Measuring Adaptation

We have implemented the protocols in Algorithms 1, 2, and 3 on the second generation Motes MOT300. In this

Table 1: The data that summarizes timing measurements for several experiments with a sensor network consisting of Mote sensors. All network topologies are summarized as geometric icons and all measurements are in seconds.

Exp. Config.	danger prop.	Shortest distance	goal prop.	safest path	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		4.20	9.10	1.37	22.63
	0.20	2.17	0.13	4.23		5.20	20.30	1.17	9.40		1.20	20.23	2.13	3.13
	1.16	3.13	2.13	10.03		1.30	16.23	1.10	2.17		0.17	6.17	0.04	19.37
	0.10	3.10	0.10	1.07		9.37	17.37	0.33	8.43		0.20	7.17	1.13	3.1
	0.23	1.33	0.13	1.07		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

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

second experiment, we used a 50 Mote MOT300 testbed. We arranged the nodes in the given topology and gave each node position information (which could be obtained using a GPS extension of the hardware.) We ran a suite of different network topologies and measured the network stabilization time when obstacles and goals are injected on-line in the network. Tables 1 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 experiments has been to determine on how quickly the network responds to the environmental change, specifically new danger sources and goal changes.

We ran all the experiments on a large table in our lab, as shown in Figure 1(right). For each experiment, we set the transmission range to be very small (9"). In all these second round experiments we focused on the network as a whole and did not use a base station (thus, we did not collect data in a central place.) To collect timing data, we used two procedures: the videotaping procedure and the logging procedure.

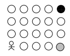
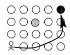
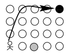
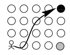
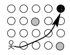
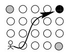
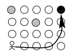
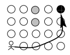
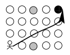
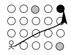
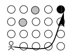
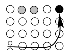
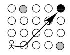
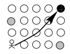
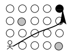
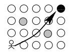
We used the videotaping procedure to capture the global behavior of the sensor node. The Mote LEDs were programmed to capture the state of the Mote. We recorded the experiment with a Sony video camera at a rate of 30 frames per second. We then analyzed the resulting video to capture the timing measurements—which gave us a resolution of $1/30^{th}$ of a second. We looked at the video sequence frame by frame and kept track of when and which LED triggered. Since the overall timings for the navigation algorithms are on the order of seconds, we believe our methodology is accurate enough.

We analyzed four metrics for each experiment: the time for the danger information to propagate from the danger/obstacle sensor to the whole network, the time for all the nodes in the network to obtain their shortest distance to the dangerous areas, the time for the goal information to propagate to the whole network, and the time for all the nodes in the network to obtain their safest path to the goal. Tables 1 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 2 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.

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

Table 2: The data of the response time for several experiments with a sensor network consisting of Mote sensors. All network topologies are summarized as geometric icons and all measurements are in seconds.

Exp. No.	Exp. Config.	Response Time	Exp. No.	Exp. Config.	Response Time	Exp. No.	Exp. Config.	Response Time
0		0	1		3.63	2		6.83
3		3.60	4		4.93	5		2.13
6		1.73	7		2.23	8		4.43
9		5.33	10		4.43	11		9.63
12		6.27	13		4.50	14		4.87
12		6.70						

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

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.

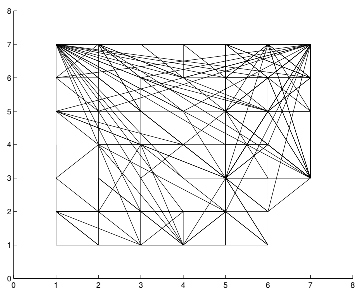


Figure 5: Measured communication graph of the experimental 7x7 grid network. Notice the absence of many point to point links we expect to be available (from example from (1,1) to (2,1)) and the presence of long links we do not expect to have, for example from (1,7) to (7,6).

