Wireless Sensor Networks for Emergency Navigation

Yu-Chee Tseng, Meng-Shiuan Pan, and Yuen-Yung Tsai
National Chiao Tung University

In an emergency, wireless network sensors combined with a navigation algorithm could help safely guide people to a building exit while helping them avoid hazardous areas.

The rapid progress of wireless communications and embedded microelectromechanical systems technologies has made wireless sensor networks possible. WSNs typically need to configure themselves automatically and support ad hoc routing. Recent research efforts have focused on various WSN features, including power management, routing and transportation, self-organization, deployment, coverage, and localization.

WSN design is generally application driven—that is, a particular application's requirements will determine how the network behaves. Current WSN projects include the Great Duck Island's habitat-monitoring application (www.greatduckisland.net/index.php); the FireBug project (http://firebug.sourceforge.net), which uses wireless sensors with the Global Positioning System to monitor wildfires; and mobile object tracking.[1]

As the "Related Work in Emergency Navigation" sidebar describes, previous navigation applications weren't designed with emergency applications in mind. In emergency applications, such as fire rescues, a safe escape path is more critical than factors such as energy consumption.

We propose a distributed navigation algorithm for emergency situations. At normal time, sensors monitor the environment. When the sensors detect emergency events, our protocol quickly separates hazardous areas from safe areas, and the sensors establish escape paths. Simulation and implementation results show that our scheme achieves navigation safety and quick convergence of the navigation directions.

PROPOSED NAVIGATION PROTOCOL

We based our protocol on the *temporally ordered routing algorithm*[2] for mobile ad hoc networks. TORA assigns mobile nodes temporally ordered sequence numbers to support multipath routing from a source to a specific destination node. The algorithm expresses the sequence number as a quintuple.

To handle mobility, TORA adopts a link-reversal procedure when hosts lose their outgoing paths. TORA's multipath routing concept fits well with the requirements of emergency navigation services.

However, we can't directly apply TORA to our environment for several reasons:

- Expressing a node's weight as a quintuple can be too costly for sensors with weak communication capability (for example, typical mote packets are only 29 bytes).
- TORA looks for shorter and multipath routes, whereas our navigation service looks for safer, but not necessarily shorter, escape paths.
- The considerations for users in hazardous and nonhazardous regions differ. Our navigation service is essentially an all-to-many routing (from all locations to one or multiple exits), and emergency locations disturb the discovery of safe paths.

We consider a sensor network deployed in a 2D indoor environment. Each sensor knows its own location in the 2D plane. By overhearing wireless signals, a

0018-9162/06/$20.00 © 2006 IEEE

Related Work in Emergency Navigation

In previous work using an algorithm to guide a robot to a goal,[1] sensor nodes act as signposts for the robot to follow. Each sensor determines the direction in which the robot should move by computing a probability value for each of its neighbors. A neighbor with a higher probability is closer to the goal.

Qun Li, Michael DeRosa, and Daniela Rus assumed that multiple emergency points (or obstacles) and one exit exist in the environment.[2] Their goal was to find a navigation path from each sensor to the exit without passing any obstacles. They designed their algorithms using the *artificial potential fields* concept. The exit generates an attractive potential, pulling sensors to the exit, while each obstacle generates a repulsive potential, pushing sensors away from it. Each sensor calculates its potential value and tries to find a navigation path with the least total potential value. Other researchers have applied this concept to other application scenarios, such as robot navigation and distributed search and rescue.[3]

Although Li and his colleagues' algorithm can find a shorter and safer path from each sensor to the exit, it has several drawbacks.

First, it can incur high message overheads. Because the algorithm constructs navigation directions from the exit to other sensors, a minor change of potential in a sensor near the exit can cause many other sensors to recompute their potentials, causing many message exchanges and even delays in making the navigation decision.

Second, the algorithm has no concept of hazardous regions. With shortest-path routing, the algorithm could determine a path that is very close to the emergency location. Consider Figure A1, in which there are two exits, A and B. When a sensor detects an emergency, the algorithm could direct some users to B, making them pass through the hazardous region. Guiding users as in Figure A2 is more desirable because the algorithm only directs users inside the hazardous region to exit B.

