Wireless Sensor Networks for Monitoring of Large Public Buildings

Murat Demirbas
Department of Computer Science and Engineering
University at Buffalo, SUNY
Buffalo, NY, 14260

Abstract

Recent advancements in micro-electromechanical-systems made it feasible to deploy low-cost, self-configuring wireless sensor networks for monitoring an area of interest with fine granularity. Due to its commercial potential, monitoring of large public buildings is a significant emerging application area for wireless sensor networks. Wireless sensor networks can be deployed for monitoring the response of structures to strain and ambient vibration (e.g, wind, earthquakes), monitoring and controlling of indoor environment (e.g., lighting, heating, air quality), and helping in extreme event response (e.g., detecting congested exits, finding safe routes during an evacuation). Here we provide a brief summary of the state-of-the-art in wireless sensor networks, investigate the feasibility of monitoring large public buildings using wireless sensor networks, and list some of the open research problems in this domain.

1 Introduction

A wireless sensor network consists of potentially hundreds of sensor nodes—which are, in essence, mini-computers with multi-modal sensing capability, such as vibration, light, heat, and acoustic sensing [22]. Sensor nodes sport a built-in radio and communicate wirelessly within 150 feet. Upon deployment, the nodes configure themselves in to a network, and data collected by several nodes in the network are forwarded to a basestation (e.g., a laptop) in a multi-hop fashion, allowing a real-time and fine-grain monitoring of an area of interest. Although the technology is quite new, it witnessed a quick acceptance; sensor networks are already deployed for environmental monitoring (monitoring nesting behavior of endangered birds in a remote island [33]), precision agriculture (monitoring of temperature and humidity in vineyards [5]), and military and surveillance purposes (classification and tracking of trespassers [2]).

The wide popularity of wireless sensor networks and their quick adoption for several applications can be attributed to their:

- *Ease of deployment.* Since sensor nodes communicate wirelessly, there is no need for a communication infrastructure set-up. Using energy-conservation techniques (putting nodes to sleep when they are idle), the nodes can last for months on a pair of AA batteries without the need for any wall-power. Furthermore, wireless sensor networks can be programmed to be self-configuring, enabling an ad-hoc mode of deployment. Therefore, deployment can be as simple as dropping nodes at certain locations in the area of interest.

- *Low-cost of deployment.* Sensor nodes are built using off-the-shelf cheap components. The nodes complement what they lack in reliability/quality with redundancy as they are deployed in large numbers. With mass production, a sensor node will cost as low as a couple of dollars in the near future. Even in this early phase, a sensor node is competitively priced for about $150 (exact pricing will vary based on the sensing modalities requested).

- *Fine grain of monitoring.* Since they are inexpensive and easy to deploy, it is possible to deploy wireless sensor nodes densely for finer-grain of monitoring.

Due to its commercial potential, monitoring of large public buildings is a significant emerging application area for wireless sensor networks. In this context, wireless sensor networks would be useful in *structural monitoring*, *in-door environmental monitoring*, and *extreme event response* scenarios.

Traditionally, structural monitoring systems have been designed using coaxial cables to transfer data from sensors to centralized data repositories. The extensive lengths of coaxical cables not only drive the cost of wired monitoring high and limit the extent of monitoring to less number of sensors than desired, but they also complicate the deployment since routing the cables and protecting the cables is a very labor-intensive task. Wireless sensor networks offer an economical, simple method for real-time and fine-grain monitoring of structures during ambient vibrations and seismic disturbances. Such monitoring of a structure starting from the construction phases throughout the actual use could enable us to perceive what is happening in detail and will potentially provide new insights about construction quality and performance of critical structural components in a building throughout its service life.

Monitoring of indoor environmental quality can vary from monitoring/controlling lighting and heating conditions and air quality to detecting presence of bio-chemical agents. Again, using wireless sensor networks a finer-granularity monitoring and a timely response can be achieved. Moreover, since wireless sensor networks are easy to deploy they are unaffected, unlike wired networks, by the interior configuration changes of offices in large buildings.

In an extreme events scenario where a portion of a building may be severely damaged by fire or an explosion, an early warning from the embedded sensor network through real-time structural strength degradation analysis can inform the occupants of the potential danger. Wireless sensor networks would not only facilitate an evacuation by providing real-time evacuation direction for occupants, but can also be very useful for enabling rescuers to locate survivors under collapsed structures in the aftermath of a collapse.

