Proc. 2002 Australasian Conference on Robotics and Automation
Auckland, 27-29 November 2002

Interacting with a Sensor Network

Ron Peterson
Department of Computer Science
Dartmouth, Hanover, NH, USA
rapjr@cs.dartmouth.edu

Daniela Rus
Department of Computer Science
Dartmouth, Hanover, NH, USA
rus@cs.dartmouth.edu

Abstract

We develop distributed algorithms for sensor networks that respond by directing a target (robot or human) through a region. The sensor network models the event levels sensed across a geographical area, adapts to changes, and guides a moving object incrementally across the network. We describe a device we call a *Flashlight* for interacting with the sensor field. This interaction includes collecting navigation information from the sensors in the local neighborhood, activating and deactivating specified areas of the sensor network, and detecting events in the sensor network. We report on hardware experiments using a physical sensor network consisting of Mote sensors.

1 Introduction

We wish to create more versatile information systems by using autonomous and distributed sensor networks: thousands of small sensors, equipped with limited memory and actuation capabilities will autonomously organize themselves and move to track a source and convey information about its location to a human user, and to the rest of the team. Such distributed active mobile sensor networks are pervasive 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 it. A collection of active sensors can follow the movement of the source to be tracked, for example a chemical plume as it spreads in the air, a fire to localize its source, or a herd of sheep grazing on the Taihape farms.

An ad-hoc network is formed by a group of mobile hosts upon a wireless local network interface. It is a temporary network formed without the aid of any established infrastructure or centralized administration. 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 communication protocols for static sensor networks. Often the network topology is unknown and the network has to discover the best route for a packet.

In this paper we focus on mobile sensor networks, where each sensor node is capable of actuation, sensing, and communication. We examine the user interaction with a massively distributed sensor network. The user may be a robot or a human traversing the network.

More specifically, we build on important previous work by [Cerpa and Estrin, 2002; Xu *et al.*, 2001; Wattenhofer *et al.*, ; Ramanathan and Hain, 2000; Chu *et al.*, 2002] and examine in more detail sensor networks that provide directions to a moving user. We developed a device we call a *Flashlight* for interacting with the sensor field. This interaction includes collecting navigation information from the sensors in the local neighborhood, activating and deactivating specified areas of the sensor network, and detecting events in the sensor network. We describe the Flashlight and present protocols for each of these tasks. Finally, we discuss an implementation of our Flashlight protocols on a physical sensor network consisting of 48 Mote sensors [Hill *et al.*, 2000; 2001] and present our experimental data.

2 Motivation: A Distributed Protocol for Guiding Navigation

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. Users of the network (robots or people) 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 interesting areas of the sensor network are those where sensors have triggered. They can be represented as obstacles.

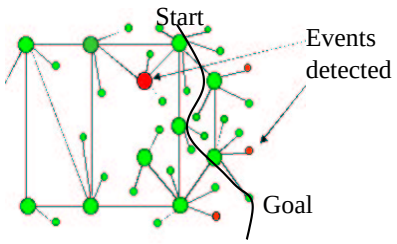

Figure 1: Navigating along the safest path as computed incrementally by sensors in a sensor network.

Such areas may include excessive heat (from volcanoes, fire, etc), people, etc. We assume that each sensor can sense the presence or absence of such an event. An event configuration protocol run across all the nodes of the network creates the event 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 the event each node is. If the sensors are uniformly distributed, *the smallest number of communication hops to a sensor that triggers "yes" to the event* is a measure of the distance. The goal is to find a path for the moving object that moves toward the events (or avoids them, depending on the application.) The user may ask the network regularly for where to go next. The nodes within broadcasting range from the user supply the next best step.

Inspired by [Lengyel *et al.*, 1990], we developed several protocols for the distributed guidance problem across sensor networks and reported the details of these algorithms in [Li *et al.*, 2003]. The map can be constructed incrementally and adaptively as an artificial potential field using hop-by-hop communication (see Figure 1). The "obstacles" correspond to events and have repulsing values and the goal has an attracting value. The potential field is computed in the following way. Each node whose sensor triggers "event" diffuses the information about the event 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. This message is used to update the potential value at the current node. The node then broadcasts a message with its new potential value and number of hops to its neighbors.