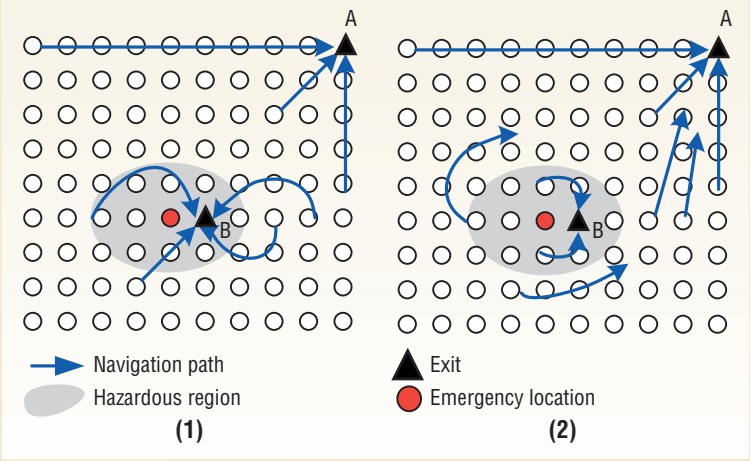


Figure A. Navigation scenarios in which the hazardous region is defined as two hops from the emergency site. (1) Users are directed to an exit in a hazardous region, even though another exit exists. (2) Only users already in the hazardous region are directed to exit B.

References

1. M.A. Batalin, G.S. Sukhatme, and M. Hattig, "Mobile Robot Navigation Using a Sensor Network," *Proc. IEEE Int'l Conf. Robotics and Automation*, IEEE Press, 2004, pp. 636-642.
2. P. Corke, R. Peterson, and D. Rus, "Networked Robots: Flying Robot Navigation Using a Sensor Net," *Proc. Int'l Symp. Robotics Research* (ISRR), *Springer Tracts on Advanced Robotics* (STAR). Springer-Verlag, *2003*.
3. Q. Li, M. DeRosa, and D. Rus, "Distributed Algorithm for Guiding Navigation across a Sensor Network," *Proc. ACM Int'l Symp. Mobile Ad Hoc Networking and Computing* (MobiHOC), ACM Press, 2003, pp. 313-325.

sensor can form wireless links with its neighbors. Depending on how the plane is partitioned, a sensor could form navigation links with its neighbors. (Because of radio's penetration capacity, a wireless link doesn't necessarily imply a navigation link.)

We therefore manually define navigation links during deployment. From these links, we construct a navigation graph. In the following discussion, all operations refer to the navigation graph, although the underlying message exchange is through wireless links.

Our design emphasizes correctness in discovering escape paths even if users must pass hazardous areas. We divide the algorithm into two phases: *initialization* and *navigation*.

Initialization phase

In the initialization phase, we assign each sensor an altitude according to its hop distance to the nearest exit. We assign sensors near the exits smaller altitudes, and sensors farther from the exits higher altitudes. The escape paths are along sensors with higher altitudes to those with lower attitudes.

To compute sensors' initial altitude, exit sensors broadcast *initial* packets to start the initialization phase. An initial packet consists of three fields: sender ID, exit sensor ID, and hop count. In an exit sensor broadcast, the hop count field is set to zero.

When a sensor receives an initial packet, it increments the hop count by one and accepts this value as its initial alti-

tude unless it has a smaller altitude. It then rebroadcasts the initial packet with the updated hop count. The initialization phase is complete when each sensor has an altitude.

Exit sensors periodically restart the initialization phase to adjust for possible topology changes. In this process, each sensor keeps a neighbor table, in which each entry is of the format `<neighbor ID, is_exit, altitude>` to track its neighbors' status.

Navigation phase

The navigation phase begins when the system detects an emergency event. The goal is for each sensor to choose the neighbor with the smallest altitude as its escape direction.

We use the following notations:

- D is a constant such that any sensor whose distance to any emergency location is less than or equal to D is considered within a hazardous region. We use hop count to calculate the distance.
- A_{emg} is a large constant that the algorithm assigns to a sensor that detects an emergency event.
- A_i is sensor i's altitude.
- I_i is the altitude of sensor i obtained in the initialization phase.
- $e_{i,j}$ is the hop count from an emergency sensor i to a sensor j.
- An EMG packet is the emergency notification packet. It has five fields: event sequence number, ID of the sensor finding the emergency event, sender's ID, sender's altitude, and hop count from the sender to the emergency sensor.