Contributions of the paper. This paper is intended to be an introduction to wireless sensor networks—with an emphasis on structural and environmental monitoring applications—for engineers and scientist unfamiliar with the computer science and engineering concepts. We try to give a thorough but general survey of the area and refer to several papers in the computer science and engineering literature for more detailed information. We believe it is important to set forth the capabilities and shortcomings of wireless sensor networks, as elaborating the shortcomings of wireless sensor networks will enable researchers to see the open research problems in this domain and listing the capabilities of wireless sensor networks will prevent researchers from misidentifying some of the easily solved problems as major challenges and speed up the progress in the field by enabling technology transfer from the computer engineering research on wireless sensor networks.

We present a brief survey of the state-of-the-art in wireless sensor network hardware platforms, software platforms, middleware services, and real-world deployments of monitoring applications in Section 2. In Sections 3.1, 3.2, and 3.3, we discuss the hardware requirements, software requirements, and open research challenges in monitoring of structures, in-door environments, and extreme events, respectively. Finally, we discuss some of the non-technical challenges in Section 4, and conclude in Section 5.

2 State-of-the-art in wireless sensor networks

In this section, we review the hardware and software platforms for wireless sensor networks, discuss the middleware services required for monitoring applications, and present some of the real-world deployments.

2.1 Hardware platforms

We can classify the hardware platforms roughly under three categories: grain-sized, matchbox-sized, and brick-sized nodes.

Grain-sized nodes. Radio Frequency IDentification (RFID) tags [29] is an example of this category. The computation power of RFID tags is severely limited; the most common use scenario for RFID tags is the transmission of the unique ID associated with the device. As such, one of their major application areas is inventory control, where they may eventually replace the bar-codes. Some RFID tags do not require any batteries and instead are powered by inductive coupling to a transmission of a reader device for transmitting a message back to the reader. RFID tags are available commercially at very low prices. A severe disadvantage for this platform is that it is hard to add any interesting sensing capability to this extremely small devices.

Another example is the Spec [13] architecture. This mm-sized nodes have 3K RAM and built-in radio. They are produced as a research prototype and are not available commercially. Although their power consumption is very low and they are very small in size, interfacing

them to sensors is a problem.

Matchbox-sized nodes. A representative architecture and the most widely accepted platform is the UC Berkeley mote platform [22]. The mote platform uses an 8-bit microprocessor, such as ATMEGA 128, ATMEL 8535, or Motorola HCS08, with a 4MHz CPU. Typically the motes have 4Kb RAM which is used for holding run-time state (values of the variables) of the program, 128Kb programmable Flash memory where the application program is downloaded (either via a programmer-board or wirelessly), and an additional Flash memory storage space up to 512Kb.

The motes provide a 51-pin connector for a sensorboard to be easily connected to the mainboard which holds the microprocessor and the radio chip. Light sensor, microphone, temperature, and humidity sensors are the most common and inexpensive sensors available. A two-axis accelerometer with a resolution of $\mp$2mg as well as a magnetometer are commercially available. It is possible to combine all these sensors on a single sensorboard and extra sensor boards can be stacked on top of a sensorboard for adding new sensing modalities. Due to the modular hardware design, prototyping new sensors on a basic sensorboard is simple.

Mica2 [22] uses Chipcon CC1000 radio chip and a custom radio-stack for communication. The transmission capacity is 40Kbits per second. CC1000 radios use the 916 Mhz or 433 Mhz radio-band for communication. 916 Mhz radio might be more preferable since its permeability through walls is better than 433Mhz, whereas, on the other hand, there might be more household devices that could potentially interfere in the 916 Mhz band. Zigbee [15] (a.k.a. 802.15.4) is a new type of radio-stack becoming popular. Zigbee uses 2.4Ghz communication. Mica mote series are also available with Zigbee radios [22].

Two AA battery, a total of 3V, power the mote platform. The mote draws 60mW power when active (CPU and radio on), and 0.036mW power when sleeping. The main consumer of power is the radio component. Even when no message is being transmitted, if the radio is on (e.g., waiting for messages to route) still a comparable amount of power is used. A mote will last only a couple of days in this mode. However, when put to sleep mode, with occasional wake up periods, it is possible to achieve deployment times close to a year (exact lifetime will depend on application characteristics). Battery dominates the form factor of the nodes; using a smaller watch battery, or no battery (e.g., by using sun light or ambient vibrations to scavenge energy) it is possible reduce the form factor of the nodes significantly.

