iMouse: An Integrated Mobile Surveillance and Wireless Sensor System

Yu-Chee Tseng, You-Chiun Wang, Kai-Yang Cheng, and Yao-Yu Hsieh
National Chiao Tung University

Incorporating the environment-sensing capability of wireless sensor networks into video-based surveillance systems can provide advanced services at a lower cost than traditional surveillance systems. The integrated mobile surveillance and wireless sensor system (iMouse) uses static and mobile wireless sensors to detect and then analyze unusual events in the environment.

The remarkable advances of microsensing microelectromechanical systems (MEMS) and wireless communication technologies have promoted the development of wireless sensor networks. A WSN consists of many sensor nodes densely deployed in a field, each able to collect environmental information and together able to support multihop ad hoc routing. WSNs provide an inexpensive and convenient way to monitor physical environments. With their environment-sensing capability, WSNs can enrich human life in applications such as healthcare, building monitoring, and home security.

Traditional surveillance systems typically collect a large volume of videos from wallboard cameras, which require huge computation or manpower to analyze. Integrating WSNs' sensing capability into these systems can reduce such overhead while providing more advanced, context-rich services. For example, in a security application, when the system detects an intruder, it can conduct in-depth analyses to identify the possible source. The "Related Work in Wireless Surveillance" sidebar provides additional information about other work in this area.

Our *integrated mobile surveillance and wireless sensor system* (iMouse) consists of numerous static wireless sensors and several more powerful mobile sensors. The benefits of iMouse include the following:

- It provides online real-time monitoring. For example, when the system is capturing events, the static sensors can immediately inform users where the events are occurring, and the mobile sensors can later provide detailed images of these events.
- It's event-driven, in the sense that only when an event occurs is a mobile sensor dispatched to capture images of that event. Thus, iMouse can avoid recording unnecessary images when nothing happens.
- The more expensive mobile sensors are dispatched to the event locations. They don't need to cover the whole sensing field, so only a small number of them are required.
- It's both modular and scalable. Adding more sophisticated devices to the mobile sensors can strengthen their sensing capability without substituting existing static sensors.

Because mobile sensors run on batteries, extending their lifetime is an important issue. We thus propose a dispatch problem that addresses how to schedule mobile sensors to visit emergency sites to conserve their energy as much as possible. We show that if the number of emergency sites is no larger than the number of mobile sensors, we can transform the problem to a maximum matching problem in a bipartite graph; otherwise, we group emergency sites into clusters