Our design doesn't adopt the full-reversal approach because it could easily guide users to pass through a hazardous region unnecessarily.

Assume a sensor x detects an emergency. It sets its altitude to A_{emg} and immediately broadcasts an EMG(seq, x, x, A_{emg}, 0) packet, which is flooded in the network. The following steps summarize the system's actions when a sensor y receives from a sensor w an EMG(seq, x, w, A_w, h) packet originating from x.

Step 1. Sensor y determines whether the packet describes a new emergency by checking the tuple (seq, x). If it is, y records the event and sets $e_{x,y}$ to $h + 1$. Otherwise, y checks if $h + 1 < e_{x,y}$. If so, y changes $e_{x,y}$ to $h + 1$.

Next, y records w's altitude (A_w) in its neighbor table. If $w = x$ and x is an exit sensor, y clears the flag *is_exit* in the table entry for x to avoid guiding users into this emergency location.

Step 2. If y changed $e_{x,y}$ in step 1 and $e_{x,y} \leq D$, y considers itself to be within the hazardous region formed by sensor x and recalculates its altitude as:

$$A_y = \max\left\{A_y, A_{emg} \times \frac{1}{e_{x,y}^2} + I_y\right\}$$

The algorithm increases the altitude of a sensor inside a hazardous region by an amount inversely proportionate to the square of its distance to the emergency location. We include the value I_y to reflect y's distance to its nearest exit. The maximum function accounts for the possibility that y is located within multiple hazardous regions and thus might receive EMG packets from several sources. In this case, y's new altitude should reflect its distance to the nearest emergency location.

Step 3. Unless it's an exit sensor, y then determines whether it has a local minimum altitude. If y is a local minimum (that is, its altitude is less than that of its neighbors), it adjusts its altitude as:

$$A_y = STA\left(A_{N_y}\right) \times \frac{1}{|N_y|} + \min\left\{A_{N_y}\right\} + \delta,$$

where N_y is the set of all of y's neighbors, $STA()$ is the standard deviation of the sensor altitudes in N_y, and δ is a small constant.

Using a standard deviation lets the system respond to emergency situations quickly. When N_y's altitudes vary significantly, y is likely near a hazardous region and should increase its altitude quickly to avoid becoming a local minimum again.

A fixed constant δ guarantees convergence. Developers should carefully choose its value because a large δ could easily guide sensors to cross hazardous regions. On the other hand, although it could help sensors find safer paths, a small δ can cost too many message exchanges.

The reciprocal of $|N_y|$ reflects the number of possible choices that a sensor has for selecting escape directions. A sensor with fewer neighbors will increase its altitude more quickly to avoid becoming a local minimum.

These elements will speed up the algorithm's convergence time. Each sensor must keep returning to this step to check whether it has become a local minimum.

Step 4. Finally, y broadcasts an EMG(seq, x, y, A_y, $e_{x,y}$) packet if either of the following conditions is true:

- The emergency packet is new.
- The sensor changed A_y or $e_{x,y}$ in the previous steps.

Step 3 uses the partial-reversal concept to adjust local minimum nodes' altitudes. Our design doesn't adopt the full-reversal approach because it could easily guide users to pass through a hazardous region unnecessarily. Using

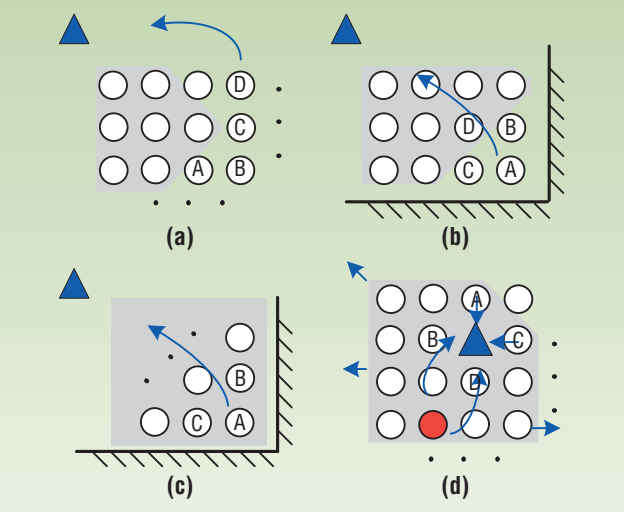


Figure 1. Navigation examples. (a) If an exit path around a hazardous region exists, sensors guide the user toward the exit sensor. (b) When sensors are surrounded by a hazardous region, they raise their altitudes to a level higher than the altitude of at least one sensor in the hazardous region. (c) Sensors that are inside a hazardous region follow similar escape paths. (d) If an exit sensor is in the hazardous region, the algorithm guides users also in the hazardous region to that sensor.

partial reversal can help guide users around a hazardous region.