Mote platform is commercially available at Crossbow [22]. As of this writing in 2005, Mica2 motes cost about $80 for the main-board. A sensorboard consisting of light, sound, and temperature is for $70 and a sensorboard that also includes an accelerometer and a magnetometer in addition to above is about $120. Telos motes are available for $80 for the main-board, and costs $120 if temperature, humidity, and light sensors are to be included. Since these platforms use off-the-shelf components, it is feasible to get the cost of production down significantly for mass production.

There are some other research prototype nodes that are not commercially available, such as the BTNodes [4] which use Bluetooth radio, that draw more power than CC1000 or Zigbee radios. There has been also some prototypes from the structural and environmental

engineering community [18,36,37]. These nodes are bulkier compared to the node platform. They use ProxLink or 9XCite radios that draw significantly more power than CC1000 or Zigbee radios.

Brick-sized nodes. Typical examples are mini linux computers communicating through 802.11 radios, such as the Stargate nodes [32]. These nodes are more powerful computationally and their 802.11 radios have higher bandwidth compared to the mote platforms. However, a big disadvantage for these nodes is that they also require more energy. It is not feasible to use AA batteries to power these nodes. These nodes are generally used as a gateway between the Internet and wireless sensor networks consisting of motes.

2.2 Software platforms

TinyOS [14], developed by UC Berkeley, is a popular operating system for wireless sensor networks. TinyOS features a *component-based architecture*; software is written in modular pieces called components. Each component denotes the interfaces that it provides. An interface declares a set of functions called commands that the interface provider implements and another set of functions called events that the interface user should be ready to handle. It is then easy to link components together by "wiring" their interfaces to form larger components (and software), similar to the way one can build larger structures using Lego blocks.

TinyOS provides a component library that includes network protocols, services, and sensor drivers. An application consists of (1) a component written by the application developer and (2) the library components that are used by the components in (1). An application developer writes only the application component that describes the sensors used in the application, the middleware services configured with the appropriate parameters based on the needs of the application (we discuss middleware services in Section 2.3).

Developing an application over TinyOS provides many benefits over coding the application from scratch as TinyOS provides useful abstractions that leads to separation of concerns, efficient concurrency control via threads, and reusability via easy reconfiguration (modular design). We elaborate on these below.

Separation of concerns. TinyOS provides a proper networking stack for wireless communication that abstracts away the underlying problems and complexity of message transfer from the application developer. A major problem in wireless communication is message collisions due to the shared nature of the communication medium. When multiple nodes in a vicinity start transmitting messages simultaneously, their transmissions overlap and corrupt each other. TinyOS provides a media access control (MAC) layer that arbitrates access to the channel using CSMA/CA techniques [27] and alleviates the effects of collisions significantly. MAC layer also detects bit-level corruptions in received messages using cyclic-redundancy-check (CRC). All of these are done by the networking component transparent to the application layer. The application developer does not have to know or worry about the details of the underlying message transfer.

Concurrency control. TinyOS provides a scheduler that achieves efficient concurrency

control. As the modus operandi in sensor networks is to react to changes in the environment (i.e. message arrival, sensor acquisition) rather then interactive or batch processing, an interrupt-driven execution model is needed to achieve a quick response time for the events and capture the data. For example, a message transmission may take up to 100msec, and without an interrupt-driven approach the node would miss sensing and processing of interesting data in this period. Scheduler takes care of the intricacies of interrupt-driven execution and provides concurrency in a safe manner by scheduling the execution in small threads. In comparison, when developing an application from scratch (over the bare hardware), the programmers often are unable to provide that level of concurrency safely. As a result, custom software without any operating system support is pretty bad in terms of responsiveness and often times incorrect due to race conditions.

Modularity. TinyOS's component model facilitates reuse and reconfigurability since software is written in small functional modules. Several middleware services are available as well-documented components. Over 500 research groups and companies are using TinyOS and numerous groups are actively contributing code to the public domain [34].

A major handicap of TinyOS is that it is still not easy to use as desired. A TinyOS application developer needs to learn NesC programming language and have familiarity with distributed/concurrent computing and network systems. These requirements prohibit TinyOS to be popular among researchers and engineers who are non-computer science and engineering majors. Still, as a bottom-line, we contend that it is easier to develop an application using TinyOS leveraging on the support provided by the library components, than to attempt developing something from scratch. Several macroprogramming approaches to simplify the task of wireless sensor networks are under development, but they will take long time to accomplish any improvement in usability. There are also shrink-wrapped commercial applications for data collection, however, no packed software fits all. Also for performance critical applications, the software should be fine-tuned and cross-layer issues become important.