Published by the IEEE Computer Society

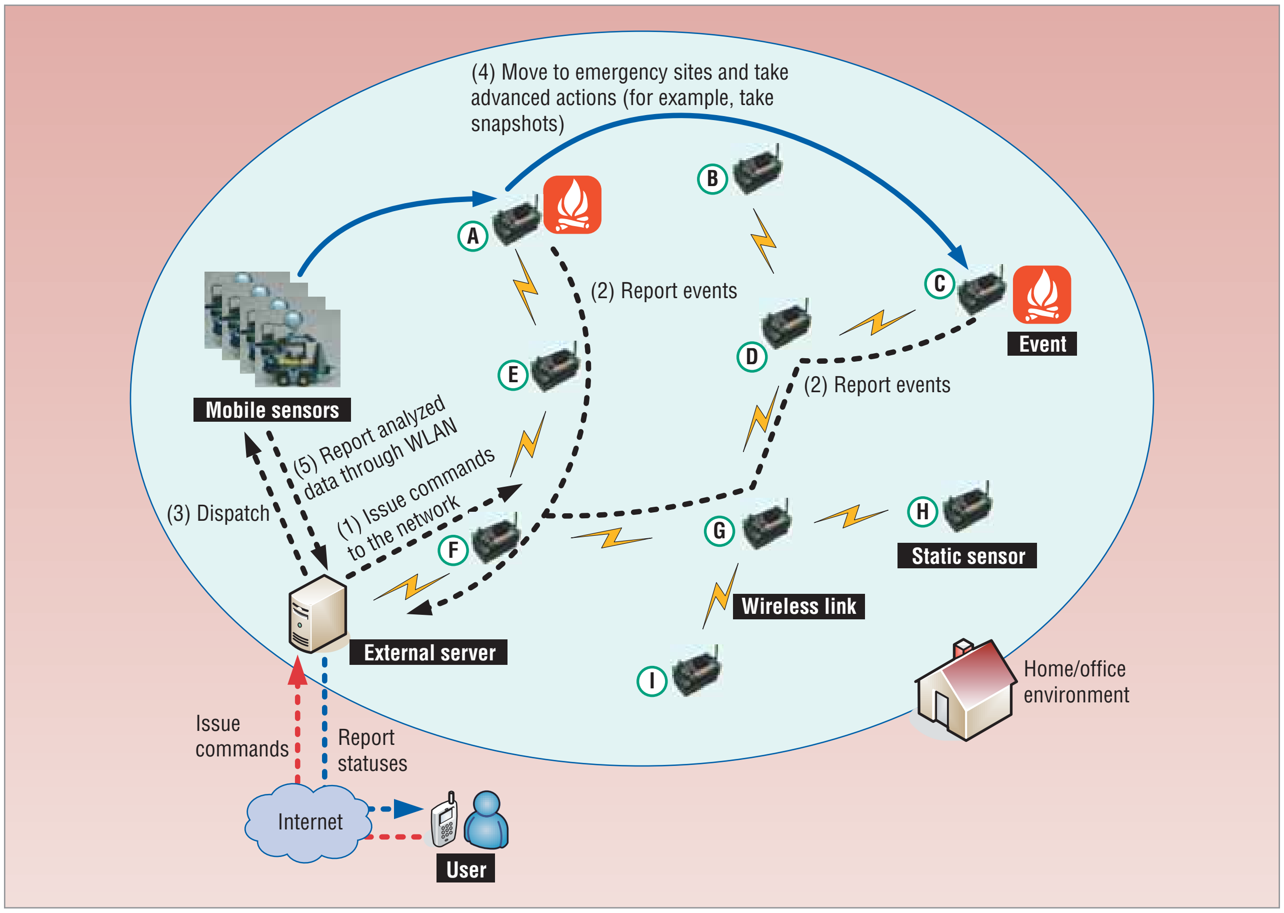


Figure 1. The iMouse system architecture. Three main components make up the iMouse architecture: static sensors, mobile sensors, and an external server. The user issues commands to the network through the server (1). Static sensors monitor the environment and report events (2). When notified of an unusual event, the server notifies the user and dispatches mobile sensors (3), which move to the emergency sites, collect data, (4) and report back to the server (5).

so that one mobile sensor can efficiently visit each cluster.

SYSTEM DESIGN

Figure 1 shows the iMouse architecture, which consists of static and mobile sensors and an external server. The static sensors form a WSN to monitor the environment and notify the server of unusual events. Each static sensor comprises a sensing board and a mote for communication. In our current prototype, the sensing board can collect three types of data: light, sound, and temperature. We assume that the sensors are in known locations, which users can establish through manual setting, GPS, or any localization schemes.[1]

An event occurs when the sensory input is higher or lower than a predefined threshold. Sensors can combine inputs to define a new event. For example, a sensor can interpret a combination of light and temperature readings as a potential fire emergency. To detect an explosion, a sensor can use a combination of temperature and sound readings. Or, for home security, it can use an unusual sound or light reading. To conserve static sensors' energy, event reporting is reactive.

Mobile sensors can move to event locations, exchange messages with other sensors, take snapshots of event scenes, and transmit images to the server. As Figure 2 shows, each mobile sensor is equipped with a Stargate processing board (www.xbow.com/Products/productsdetails.aspx?sid=229), which is connected to the following:

- a Lego car (http://mindstorms.lego.com/eng/default.asp), to support mobility;
- a mote, to communicate with the static sensors;
- a webcam, to take snapshots; and
- an IEEE 802.11 WLAN card, to support high-speed, long-distance communications, such as transmitting images.

The Stargate controls the movement of the Lego car and the webcam.

The external server provides an interface through which users can obtain the system status and issue commands. It also maintains the network and interprets the meanings of events from sensors. On detecting a potential emergency, the server dispatches mobile sensors to visit emergency sites to obtain high-resolution images of the scene. The dispatch algorithm also runs on the server.

System operations and control flows

To illustrate how iMouse works, we use a fire emergency scenario, as Figure 1 shows.