If users are inside a hazardous region when an emergency occurs, a nearby exit sensor can guide the users either to this exit or to exits in nonhazardous regions.

Our algorithm uses a hybrid approach. Sensors inside hazardous regions can choose an exit sensor that is also in a hazardous region if the exit sensor is within one hop from them. However, sensors in nonhazardous regions will never choose an exit inside a hazardous region unless the sensor is surrounded by hazardous regions or the safe areas offer no proper exits. In this case, sensors continue increasing their altitudes until they reach a level higher than the sensors in hazardous regions.

So, the escape rules for any sensor are as follows:

- If y is in a hazardous region and it detects an exit sensor in N_y that is also in a hazardous region, y chooses this exit sensor.
- In all other cases, y directs users to its neighboring sensor with the lowest altitude.

As long as at least one exit sensor isn't located in an emergency location, our protocol can find an escape path for each nonexit sensor in a finite number of steps. Disregarding exit sensors, only sensors that are local minimums have no escape paths. Because δ is a nonzero constant, our protocol is convergent because it has a progress property in the sense that the number of sensors with no escape paths will decrease.

The value of A_{emg} will affect the navigation results. If the value is too small, altitudes at the boundaries of hazardous regions can be smaller than some sensors' initial altitudes. To avoid this problem, assuming that the sensor's maximum altitude in the initialization phase is MAX_{ini}, A_{emg}'s value should be at least larger than $MAX_{ini} \times (D + 1)^2$.

Navigation examples

Figure 1a shows a hazardous region that isn't closed (that is, an exit path exists around the region). In this scenario, sensors A, B, and C can temporarily become a local minimum. Suppose sensor D has already found an escape path. Sensors A, B, and C will eventually find their escape paths via D.

In Figure 1b, sensors A, B, and C are surrounded by a hazardous region. In this scenario, the three sensors should raise their altitudes to a level higher than the altitude of at least one sensor in the hazardous region. Assume sensor D has the smallest altitude in the hazardous region. With a proper δ, our algorithm will likely guide users to an exit via D.

Figure 1c is similar to the case in Figure 1b except that sensors A, B, and C are all inside the hazardous region, and thus have similar escape paths. In Figure 1d, an exit sensor is in the hazardous region. Sensors A, B, C, and D, which are the exit sensor's direct neighbors, will guide users to that exit. Sensors that aren't the exit sensor's direct neighbors will guide users out of the hazardous region via the shortest paths first, and then to other exits outside of the hazardous region unless no such exits exist.

Figure 2 shows how altitude changes occur in a 7 × 7 grid network with D = 2. Three emergency events occur in coordinates (S2, 4), (S6, 7), and (S5, 2), in that order. An exit is located in (S1, 7). Both the side and top views show altitude changes, in decibels (dB). Navigation paths are from sensors with higher altitudes to sensors with lower altitudes.

SIMULATION RESULTS

We first consider a 10 × 10-grid network. Each sensor has four navigation links to neighboring sensors on the east, west, north, and south. We set A_{emg} to 200 and δ to 0.1.

We compare our algorithm with the algorithm presented by Qun Li, Michael DeRosa, and Daniela Rus, using packet count and convergence time as performance metrics. As the "Related Work in Emergency Navigation" sidebar describes, the packet count doesn't include packets used during the initialization phase.

We simulate an unslotted Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) protocol following the IEEE 802.15.4 standard[3] with a data rate of 20 Kbps. We measure convergence time in microseconds (μs).