Mantis [1] is another open source, multi-threaded operating system written in C for the mote platform.

2.3 Middleware services

Middleware is the software that supports building of complex distributed applications. The idea behind providing middleware services is that of separation of concerns and reusability. Several environmental monitoring applications require synchronized timers at the nodes for accurate timestamping of sensed data, localization of the nodes for tagging sensed data with location information, sleep schedules at the nodes to elongate deployment lifetime without jeopardizing loss of interesting event detections, and a mechanism for giving on-the-fly directions to certain regions of the network to change the modality and method of sensing. Instead of developing such services from scratch for each environmental monitoring application, the researchers have developed general purpose, customizable, and reusable software for these services. These software has been bundled as TinyOS components and made available at the public domain [34]. When a time synchronization capability is needed for an application,

one can simply wire a TinyOS time synchronization component for adding this service to the application. This way, the time and effort required for developing applications are reduced greatly.

Next, we mention some of the implemented services under TinyOS.

Time synchronization. Since nodes have timers with varying skews, a time synchronization protocol is necessary to keep a consistent and accurate global clock at each node in the network. Using MAC layer timestamping and clock skew estimation several groups developed services that synchronize nodes within microsecond level accuracy [7,20]. For environmental monitoring applications a light-weight post-facto synchronization approach that avoids any synchronization messages is possible. In this scheme the nodes keep accurate track of the delay induced on a sensed data by summing up the stay-time of the data at each node as the data is forwarded toward the basestation. When the basestation receives the data, it uses its own clock value as a reference and by subtracting the delay induced on the data during transmissions, it figures out the sensing time of the data [39].

Localization. For several environmental monitoring applications, it is important to tag the sensed data with the node that senses the data and the location/coordinates of the node. GPS [8] is costly, consumes a lot of energy, and does not work in-doors. Preconfiguring each node with location/coordinate information would be very tedious for big deployments. Several protocols have been proposed to solve the localization problem by enabling nodes to use information about nearby anchors (nodes that are preconfigured with location information) to triangulate and estimate their own locations [11, 28]. However, these protocols are not robust for real-world deployments yet. For real-world deployments preconfiguration and application-based solutions are used.

Power management. It is possible to elongate deployment lifetime from days to several months by selectively putting some nodes to sleep. However, it is important to coordinate which nodes to put to sleep and when to wake them up, otherwise some transmitted messages will be lost since all the receivers are in sleep mode. Several topology control protocols [12,40] have been proposed, but the final selection should be based on the application characteristics. TinyOS provides support for motes to enter and exit sleep mode.

Routing. We restrict our discussion of routing protocols to the *convergecast traffic* that emerge in environmental monitoring applications. Here, the nodes send data to a basestation for collection. This traffic is usually forwarded over multihops, that is, each node acts as a relay/router for some nodes farther away to the basestation than itself. Most common approach for convergecast is to embed a spanning tree—rooted at the basestation— over the networks, and use this structure for convergecast [38]. LGR [6] embeds a logical grid on the network and emphasizes local recovery of routing when some nodes and some link connectivity fail.

Even though CC1000 radio can transmit 40Kbits/sec, a node should regulate its data transmission rate as it also needs to relay data from other nodes. Since multiple nodes are originating traffic, rate control and contention management are major issues that the routing layer needs to address. In response to an event multiple nodes may start sending several

packets in a bursty manner, which may lead to more than %50 data loss [41]. Using implicit acknowledgements, arbitrated retransmission, and prioritization of traffic, reliable transmission can be achieved [41].

Querying/Reprogramming. It is useful to be able to change the modality of the sensed data dynamically at run-time. For example, we may want to increase the sampling rate of accelerometer data for some nodes to closely monitor a phenomena, or turn off the acoustic sensors on some nodes since they are not needed. TinyDB [19] provides an SQL [21] like high-level language for submitting such queries to the basestation, which disseminates them to the network. More drastically, we might occasionally want to upgrade the programs on the nodes to fix a bug or add a feature without having to take-down and re-deploy the network. Trickle [16] is a reprogramming protocol that achieves reliable wireless reprogramming of the nodes.

2.4 Real-world deployments

Several wireless sensor networks are deployed in the recent couple of years. On Great Duck Island off the coast of Maine, researchers deployed a sensor network to monitor (via light, temperature, and infrared–presence detection– sensors) the nesting grounds of seabirds [33]. The network consisting over 100 Mica2 [22] nodes stayed operational during summer and fall of 2002, and also on summer of 2003. In a vineyard in Oregon, a 60 node sensor network is deployed for mapping growth conditions and susceptibility to fungal infections in a vineyard by monitoring temperature and humidity conditions [5]. An 80 node network is deployed for collecting readings on microclimates surrounding redwood trees in Santa Cruz via the temperature, humidity, light, atmospheric pressure sensors [35].