On receiving the server's command, the static sensors form a treelike network to collect sensing data. Suppose static sensors *A* and *C* report unusually high temperatures, which the server suspects to indicate a fire emergency in the sensors' neighborhoods.

The server notifies the users and dispatches mobile sensors to visit the sites. On visiting *A* and *C*, the mobile sensors take snapshots and perform in-depth analyses. For example, the reported images might indicate the fire's source or identify inflammable material in the vicinity and locate people left in the building.

Each static sensor runs the algorithm in Figure 3. The server periodically floods a tree-maintenance message to maintain the WSN. It also records each static sensor's location and state, which is initially set to normal. Tree-maintenance messages help the static sensors track their parent nodes. To distinguish new from old messages, tree-maintenance messages are associated with unique sequence numbers. The goal is to form a spanning tree in the WSN.

When a sensor receives an input above a threshold, indicating an event, the sensor reports that event to the server. To avoid sending duplicate messages, each sensor keeps a variable event flag to indicate whether it has already reported that event. When a sensor detects an event and the event flag is false, the sensor reports that event and sets the flag to true. The server collects multiple events and assigns them to mobile sensors in batches. When a mobile sensor visits an event site, it asks the local sensor to clear its event flag.

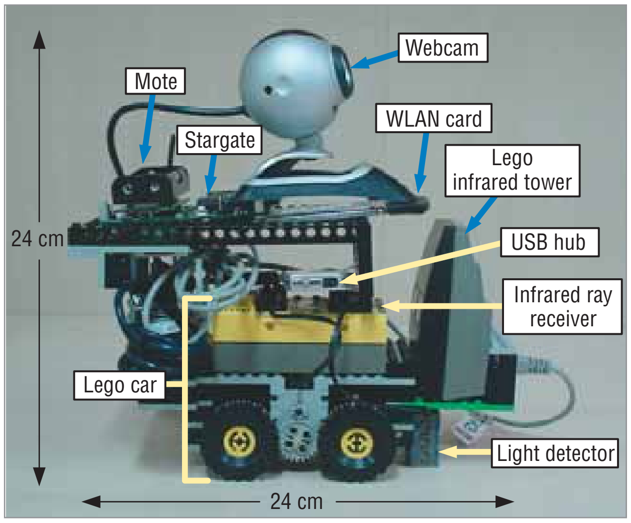


Figure 2. The mobile sensor. Attached to the Stargate processing board are a mote, a webcam, and an IEEE 802.11 WLAN card. A Lego car provides mobility.

Mobile sensor dispatch and traversal problems

Because mobile sensors are battery powered, we assign them to emergency sites to conserve their energy as much as possible. Specifically, we consider a set *L* of *m* emergency sites to be visited by a set *S* of *n* mobile sensors, where each site must be visited by one mobile sensor. We allow an arbitrary relationship between *m* and *n*. The goal is to maximize the mobile sensors' total remaining energy after sites are visited.

Our dispatch solution depends on the relationship of *m* and *n*. When $m <= n$, we can convert the problem to one of finding a maximum matching in a weighted bi- partite graph $G = (S \cup L, S \times L)$, where the vertex set is $S \cup L$ and the edge set is the prod-