Figure 3 shows the simulation results. In case 1, the sensor located in the middle of the network detects an

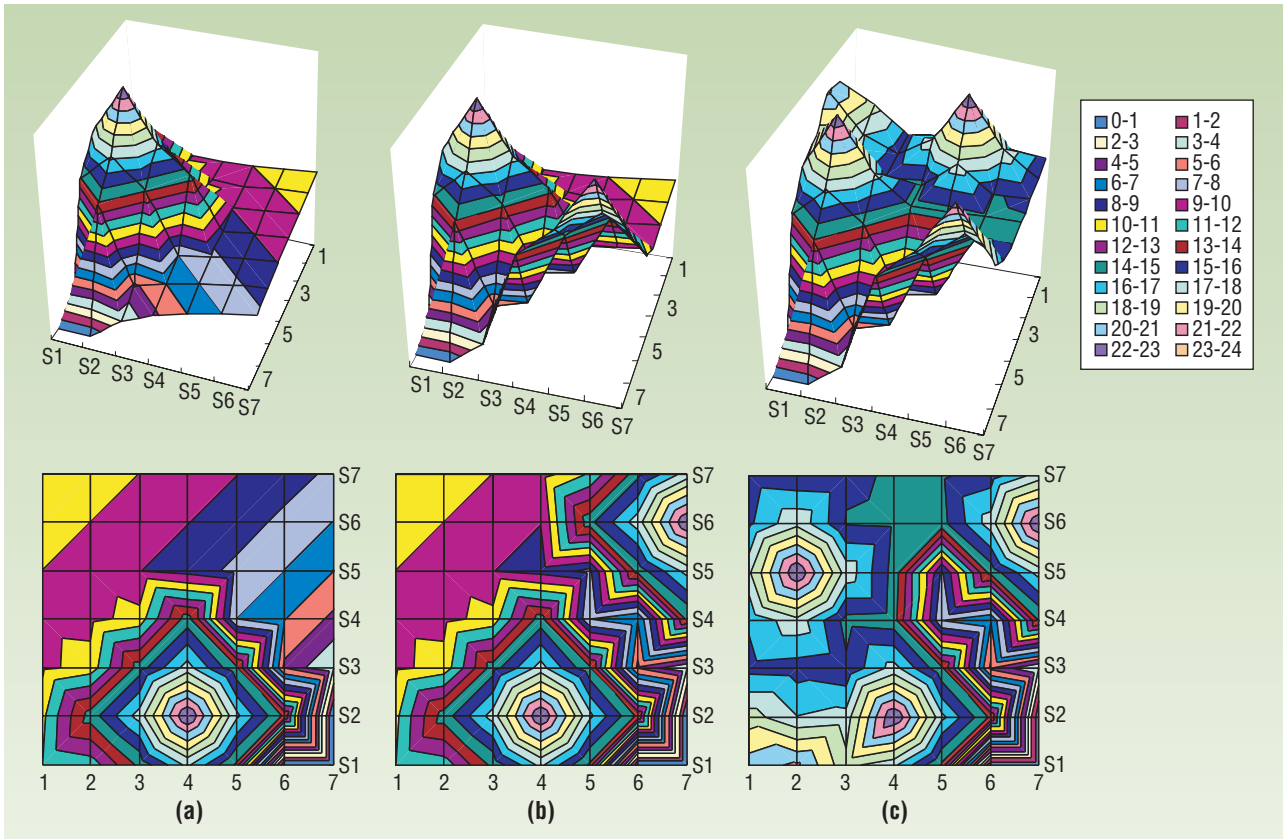


Figure 2. Examples of altitude changes when emergency events occur in (a) coordinates (S2, 4), (b) coordinates (S6, 7), and (c) coordinates (S5, 2).