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 identified with a distributed protocol using dynamic programming. In [Li *et al.*, 2003] we prove that our algorithm does not get stuck in local minima. A user of the sensor network can get continuous feedback from the network on how to traverse the area. The user asks the network for where to go next. The neighboring nodes reply with their current values. The user sensor chooses the best possibility from the returned values.

3 An Interaction Device: The Flashlight

The navigation guidance application is an example of how simple nodes distributed over a large geographical area can assist with global tasks. This application relies on the ability of the network user to interact with the network as a whole and with specific nodes in the network. This interaction is directed at retrieving data from the network (such as collecting local information from individual nodes and collecting global maps from the network) and injecting data into the network (such as configuring the network with a new task or reprogramming its nodes).

The ability to re-task and reposition sensors in a network by sending state changes or uploading new code greatly enhances the utility of such a network. It allows different parts of the network to be tailored to specific tasks, capabilities to be added or changed, and information to be stored in the nodes in the network. When robots or people interact with the network, the sensors become an extension of the user capabilities, basically extending their sensory systems and ability to act over a much large range.

We have developed a hand-held device that allows a user of the network (a human or a robot) to interact with the network as a whole or to talk to individual nodes in the network. This device is called a *sensory Flashlight* and is based on the optical flashlight metaphor. When pointed in a specific direction, the Flashlight collects information from all the sensors located in that direction and provides its user with sensory feedback. The device can also issue commands to the sensors in that direction.

Applications of the Flashlight device for interacting with the sensor network include: (1) Guiding robots or people along paths that may change over time; (2) Reconfiguring a wireless sensor network in a patterned way; (3) Interacting with a wireless sensor network, both consuming and providing information stored within the network, changing and reacting to its topology, re-tasking the network; (4) Invisible markup of a geographic region with information; (5) Sensor management; and (6) Efficiency improvements in message routing.

In this section we describe the Flashlight hardware and illustrate its capabilities with algorithms for three tasks: (1) using the Flashlight to activate or deactivate a specified area of the sensor network; (2) using the Flashlight to detect events in the sensor network; and (3) using the Flashlight to provide guidance feedback across the sensor network. We also describe the implementation of these algorithms and present experimental data.

3.1 The Hardware

The Flashlight prototype we designed and built is shown in Figure 2(left). This device can be carried by a human user or placed on a mobile robot (or flying robot) to interact with a sensor field. The *beam* of the Flashlight is sensor-to-sensor, multi-hop routed RF messages which send or return information.

The Flashlight consists of an analog compass, alert LED, pager vibrator, a 3 position mode switch, a power switch,

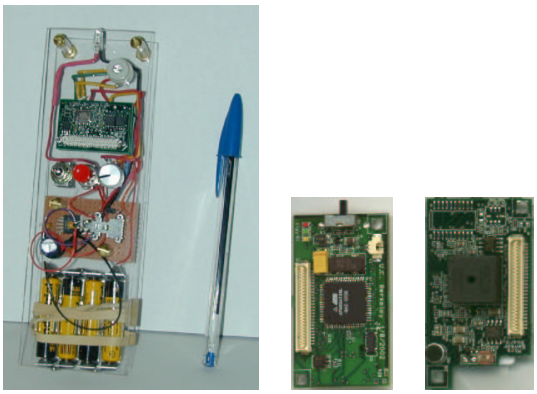

Figure 2: The left figure shows the Flashlight prototype. The center figure shows a Mote board. The right figure shows the Mote sensor board.

a range potentiometer, some power conditioning circuitry, and a microcontroller based CPU/RF transceiver. The processing and RF communication components of the Flashlight and the sensor network are Berkeley Motes [Hill *et al.*, 2001], shown in Figure 2(center,right). A switch selects the sensor type (light, sound, temperature, etc.) When the user points the Flashlight in a direction, if sensor reports of the selected type are received from any sensors in that direction, a silent vibrating alarm activates. The vibration amplitude can be used to encode how far (in number of hops) was the sensor that triggered. The potentiometer is used to set the detection range (calibrated in number of network hops from sensor to sensor.) The electronic compass supplies heading data indicating the pointed direction of the device.