Nodes were configured to log records of the packets sent and received, corresponding time, and related internal events. The network was a 7x7 grid with 49 Motes evenly placed on the grid. The neighboring Motes were spaced 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

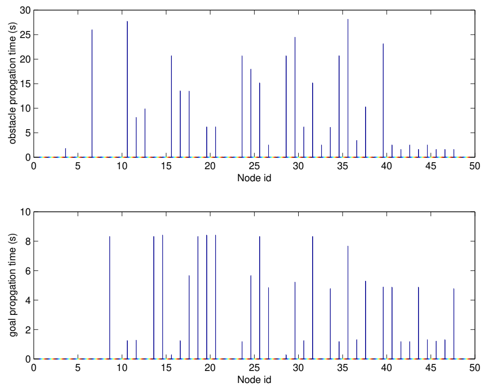


Figure 6: The obstacle and goal propagation time distribution. In our experiment, the nodes were arranged in a 7x7 grid, with obstacles placed at (1,1) and (7,7). The nodes on x-axis are sorted according to the Manhattan distance to (1,1), while the y-axis shows propagation time (in seconds). shown on the y-axis.

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.

Fig. 5 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

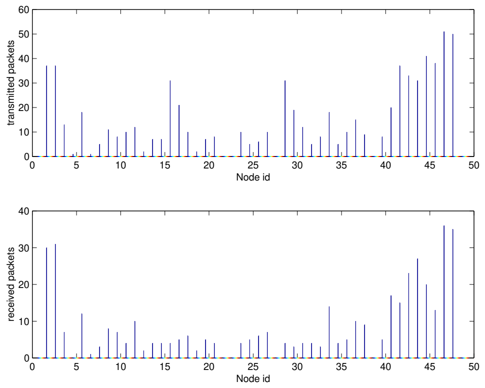


Figure 7: Transmission count distribution. The nodes on x-axis are sorted according to the Manhattan distance to (1,1), while the y-axis represents the number of packets (send/receive) for that node.

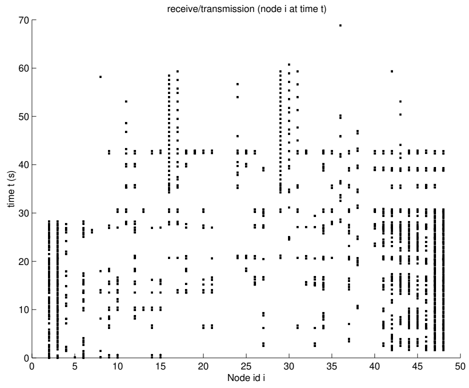


Figure 8: Transmission/reception of packets by individual nodes over time. The bottom part signifies the propagation of the information from the two obstacles (1 and 49) in the first period of the experiment when only the obstacles were turned on. The top part is the goal information propagation in the second half of the experiment when only the goal was turned on..

the connection graph is. Note the Motes in the right corner which are completely disconnected.

Fig. 6 (top) shows the obstacle information propagation time, that is, the time when a mote receives a stable potential value. Some Motes get the 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 Fig. 6 (bottom), which shows the goal propagation time, defined as the time for a mote to get a stabilized integration value to the goal.

Fig. 7 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.

Fig. 8 plots the 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. 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 to be 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.

4.3 Performance Optimization

We optimized the message broadcasts using the methods described in section 3.1 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 7x7 grid as ones in section 4.2. 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 (Fig. 10). 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 (Fig. 11). Compared to our previous scheme, we see much better 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 less packets were suppressed because of the congestion and all the nodes had quite large probability to broadcast their best computed value to the network.

3. The packet send/receive analysis (Fig. 12). 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 large 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.

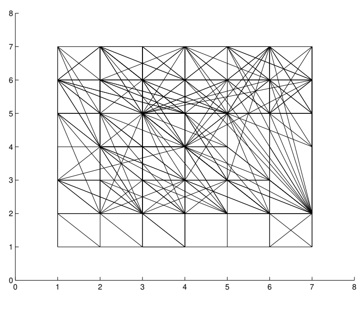


Figure 9: Measured communication graph of the experimental 7x7 grid network.