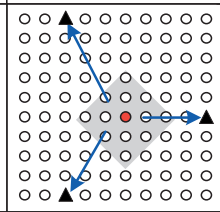
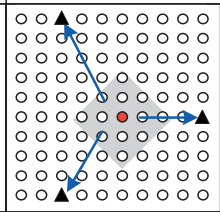
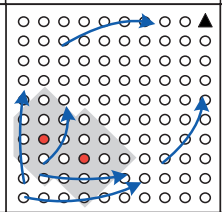
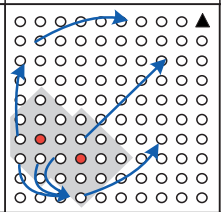
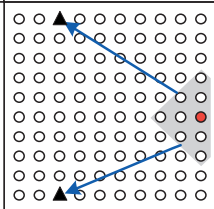
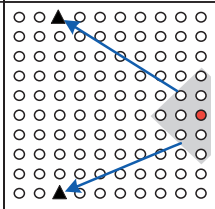
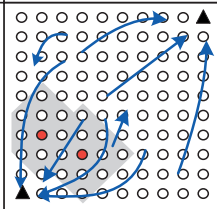
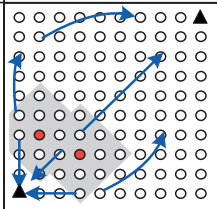
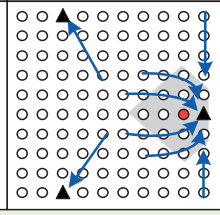
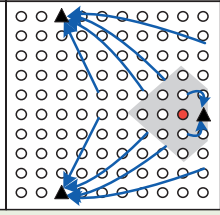
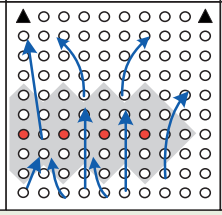
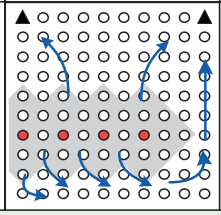
Number	Qun Li, Michael DeRosa, and Daniela Rus		Our method (D = 2)		Number	Qun Li, Michael DeRosa, and Daniela Rus		Our method (D = 2)	
	Path	Packet count/ convergence time	Path	Packet count/ convergence time		Path	Packet count/ convergence time	Path	Packet count/ convergence time
1		660/ 530.81		100/ 59.95	4		979/ 468.83		252/ 78.99
2		1215/ 350.62		130/ 87.5	5		731/ 372.0		264/ 107.15
3		742/ 442.52		137/ 87.89	6		1254/ 350.1		408/ 67.29

Figure 3. Comparison of packet count and convergence time (in μs) in a 10 × 10 grid network.

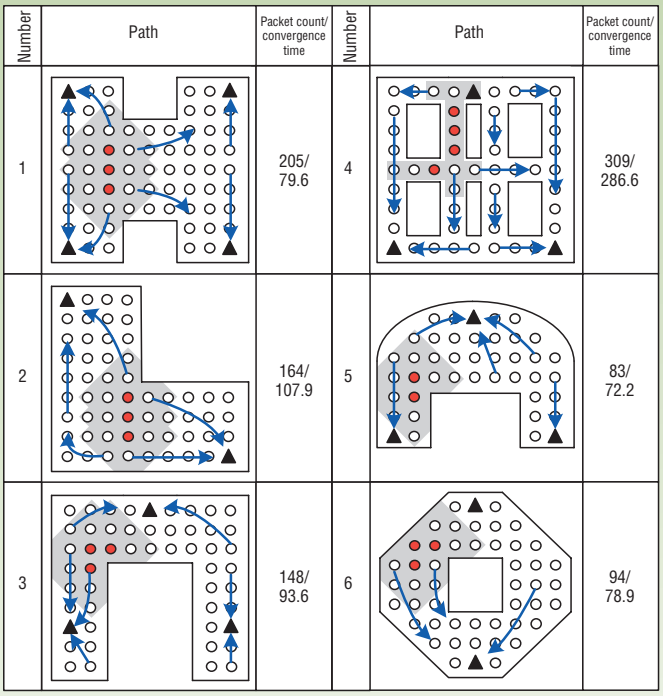


Figure 4. Navigation results in various forms of networks.