The Flashlight uses one Berkeley Mote (http://today.CS.Berkeley.EDU/tos/) as a main processor and sensor board. The Mote handles data processing tasks, A/D conversion of sensor output, RF transmission and reception, and user interface I/O. It consists of an Atmel ATMega128 microcontroller (with 4 MHz 8 bit CPU, 128KB flash program space, 4K RAM, 4K EEPROM), a 916 MHz RF transceiver (50Kbits/sec, 100ft range), a UART and a 4Mbit serial flash. A Mote runs for approximately one month on two AA batteries. It includes light, sound, and temperature sensors, but other types of sensors may be added. Each Mote runs the TinyOS operating system.

A moving Flashlight interacts with a wireless sensor network consisting of Mote sensors. The sensors are currently programmed to react to sudden increases in light and temperature intensity, but other sensory modes are possible. The Flashlight and all sensors know their location coordinates. These are currently provided to each sensor, but the location parameters can be acquired with GPS or with a calibration procedure.

3.2 The Communication Protocols

The Flashlight has three modes, Activate Sensors, Deactivate Sensors, and Detect, selected by the mode switch. In this section we describe each of these protocols.

Algorithm 1 The Route Update routing algorithm.

```
1:  if NumberOfHops < k* MessageSize then
2:      if RouteIsSet AND MessageIsFromParent then
3:          Set route timeout
4:          Increment message hop count
5:          Add network ID to message hop history
6:          Broadcast updated message
7:      if NOT RouteIsSet then
8:          Record route to Flashlight
9:          Set route timeout
10:         Increment message hop count
11:         Add network ID to message hop history
12:         Broadcast updated message
```

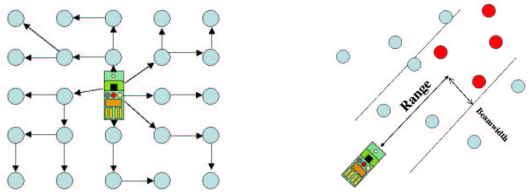

Figure 3: (Left) A snapshot of the sensor network after each sensor has established a multi-hop route to the Flashlight. (Right) The directional activation of sensors. A VR message travels in a specified direction. The sensors contained within a given range (shown in black) have been selected.

Route Updates

The Route Update protocol establishes a *multi-hop* path from each sensor to the Flashlight (see Figure 3(Left)). These paths are then used by other functions of the Flashlight to collect data from the sensors.

In all modes the device sends out Route Update messages every t seconds which are used and forwarded by the sensors to determine a valid *multi-hop route* to the Flashlight. These routes depend on the network configuration and may change over time.

The Route Update (RU) messages are 32 bytes and consist of the Flashlight Network ID, the Network IDs for the last k hops and the CRC checksum.

Each sensor uses this information to store the network ID of a parent through which it can route messages to the Flashlight (see Algorithm 1). The messages are propagated across the entire sensor network in a hop-by-hop fashion. Each sensor prevents loops by allowing only RU messages from its parent node until a timeout limit is reached. After the timeout, the sensor chooses the first rebroadcasting node it hears as its parent. The CRC checksum is used to discard corrupted messages since TinyOS has minimal measures for preventing message collisions.