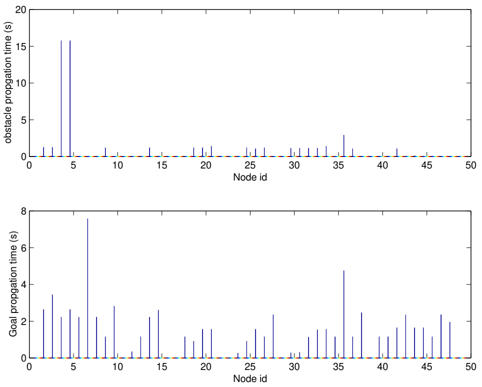


Figure 10: The obstacle and goal propagation time distribution as a function of the sensor node id.

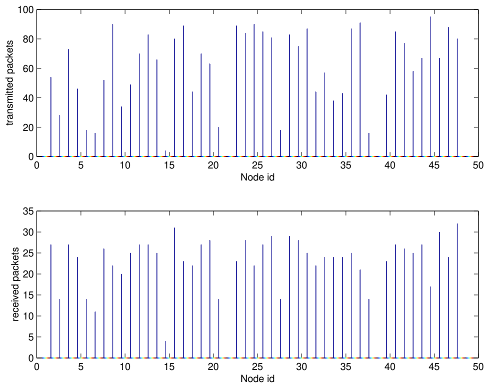


Figure 11: Transmission count distribution.

4.4 Lessons Learned

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

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

1. *Data loss.* Data loss is not rare in sensor 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.

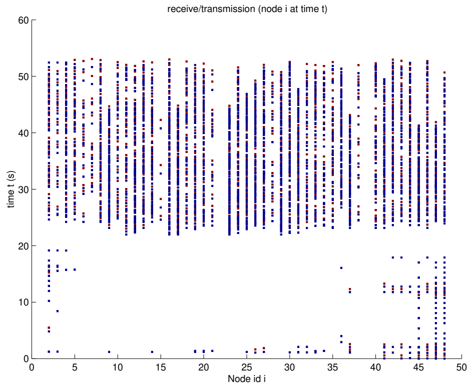


Figure 12: Transmission/reception of packets by individual nodes over time.

3. *Congestion.* Network congestion is very likely when the message rate is high. This is aggravated when the nodes in proximity of each other try to send packets at the same time. For a sensor network, because of its small memory and simplified protocol stack, congestion is a big problem.

4. *Other unpredictable network conditions.* In our sensor networks nodes that should be several hops away from each other occasionally come in direct communication range. We expect many transitory links (on and off) in a unstable network due to the impact of the unpredictable conditions.

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

5. USING SENSOR NETWORKS TO DISTRIBUTE INFORMATION

Section 3 provided an example for how to use a sensor network as a distributed information repository about the environment in the context of a navigation guiding application. In this section we examine in more detail how to represent the information needed by our algorithm effectively in a sensor network. We thus examine the use of a sensor network as a distributed information repository.

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

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

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

In order to address the information storage question, consider the proximity area S covered by a group of sensors. All local (environmental) information about S is the same for all these sensors. To use Algorithm 3, at least one of the sensors in S must store information about the goals. Let $\lambda \cdot S$ be the number of sensors in the area where λ is the density of the sensor distribution and S is the area of the field in question. Suppose each sensor has memory m. Then $m\lambda S$ is the total memory across all sensors. Let the amount of information to be recorded be $\sum m_i$ where m_i is the size of information i. If $m\lambda S \geq \sum m_i$, then it is possible that in the proximity area S, all the required information can be found locally using Algorithm 3. To achieve this information distribution when the amount of information is too large for a node's memory (that is, $m < \sum m_i$), we can use a random, independent and distributed method to store the information on each sensor. Each sensor node randomly keeps a piece of information with probability $p = \frac{m}{\sum m_i}$. When it receives a piece of information, the probability that the information can be found in this area is $1 - (1-p)^{\lambda S}$ (see Figure 13). Currently, we are also exploring some other approaches to cooperative caching data among proximity sensors.