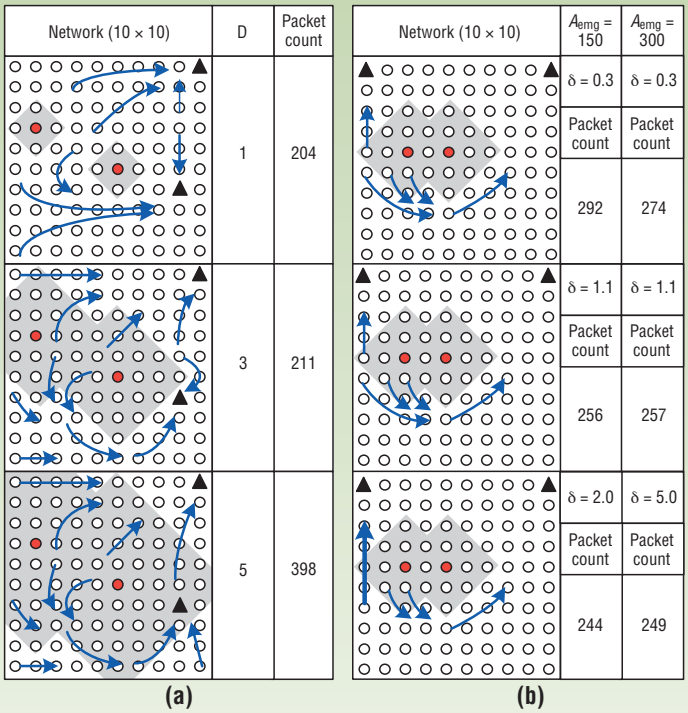


Figure 5. Effect of parameters on system performance. (a) Effect of D on the quality of escaping paths and message overheads. (b) Effects of δ and A_{emg} on the quality of escape paths and message overheads.

emergency event. However, the emergency event doesn't change neighboring sensors' relative altitudes.

Our algorithm sends few messages and quickly converges.

In case 2, sensor placement is the same as in case 1, except that exit sensor A detects an emergency. Although both algorithms will compute the same navigation paths, Li, DeRosa, and Rus's algorithm incurs more messages, because sensors near A will first be attracted to, and then repelled from, A.

In case 3, a sensor detects an emergency event near exit A. Without the hazardous regions concept, the system will guide some users through the hazardous region. Because this is undesirable, our algorithm effectively avoids guiding users through the hazardous region.

In case 4, some sensors are bounded by hazardous regions. Although guiding users through hazardous regions is inevitable, our scheme will choose paths that are as far away from emergency locations as possible.

In case 5, we add an exit in the lower left corner. Li, DeRosa, and Rus's algorithm directs some sensors to pass through the hazardous regions to reach that exit, but our algorithm avoids this problem.

In the scenario in case 6, emergencies almost partition the network. Again, our algorithm discovers safer navigation paths than Li, DeRosa, and Rus's algorithm.

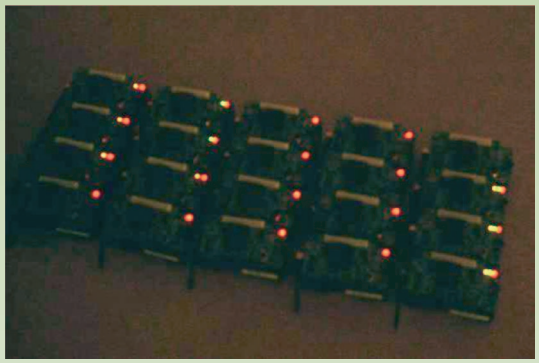
Figure 4 illustrates our navigation results in various network configurations. The results show that our scheme effectively leads people to exits and helps them avoid hazardous regions. They also imply that our protocol is applicable to various building types.

We've also simulated our algorithm in a large-scale sensor network with 2,500 sensor nodes. The algorithm selects between 1 and 5 percent of random sensors as exits, and it selects 1 percent of random sensors as emergency points. We set the parameter D to 5. The convergence times of 1 to 5 percent of the exit sensors are 21.6 seconds, 10 seconds, 9.1 seconds, 2.9 seconds, and 2.0 seconds, respectively. These results demonstrate our algorithm's scalability in large networks.

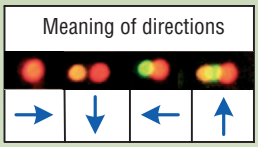
In addition to reflecting the dangerous range affected by an emergency event, D's value can also affect the navigation results and system performance.

For a small network, a D that is too large is meaningless because a few emergency events

(a)



(b)



(c)

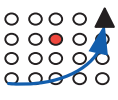
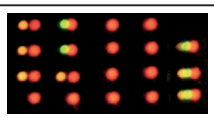
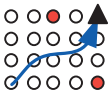
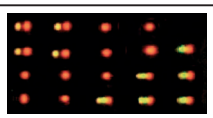
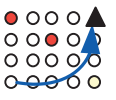
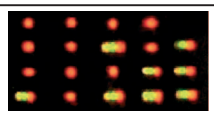
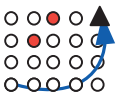
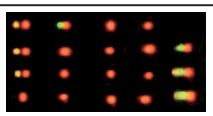
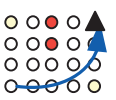
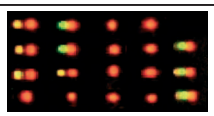
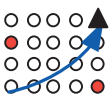
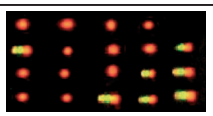
Number	Simulation	Experiment	Packet count/ simulation experiment	Number	Simulation	Experiment	Packet count/ simulation experiment
1			25/26	4			39/47
2			36/40	5			38/44
3			40/44	6			36/40