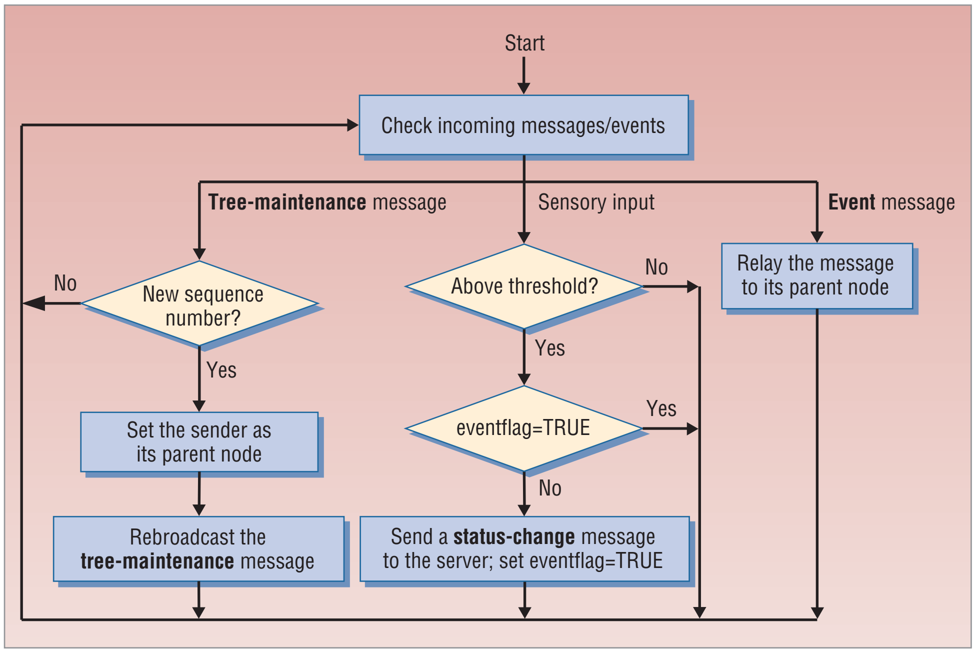

Start
Check incoming messages/events
Tree-maintenance message
Sensory input
Event message
No
New sequence number?
Above threshold?
No
Relay the message to its parent node
Yes
Yes
Set the sender as its parent node
eventflag=TRUE
Yes
No
Rebroadcast the tree-maintenance message
Send a status-change message to the server; set eventflag=TRUE

Figure 3. The algorithm executed by static sensors. Three types of messages activate a static sensor: tree-maintenance message, sensory input, and event message.

uct $S \times L = \{(s_i, l_j) | s_i \in S, l_j \in L\}$. We set the weight of (s_i, l_j) to $e_i - e_{mv} \times d(s_i, l_j)$, where e_i is the current energy of s_i; e_{mv} is the energy cost for a mobile sensor to move by one unit; and $d(s_i, l_j)$ is the distance from s_i's current location to l_j. The solution is the maximum matching P of G, which we can find through traditional maximum-weight matching solutions.[2] Alternatively, we can set our objective to minimizing mobile sensors' total moving distances. We can also use maximum-matching to achieve this by setting the weight of (s_i, l_j) to $-e_{move} \times d(s_i, l_j)$.

When $m > n$, some mobile sensors must visit multiple sites. To solve this problem, we divide emergency sites into n clusters (for example, by the classical K-means method) and assign each group to one mobile sensor. In this case, each mobile sensor's cost will include moving to the closest site in each group and then traversing the rest of the sites one by one. Given a set of locations to be visited, we can use a heuristic to the traveling salesman problem[2] to determine the traversal order.

IMPLEMENTATION AND EXPERIMENTAL RESULTS

Our static sensors are MICAz motes (www.xbow.com/Products/productdetails.aspx?sid=164). A MICAz is a 2.4-GHz, IEEE 802.15.4-compliant module allowing low-power operations and offering a 250-Kbps data rate with a direct-sequence spread-spectrum (DSSS) radio.

The Stargate processing platform consists of a 32-bit, 400-MHz Intel PXA-255 XScale reduced-instruction-set computer (RISC) with a 64-Mbyte main memory and 32-Mbyte extended flash memory. It also has a daughterboard with an RS-232 serial port, a PCMCIA slot, a USB port, and a 51-pin extension connector, which can be attached to a mote. It drives the webcam through a USB port and the IEEE 802.11 WLAN card through its PCMCIA slot. The Stargate controls the Lego car via a USB port connected to a Lego

Related Work in Wireless Surveillance

Traditional visual surveillance systems continuously videotape scenes to capture transient or suspicious objects. Such systems typically need to automatically interpret the scenes and understand or predict actions of observed objects from the acquired videos. For example, Wu-chi Feng and his colleagues proposed a video-based surveillance network in which an 802.11 WLAN card transmits the information that each video camera captures.[1]

Researchers in robotics have also discussed the surveillance issue.[2] Robots or cameras installed on walls identify obstacles or humans in the environment. These systems guide robots around these obstacles. Such systems normally must extract meaningful information from massive visual data, which requires significant computation or manpower.

Some researchers use static WSNs for object tracking.[3,4] These systems assume that objects can emit signals that sensors can track. However, results reported from a WSN are typically brief and lack in-depth information. Edoardo Ardizzone and his colleagues propose a video-based surveillance system for capturing intrusions by merging WSNs and video-processing techniques.[5] The system complements data from WSNs with videos to capture the possible scenes with intruders. However, cameras in this system lack mobility, so they can only monitor some locations.