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

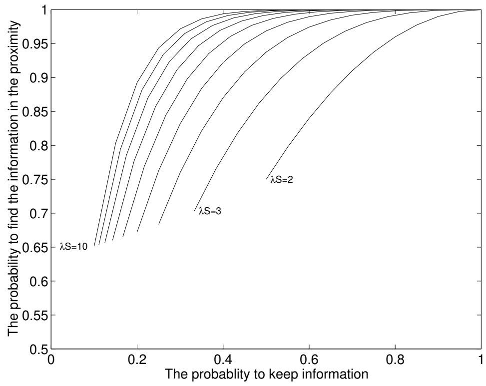


Figure 13: This figure shows the probability (Y-axis) that a piece of information can be found in S, some neighborhood of a sensor. The X-axis is the probability that the sensor keeps a piece of information. We plot for various numbers of sensors in the area from $\lambda S = 2$ to $\lambda S = 10$ where λS is the number of sensors in that area. As the number of the sensors increases, the probability to find some information in that area is close to 1 even though the probability that a sensor keeps the information is small.

6. CONCLUSION

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

Acknowledgments

We are very grateful to Deborah Estrin for introducing us to the Motes hardware. We thank the TinyOS group for providing great support. We are also grateful to Ron Peterson for his invaluable help and discussions. We thank all the reviewers for their helpful comments and Professor Suresh Singh for shepherding this paper.

Support for this work has been provided in part by the Institute for Security Technology Studies (ISTS), NSF award 0225446, I1S-9912193, ONR award N00014-01-1-0675, and DARPA TASK program award F30602-00-2-0585.

Algorithm 4 Sensor information query algorithm

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

7. REFERENCES

[1] Javed Aslam, Qun Li, and Daniela Rus. Three power-aware routing algorithms for sensor networks. *Wireless Communications and Mobile Computing*, 3(2):187–208, March 2003.

[2] M. Batalin and G.S. Sukhatme. Efficient exploration without localization. In *ICRA*, Taipei, May 2003.

[3] Srdan Capkun, M. Hamdi, and J. P. Hubaux. GPS-free positioning in mobile ad-hoc networks. *Journal of Cluster Computing*, April 2002.

[4] Jae-Hwan Chang and Leandros Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *Proc. IEEE INFOCOM*, Tel Aviv, Israel, Mar. 2000.

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

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

[7] Deepak Ganesan, Bhaskar Krishnamachari, Alec Woo, David Culler, Deborah Estrin, and Stephen Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. In *UCLA Computer Science Tech. Report 02-0013*, 2002.

[8] Piyush Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, IT-46(2):388–404, March 2000.

[9] Z. J. Haas. A new routing protocol for the reconfigurable wireless network. In *Proceedings of ICUPC'97*, pages 562 –566, San Diego, Oct. 1997.

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

[11] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. of Mobicom*, Boston, August 2000.

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

[13] Y. B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Proceedings of ACM Mobicom*, pages 66 – 75, 1998.

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

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

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

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

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

[19] Radhika Nagpal, Howard Shrobe, and Jonathan Bachrach. Organizing a global coordinate system from local information on an ad hoc sensor network. In *IPSN LNCS 2634*, pages 333–348, Palo Alto, April 2003.

[20] Dragos Niculescu and B. R. Badrinath. Ad hoc positioning system (APS) using AOA. In *INFOCOM*, San Francisco, CA, April 2003.

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

[22] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Mobicom*, pages 166–179, Rome, July 2001.

[23] Anna Scaglione and Sergio Servetto. On the interdependence of routing and data compression in multi-hop sensor networks. In *ACM Mobicom*, Atlanta, GA, 2002.

[24] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad-hoc networks. In *Mobicom*, pages 181–190, Dallas, TX, Oct. 1998.

[25] N. Sundaram and P. Ramanathan. Connectivity based location estimation scheme for wireless ad hoc networks. In *Proceedings of Globecom*, Nov. 2002.

[26] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, COM-32(3), March 1984.

[27] Chieh-Yih Wan, Andrew Campbell, and Lakshman Krishnamurthy. PSFQ: A reliable transport protocol for wireless sensor networks. In *the 1st ACM International Workshop on WSNA*, Atlanta, GA, September 2002.

[28] Fan Ye, Haiyun Luo, Jerry Cheng, Songwu Lu, and Lixia Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *ACM Mobicom*, Atlanta, GA, 2002.