Activate/Deactivate Region

The Flashlight can turn on all the sensors in a specified geographical area to *activate* the area or it can turn them off to *deactivate* that area. Activate allows the network and its

Algorithm 2 The Vector Route routing algorithm.

```
1: if ThisMessageID ≠ AnyOfLastkMessageIDs then
2:     Update message ID list
3:     Calculate Flashlight direction vector
4:     Calculate perpendicular to Flashlight vector that
       goes through the sensor location
5:     Find point on Flashlight vector that is closest to this
       sensor
6:     Calculate distance of sensor from Flashlight beam
       center
7:     if the point on Flashlight vector is not behind the
       Flashlight then
8:         DistanceFromBeam = distance of sensor from
           Flashlight beam center
9:         DistanceFromFlashlight = distance of sensor from
           Flashlight
10:        if (Range < DistanceFromFlashlight) AND (Dis-
           tanceFromBeam < Beamwidth) then
11:            Activate or Deactivate sensor based on flag
12:            Rebroadcast Vector Route message
```

user to collect higher-resolution information about given locations when all the sensors in the area are on. Activate can be augmented with a probabilistic component that selects only a fraction of the sensors in the area. Deactivate allows the network to switch off all (or a fraction of) the sensors in an area. This functionality is especially useful during a quiet period, when no events trigger and sleeping sensors preserve battery power to extend the network lifetime.

In the Activate mode the Flashlight sends out a Vector Route message regularly, every s seconds. The Vector Route (VR) message is 32 bytes long and consists of the following information: 2 bytes - Flashlight Network ID; 4 bytes - Flashlight Latitude; 4 bytes - Flashlight Longitude; 2 bytes - Range to travel in feet; 2 bytes - Direction to travel; 2 bytes - Beamwidth; 2 bytes - Hop count; 4 bytes - Unique message ID; 1 byte - Activate/Deactivate flag; 1 byte - mode flag; and 2 bytes - CRC checksum.

Algorithm 2 shows the Vector Route protocol that is used to activate an area. At a top level, a message carrying the geometry of an area (specified as a direction dir and distance dis from the device) propagates through the network in dir, selecting all the sensors that are at least dis away. Each sensor uses the message information and its local state to determine whether it is part of the activation area or not. This protocol can be extended easily to accomodate areas of any given geometry.

Detect

Activated sensors watch for events and send messages to the Flashlight using the paths computed with Route Update messages. The Flashlight can store time-stamped sensor triggers in a database for later use. Sensor detects can also be used instantly to present feedback about the direction and distance to the sensor that triggered. Such feedback can be used to provide locomotion direction for a robot or human traversing the sensor field to find the sensor that triggered. More detailed information and history may be stored locally on the sensor for future consumption.

Algorithm 3 The Detect routing algorithm.

```
1: if RouteIsSet AND (RouteTimeOut > 0) AND
   (ThisMessageID ≠ PreviousMessageID) then
2:     PreviousMessageID = ThisMessageID
3:     Add network ID to hop history
4:     Send the data message to parent
```

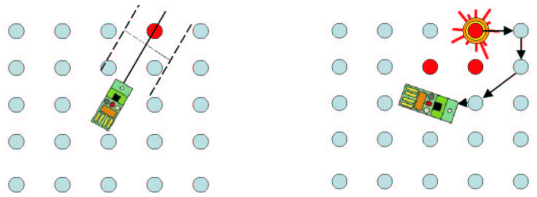

Figure 4: (Left) A snapshot of the sensor network after an area of the network (the red node) has been activated. (Right) A sensor detect event is forwarded to the Flashlight along a mulithop route.

Sensor Detect messages contain the coordinates of the sensor and sensor data. In our implementations, we have used light and temperature intensity data.

Detect messages are generated from a sensor when it detects a change in its current sensory values, based on a threshold. Detect messages are then forwarded from sensor to sensor along the routes previously established by the RU messages (see Figure 4(Right)). Upon reaching the Flashlight, the heading toward the detection and the number of hops to the sensor that triggered are stored in a table. The Flashlight gives feedback to a human user by lighting its LED and turning on a vibration if its current physical heading matches a heading stored in the table. The vibration amplitude depends on the number of hops to the the sensor that triggered.