Researchers have also proposed mobilizers to move sensors to enhance coverage of the sensing field[6] and to strengthen the network connectivity.[7] Other work addresses the pursuer-evader game, in which a pursuer must intercept an evader in the field with the assistance of WSNs.[8] To our knowledge, no one has adequately addressed the integration of WSNs with surveillance systems, which motivates us to propose the iMouse system.

References

1. W.C. Feng et al., "Panoptes: Scalable Low-Power Video Sensor Networking Technologies," *ACM Trans. Multimedia Computing, Comm., and Applications*, vol. 1, no. 2, 2005, pp. 151-167.
2. J.H. Lee and H. Hashimoto, "Controlling Mobile Robots in Distributed Intelligent Sensor Network," *IEEE Trans. Industrial Electronics*, vol. 50, no. 5, 2003, pp. 890-902.
3. X. Ji et al., "Dynamic Cluster Structure for Object Detection and Tracking in Wireless Ad-Hoc Sensor Networks," *Proc. IEEE Int'l Conf. Comm.*, IEEE Press, 2004, pp. 3807-3811.
4. W. Zhang and G. Cao, "DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks," *IEEE Trans. Wireless Comm.*, vol. 3, no. 5, 2004, pp. 1689-1701.
5. E. Ardizzone et al., "An Integrated Architecture for Surveillance and Monitoring in an Archeological Site," *Proc. ACM Int'l Workshop Video Surveillance and Sensor Networks*, ACM Press, 2005, pp. 79-85.
6. N. Heo and P.K. Varshney, "Energy-Efficient Deployment of Intelligent Mobile Sensor Networks," *IEEE Trans. Systems, Man and Cybernetics—Part A: Systems and Humans*, vol. 35, no. 1, 2005, pp. 78-92.
7. P. Basu and J. Redi, "Movement Control Algorithms for Realization of Fault-Tolerant Ad Hoc Robot Networks," *IEEE Network*, vol. 18, no. 4, 2004, pp. 36-44.
8. C. Sharp et al., "Design and Implementation of a Sensor Network System for Vehicle Tracking and Autonomous Interception," *Proc. 2nd European Workshop Wireless Sensor Networks*, IEEE Press, 2005, pp. 93-107.

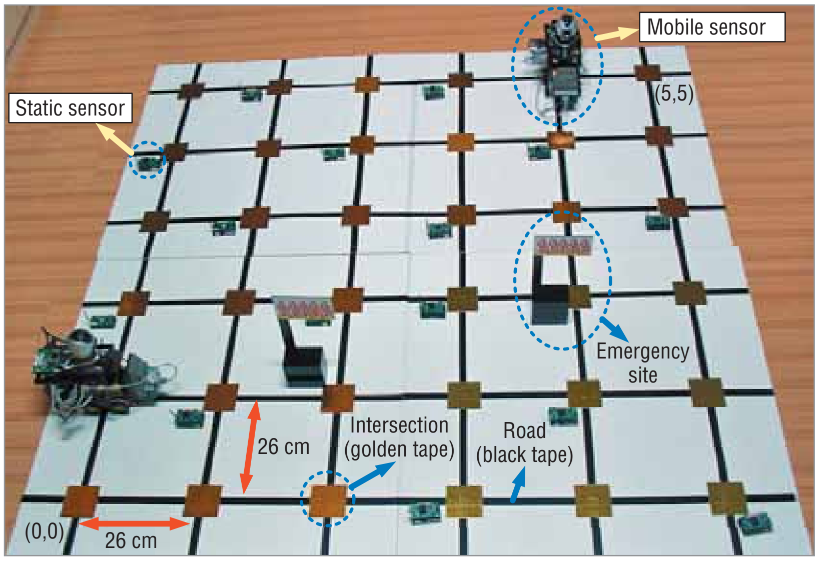


Figure 4. A 6 × 6 grid-like sensing field used in our experiment. Colored tape placed on the floor (black for roads, golden for intersections) is used to navigate the Lego car.