"A Line In the Sand" network [2] that detects, classifies, and tracks various types of objects (such as people and cars) was developed as part of the DARPA/NEST program. The field test is conducted using 90 preconfigured nodes at known locations, 78 containing magnetometer sensors & 12 containing micro-power impulse radar (MIR) sensors at the MacDill Air Force Base in Florida in summer of 2003. In December 2004, this surveillance network is scaled an area 1.3km by 300m with about 1000 sensor nodes and around 200 backbone nodes making it the largest wireless sensor network assembled to date [3]. This project provided a rich set of experiences and better understanding of the "real" problems that are posed by wireless networks of extreme scale.

Again as part of the DARPA/NEST program, a wireless sensor network-based system that detects and accurately locates shooters is developed [31]. The system consists of a large number of acoustic sensors that listen for gunshots and then triangulate shooter position. The system performance is analyzed using real measurement data obtained at a US Army MOUT (Military Operations in Urban Terrain) facility. Accuracy of the system is about 1m, and latency is only two seconds.

3 Monitoring of Large Public Buildings

In this section we discuss the hardware and software requirements for applications of wireless sensor networks in structural monitoring, in-door environmental monitoring, and extreme events response. We discuss the challenges and open problems in these applications from the computer science and engineering perspective.

3.1 Structural monitoring

Hardware requirements. Accelerometer and strain gauges are the two sensors that are most useful for structural monitoring applications. Structural monitoring applications aim to find out the natural vibration frequencies of a structure under naturally induced or forced vibrations, and use this information to detect any structural damage or wear the system might have. The stress and strain on various beams in the structure is also useful for monitoring the health of a structure.

Several structural engineering papers have presented test deployments for structural monitoring. The accelerometers used in these deployments have sensitivity on the order of micro-g's (16 bit A-D resolution may be needed) and a range of 1-2g. Sampling frequencies of 200Hz is deemed high enough to capture the major characteristics of the structural vibration.

The two-axis accelerometer (ADXL202) provided with the mote platform has a $\mp$2mg sensitivity and a range of -2g to 2g. 200Hz sampling is easily achievable under TinyOS [22]. High frequency sampling up to 6.67KHz is possible with recent additions to TinyOS component. A recent work [26] discusses integration of two high-sensitivity accelerometers (ADXL202E and Silicon Designs 1221L) with the mote architecture. Another high sensitivity and low noise accelerometer sensorboard for the mote platform is developed in [30].

Nagayama et al. developed a strain sensorboard for the mote platform with a measurement range of 1 - 2000 microstrain and a noise level of 1 microstrain [25]. These strain sensors have been tested on a structural model of a three story building and their output is found to be comparable with conventional wired strain sensors. Several strain sensor solutions, though not on the mote platform, is commercially available at [23].

Software requirements. Two main challenges for structural monitoring software is dealing with the high sampling rate of the sensors and reliable transfer of the resulting large data to the basestation.

TinyOS's event-driven concurrent execution model simplifies the task of dealing with the sampling rates of 200Hz. The time synchronization services we discussed above achieves accuracy on the order of microseconds and enable us to construct consistent global snapshots of the structures behavior from the collected data.

The networking challenges for reliable transmission of the large amount of bursty traffic are discussed in [26, 39, 41]. In [39], wavelet compression of the data and event thresholding is suggested for overcoming the bandwidth limitations. In the proposed scheme, nodes perform

wavelet compression of the sensed data only if the sensed value is over a certain threshold and transmit low-resolution compressed summaries of data for reducing the traffic.

Recently there has been considerable work by structural engineering community on wireless monitoring of structures. In [18, 36, 37], a hardware platform, similar to the mote platform, is built from off-the shelf components. These nodes use ProxLink or 9XCite modems for wireless communication and require significantly more power (requires 5 AA batteries) than the mote platform. The test deployments with this platform, for example on Geumdang Bridge with 14 nodes [36], consist of single-hop networks, the basestation is always within 150 feet of the nodes in the network. Since the software for the deployments are written from scratch, no operating systems support for MAC layer arbitration of channel access is available.