4 Experiments

We have implemented the Flashlight communication infrastructure described in Section 3 and Section 2 and verified their correctness. We also implemented two guidance applications. The first application computes the safest path across a dangerous region—for example across a forest fire or a contaminated compound. In this application, the sensors are assumed to record danger levels. The safest path across the sensor field is then computed and updated incrementally, as the danger levels change [Li *et al.*, 2003]. In the second application, we use this system to guide a firefighter or robot to a victim trapped in smoke; we then guide the people out. In this application, a sensor triggers (for example by user contact). A path from this sensor to an outside base station is then computed. The Flashlight interacts with the sensors one-by-one, each time giving the next direction of movement to the user.

These applications have been implemented using the Flashlight we built and a 48 node Mote sensor network

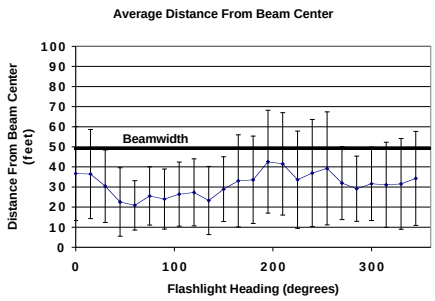

Figure 6: The average beam width for selected sensors for a suite of Vector Route messages. The x axis shows the Flashlight heading. The y axis shows the average distance from the beam center of the selected sensors for the given Flashlight heading.

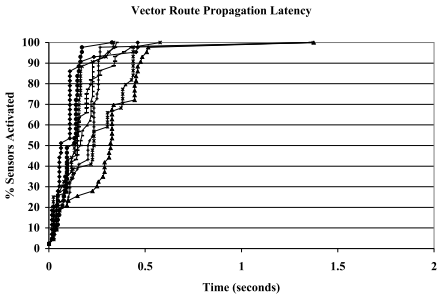

Figure 7: Vector Route message latency. The x axis is time. The y axis shows sensor percentage. The curves correspond to different experiments.

running our communication protocols. These experiments were also used to collect data about the effectiveness of our communication protocols.

4.1 Experimental Setup

We have placed 48 Mote sensors in a grid pattern and in a U-shaped pattern for all our evaluation experiments (see Figure 5). The sensors were provided with location coordinates corresponding to a horizontal separation of 32.4 feet and a vertical separation of 35.4 feet. The Flashlight was placed at the location of one of the grid sensors and given its coordinates[1]. A software adjustment allows us to scale down the area covered by the sensor network to carry out experiments on a desk top. We adjusted the transmission power of the Mote sensors to a half a foot using the digital potentiometer.

It is generally difficult to collect data from a distributed set of sensors connected only by an unreliable low bandwidth radio link. As a result, 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 resoltuion 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.

4.2 Beam accuracy experiments

In the first experiment we measured the accuracy of the beam estimation in the Vector Route algorithm (see Algorithm 2). We ran the Vector Route algorithm and observed the sensors that were identified to be within the beam width (these sensors' LED lit up.) We placed the Flashlight at one corner of the sensor grid and repeated the Vector Route algorithm for several orientations. We then computed the distance of each activated sensor from the actual beam center path. The distances for each angular increment were averaged and the results are shown in Figure 6.

Since the beam width parameter was chosen at 50 feet, we expected the average distance to the beam to be less than 50 feet. In general, the algorithm did quite well, with the sensors that were activated being mostly within the beam. We observed some error in the experiments conducted around a Flashlight heading around true North and West. We believe these errors are due to the non-homogeneous magnetic environment where the experiment was conducted. This introduced nonlinearities in the compass readings within the Flashlight. Such nonlinearities are likely to show up in actual use as well, hence a magnetic compass is not the best device to use for getting bearing.

4.3 Message propagation latency experiments

In a different set of experiments, we measured the message propagation latency for Vector Route messages. To maximize the number of hops in the network we used the U-shaped testbed. All the sensors were given the same location, so that they would all be within the beamwidth of a VR message simultaneously and propagation time from the start to the end of the chain could be measured. Note that messages often jump over more than one sensor along the chain due to the variability of RF transmission/reception range. Thus, although we had 48 sensors, the max hop range for the network was less.