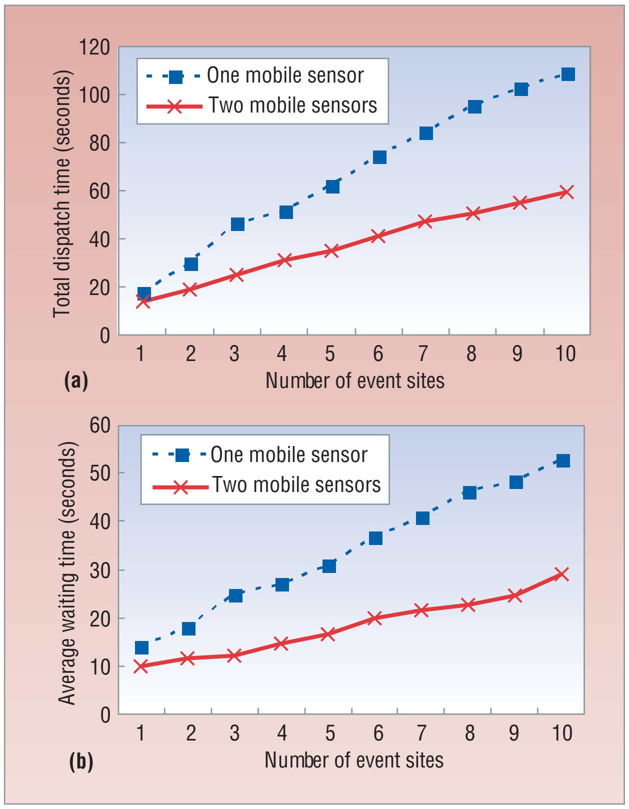


Figure 5. Experimental performance of (a) dispatch time and (b) average waiting time. As the graphs show, using two mobile sensors reduces dispatch and waiting times.

infrared tower, as Figure 2 shows. An infrared ray receiver on the front of the Lego car receives commands from the tower, and two motors on the bottom drive the wheels.

Navigating a mobile sensor or robot is difficult without some auxiliary devices. David Johnson and colleagues used wallboard cameras to capture mobile sensors' locations,[3] while Jang-Ping Sheu and his colleagues suggested using signal strength to do so.[4]

Our current prototype uses the light sensors on the Lego car to navigate mobile sensors. We stick different colors of tape on the ground, which lets us easily navigate the Lego car on a board. In our prototyping, we implemented an experimental 6 × 6 grid-like sensing field, as Figure 4 shows. Black tape represents roads, and golden tape represents intersections. We constructed the system by placing two mobile sensors and 17 static sensors on the sensing field. For static sensors, a light reading below 800 watts simulates an event, so we cover a static sensor with a box to model a potential emergency.

We use a grid-like sensing field and a grid-like static sensor deployment only for ease of implementation. In general, the static WSN's topology can be irregular.

Three factors affect the mobile sensors' dispatch time:

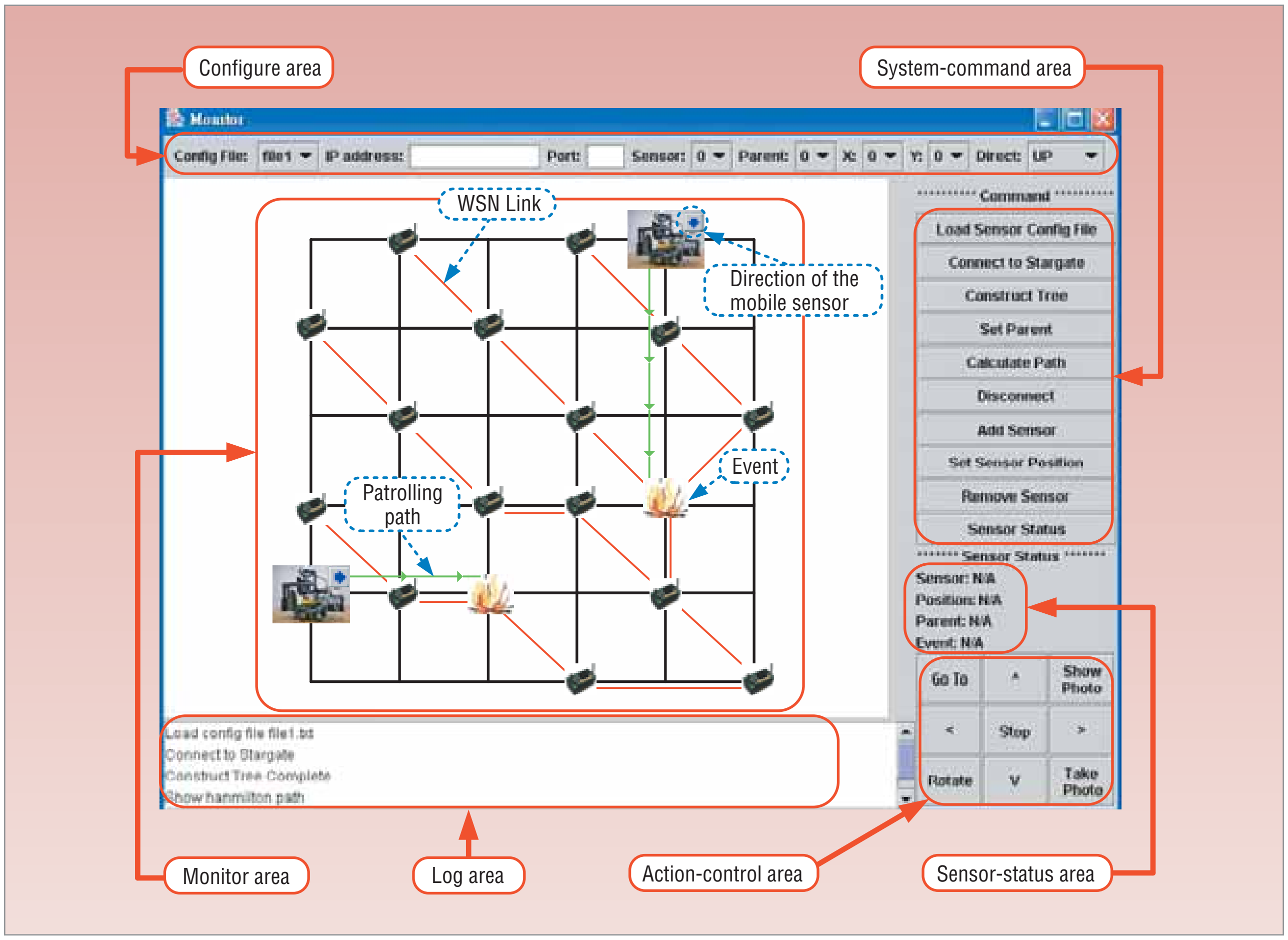
- the time that a mobile sensor takes to cross one grid-unit (about 26 centimeters),
- the time that a mobile sensor takes to make a 90-degree turn, and
- the time that a mobile sensor takes to make snapshots and report the results.

In our current prototype, the times are 2.5, 2.2, and 4.0 seconds, respectively.

Figure 5 shows experimental results with one mobile sensor initially placed at (0, 0) and two mobile sensors placed at (0, 0) and (5, 5). We generate some random events and evaluate the dispatch time (from when the server is notified of these events to when all event sites are visited) and the average time of each site (from when an event is detected to when a mobile sensor visits the site). Clearly, using two mobile sensors significantly reduces dispatch and waiting times.

At the external server, users monitor the system's status and control mobile sensors through a user interface, as Figure 6 shows. The user interface includes six major components:

- The *configure* area lets users input system configuration information, such as mobile sensors' IP addresses, ports, and sensors' positions.
- The *system-command* area provides an interface to let users control the overall system, such as issuing a



Config File: file1 IP address: Port: Sensor: 0 Parent: 0 X: 0 Y: 0 Direct: UP

Sensor: N/A
Position: N/A
Parent: N/A
Event: N/A

```
Load config file file1.txt
Connect to Stargate
Construct Tree Complete
Show hanmilton path
```

Figure 6. User interface at the external server. The interface consists of six areas through which the user can monitor the system's status and control the mobile sensors.

tree-maintenance message, adjusting the WSN's topology, and connecting and disconnecting a specified mobile sensor.

- The *sensor-status* area shows the current status of a static sensor being queried.
- The *action-control* area lets users control the mobile sensors' actions, including movement and taking snapshots.
- The *monitor* area shows the WSN's network topology and the mobile sensors' patrolling paths. When a sensor detects an event, a fire icon appears in the corresponding site.
- The *log* area displays some of the system's status messages.