Figure 6. Simulation and experimental results. (a) A mote's light-emitting diodes during an experiment; (b) LEDs indicate navigation directions; (c) comparison of results.

can result in a network that's entirely covered by hazardous regions. Figure 5a shows different settings of D in a 10 × 10 grid network. A small D can result in users being guided along paths close to emergency sources. Although a large D can help users find safer paths, it also increases the message overhead.

Figure 5b shows the effects of A_{emg} and δ and on escape path quality and message overhead. Using a small A_{emg} with a large δ might make it easier to guide users across hazardous regions, as in the third case in Figure 5b. A larger δ can quickly increase the altitudes of sensors with local minimum to values larger than sensors in hazardous regions. We thus recommend using a relatively large A_{emg} with a relatively small δ. In our current design, we configured parameters D, A_{emg}, and δ at the deployment stage, and we can use manual simulations to determine them.

PROTOTYPE EXPERIENCES

Our prototype system uses MICAz Mote radio platforms (www.xbow.com/Products/productsdetails.aspx?sid=3) and incorporates photo sensors, which read light degrees, to simulate and trigger emergency events. In the system, a light degree above a threshold indicates an emergency event, and a sensor reading this light degree will broadcast EMG packets. Because broadcast communications are unreliable,[4] packets can be lost. Sensors therefore periodically rebroadcast EMG packets to improve reliability.

Figure 6a shows the testing of a 4 × 5 grid network. The motes' light-emitting diodes (LEDs) indicate navigation directions, as Figure 6b shows.

Figure 6c shows results from our simulation and experimental testing of the network, including message costs and navigation paths. Because of packet loss, the experimental results show a slightly higher message overhead. Because the network is small, the navigation paths in both the simulations and the experiments are exactly the same.

As part of our future research, we plan to improve the user interfaces, such as the LED display. We will also extend the results to 3D environments.

Acknowledgments

The following organizations cosponsor Yu-Chee Tseng's research: the National Science Council under grant number 93-2752-E-007-001-PAE; the Institute for Information Industry under the Communications Software Technology

Project; the Ministry of Economic Affairs ROC under grant number 94-EC-17-A-04-S1-044; the Industrial Technology Research Institute, Taiwan; and Intel.

References

1. C-Y. Lin and Y-C. Tseng, "Structures for In-Network Moving Object Tracking in Wireless Sensor Networks," *Proc. Broadband Wireless Networking Symp.* (BroadNet), IEEE CS Press, 2004, pp. 718-727.
2. V.D. Park and M.S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," *Proc. IEEE INFOCOM*, IEEE Press, 1997, pp. 1405-1413.
3. *IEEE Std. 802.15.4, IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-rate Wireless Personal Area Networks (LRWPANs)*, IEEE, 2003.
4. Y. C. Tseng, S-Y. Ni, and E- Y. Shih, "Adaptive Approaches to Relieving Broadcast Storms in a Wireless Multihop Mobile Ad Hoc Network," *IEEE Trans. Computers*, vol. 52, no. 5, 2003, pp. 545-557.

Yu-Chee Tseng *is the chair and a professor in the Department of Computer Science at National Chiao Tung University. His research interests include mobile computing, wireless communication, and parallel and distributed computing. He received a PhD in computer and information science from Ohio State University. Tseng is a member of the ACM and a senior member of the IEEE. Contact him at yctseng@cs.nctu.edu.tw.*

Meng-Shiuan Pan *is a PhD student at National Chiao Tung University. His research interests are wireless sensor networks and mobile computing. He received an MS in communications engineering from National Tsing Hua University. Contact him at mspan@csie.nctu.edu.tw.*

Yuen-Yung Tsai *is an MS student at National Chiao Tung University. His research interests are wireless sensor networks and mobile computing. He received a BS in computer science from National Chiao Tung University. Contact him at uytsai@csie.nctu.edu.tw.*