The results for the VR message propagation times are shown in Figure 7 for sensor activation times. The propagation times are the sum of the delays caused by message transmission time, computational load[2] causing delays in forwarding, collision delays causing message loss, and hop range variability which affects the number of sensors reached by each message.

4.4 Directional Guidance

We have deployed 12 Mote sensors along corridors in our building and used the Flashlight and the communication infrastructure presented here to guide a human user out of the building. The Flashlight interacted with sensors to compute the next direction of movement towards the exit. For each interaction, the user did a rotation scan until the Flashlight was pointed in the direction computed from the sensor data. The user then walked in that direction to the next sensor. Each time we recorded the correct direction and the

[1] By placing the Flashlight at grid corners we accomplish "virtual grids" of 200 sensors.

[2] Trigonometry is expensive for a 4MHz 8 bit microcontroller with no floating point co-processor and software implemented multiplies and divides.


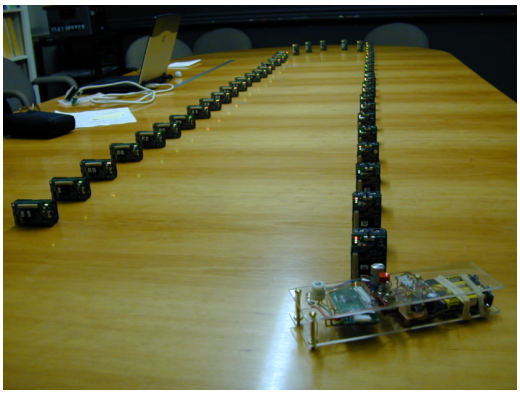

Figure 5: (Left) The Mote grid testbed. (Right) The Mote U-shape testbed.

direction detected by the Flashlight. The directional error was 8% (or 30 degrees) on average, and it was due to the non-homogeneous magnetic field in the building, especially around metal stairs. However, because the corridors and office doorways are wide, and the sensors sufficiently dense, the exit was identified successfully. The user was never directed towards a blocked or wrong configuration. We envision an application where the sensors will collect temperature gradients and the guidance algorithm will compute the safest path to the exit.

5 Conclusions

We have discussed sensor networks that can interact with robot or human users. We have described the Flashlight, an interaction device for sending commands to the sensor network and discussed several protocols in the context of navigation guidance applications: activating a given area of the sensor network, deactivating a given area of the sensor network, detecting events in the sensor network and using these events for navigation. We have implemented these protocols on a network of 48 Mote sensors and presented some experimental data collected from this testbed.

This work has given us several insights into using ad-hoc networks for robot interactions with sensor networks. Data loss is not rare in sensor networks. This is due to network congestion, transmission interference, and garbled messages. The transmission range of one direction may be quite different from that of the opposite direction. Thus, the assumption that if a node receives a packet from another node, it can send back a packet does not hold. 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. The uncertainty introduced by data loss, asymmetry, congestion, and transient links is fundamental in sensor networks and should be carefully considered in developing models and algorithms for systems that involve sensor networks.

Acknowledgment

Support for this work was provided through the NSF awards IRI-9714332, EIA-9901589, IIS-9818299, and IIS-9912193, ONR award N00014-01-1-0675, an AFOSR MURI award, Darpa, and the Sloan foundation. We are grateful for it.

References

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

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

[Hill *et al.*, 2000] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *ASPLOS*, 2000.

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

[Lengyel *et al.*, 1990] 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.

[Li *et al.*, 2003] Q. Li, M. de Rosa, and D. Rus. Distributed algorithms for guiding navigation across a sensor net. In *submitted to MobiHoc 2003*, 2003.

[Ramanathan and Hain, 2000] Ram Ramanathan and Regina Hain. Topology control of multihop wireless networks using transmit power adjustment. In *INFOCOM*, 2000.

[Wattenhofer *et al.*,] R. Wattenhofer, L. Li, P. Bahl, and Y. Wang. Distrib. topology control for power efficient operation in multihop wireless ad hoc networks. In *INFOCOM 2001*.

[Xu *et al.*, 2001] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *MOBICOM 2001*, July 2001.