SIMULATION RESULTS

Our experiments considered only grid networks. To help us understand the sensor dispatch problem in general irregular WSNs, we developed a simulator. We evaluated the average waiting time for an event location being visited and mobile sensors' total energy consumption. We compared a greedy algorithm against the *K*-means algorithm.

Given a set *S* of mobile sensors and a set *L* of emergency sites, the greedy algorithm contains a sequence of iterations. In each iteration, we assigned the mobile sensor with the smallest distance to the nearest location. We repeated this process until all mobile sensors were assigned to locations. If there were unvisited locations, the first mobile sensor reaching its destination picked its next location in the same greedy manner. This continued until all locations were visited.

In our simulation, the sensing field was 15×15 meters. We assumed that moving one meter takes one Joule and 10 seconds. A mobile sensor moves in a straight line from one location to another. Each experiment had 100 rounds, and in each round a certain number of events were generated at random locations. After each round, mobile sensors stayed at their final destinations and waited for the next schedules. We marked each simulation result with a 90 percent confidence interval.

Our comparison results under different numbers of event sites and mobile sensors showed that the greedy algorithm performs better than the *K*-means algorithm because of its time-critical nature. The K-means algorithm gives mobile sensors unbalanced job assignments, causing some event sites to wait longer even when some mobile sensors are idle. On the other hand, the *K*-means algorithm is more energy-efficient because of its clustering approach, which exploits event locality.

The iMouse system integrates WSN technologies into surveillance technologies to support intelligent mobile surveillance services. We can enhance or extend iMouse in several ways. First, we can improve mobile sensor navigation by, for example, integrating localization schemes to guide mobile sensors instead of using color tapes. Second, we can exploit coordination among mobile sensors, especially when they're on the road. Finally, we need to further investigate how we can use mobile sensors to improve the network topology. ■

Acknowledgments

Yu-Chee Tseng's research is cosponsored by Taiwan's Ministry of Education ATU Program; National Science Council of Taiwan grants no. 93-2752-E-007-001-PAE, 96-2623-7-009-002-ET, 95-2221-E-009-058-MY3, 95-2221-E-009-060-MY3, 95-2219-E-009-007, 95-2218-E-009-209, and 94-2219-E-007-009; Taiwan's Ministry of Economic Affairs under grant no. 94-EC-17-A-04-S1-044; the Industrial Technology Research Institute of Taiwan; Microsoft; and Intel.

References

1. T. He et al., "Range-Free Localization Schemes for Large-Scale Sensor Networks," *Proc. 9th ACM Int'l Conf. Mobile Computing and Networking* (MobiCom 03), ACM Press, 2003, pp. 81-95.
2. J.R. Evans and E. Minieka, *Optimization Algorithms for Networks and Graphs*, 2nd ed., Marcel Dekker, 1992.
3. D. Johnson et al., "Mobile Emulab: A Robotic Wireless and Sensor Network Testbed," *Proc. 25th Ann. IEEE Conf. Computer Comm.* (Infocom 06), IEEE Press, 2006.
4. J.P. Sheu, P.W. Cheng, and K.Y. Hsieh, "Design and Implementation of a Smart Mobile Robot," *Proc. IEEE Int'l Conf. Wireless and Mobile Computing, Networking and Comm.* (WiMob 05), IEEE Press, 2005, pp. 422-429.

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

You-Chiun Wang is a postdoctoral research associate in the Department of Computer Science at National Chiao Tung University. His research interests include wireless communication and mobile computing, QoS management and wireless fair scheduling, mobile ad hoc networks, and wireless sensor networks. Wang received a PhD in computer science from National Chiao Tung University. Contact him at wangyc@cs.nctu.edu.tw.

Kai-Yang Cheng is a master's student in the Department of Computer Science at National Chiao Tung University. His research interests include wireless communication and mobile computing, mobile ad hoc networks, and wireless sensor networks. Cheng received an MS in computer science and information engineering from National Chiao Tung University. Contact him at kycheng@cs.nctu.edu.tw.

Yao-Yu Hsieh is a graduate student in the Department of Computer Science at National Chiao Tung University. His research interests include wireless communication and mobile computing, mobile ad hoc networks, and wireless sensor networks. Hsieh received a BS in computer science and information engineering from National Chiao Tung University. Contact him at shiehyyg@cs.nctu.edu.tw.