Open research problems. One of the major challenges for deploying wireless sensor networks for real-world structural monitoring is the lifetime of deployment. Effective power management and topology control protocols are needed to increase the lifetime of deployments. The lifetime for sensors that are easily accessible (and hence easier to replace the batteries) should be on the order of months, and for deeply embedded ones (for example inside the building structure) the lifetime should be on the order of decades. These latter nodes should be sleeping most of the time and should wake up only in response to an extreme event: such as seismic vibration over a threshold. Powering those nodes through other means, maybe as in RFID or parasitically from vibrations should also be investigated.

Model-based monitoring [10] is a promising idea for energy-efficiency. The idea here is to first construct a model of the characteristics of the system and transmit data only when a reading differs significantly from that estimated by the model. Other promising ideas for power management includes tradeoffs with latency and accuracy. Radio must be turned off for power saving, however, then additional delays will be incurred for the sensed data to be transferred via multiple hops. BMAC [27] studies the tradeoffs between energy saving and latency. Using guidance from structural engineers it is also possible to consider approximations for the sensed data.

Localization of the nodes is another remaining challenge. The environmental monitoring applications in Section 2.4 used some GPS-enabled nodes and ranging based localization solutions. In real-world structural monitoring deployments especially for in-building deployments GPS information would not be available, moreover line-of-sight ranging-based techniques may not be applicable. The test deployments for structural monitoring above used preconfigured location information at the nodes, however, this approach would not scale for large networks. Based on the feedback from structural engineers it may be possible to develop application-dependent lightweight localization services for structural monitoring.

Networking services may also need to consider some quality of service guarantees in reliable transfer. Prioritization of traffic could be useful. We might want to deliver data from half of the nodes at each cluster immediately to achieve a short reaction time and fairness; the remaining data trickle later on to achieve completeness. Another promising idea for data collection is using data mules, that is exploiting the natural mobility in the system. For example, in a bridge monitoring scenario, the readings from the sensor nodes can be

uploaded to passing cars (only to those equipped with nodes) and hence carried in a parasitic manner until the toll-booths where it is uploaded to the basestation responsible for the bridge monitoring.

Finally, another significant challenge for real-time deployments is the management of the sensor network. Robust querying and reprogramming services are needed to this end. It is also important to maintain the accuracy and calibration of the sensed data. Temperature changes affect accelerometers, clocks, and other sensors. Using the temperature sensor on the nodes, though, these affects may be compensated.

3.2 In-door environment monitoring

Hardware requirements. Temperature, humidity, true-light sensors, infrared-based presence sensors, and chemical sensors are useful for in-door environmental monitoring systems. The applications foreseen for in-door monitoring are controlling of heating, AC, or light intensity guided by the communication from these sensors. Standard sensorboards contain temperature and humidity sensors [22]. True-light sensors are available from Moteiv [24]. Infrared sensors have been used on ExScal motes [3]. Chemical sensors are composed of electrodes with specialized chemical coating that change resistance when exposed to certain chemical agents. There has been sensors for NO_x, CO_x, nerve gases, anthrax etc. However, we are not aware of any of these ported on the mote platform.

Grain-sized nodes, for example RFID nodes, could be useful for inventory control applications. RFID readers can be embedded in door frames to read RFID information as items enter or exit the room. Badge-based in-door localization systems [28] would be useful for tracking of personnel inside buildings.

Software requirements. Querying and wireless reprogramming are essential components of indoor environmental monitoring. It would be useful to be able to change the operating mode of the sensor network dynamically. For example, after office hours, the network might be put into surveillance network mode, and during office hours environmental control mode would be restored.

We have mentioned some outside environmental monitoring applications in Section 2.4, however, we are not aware of any long-lived deployment of in-door monitoring applications yet.

Open research problems. It is possible to exploit the application domain for improving the performance of these systems. In buildings, 802.11 WAPs are highly likely to be deployed. Since 802.11 radios have higher bandwidth and improve the performance of data transfer, using 802.11 WAPs as gateways for the wireless sensor networks some of the networking challenges can be alleviated. Scalability issues when hundreds of nodes are concerned can be resolved by using such heterogeneous (or multi-tiered) networks, where a cluster of nodes coordinate with nearby 802.11 nodes. Integration of RFID platforms is another open area.

3.3 Extreme event response

Hardware requirements. Extreme event detection applications could run on the nodes deployed for structural monitoring and in-door environmental monitoring. Upon detection of an extreme event, such as fire, explosion, structural strength degradation, or chemical attack, the system should warn occupants and lead them to safe exit paths. Thus, there is a need for integrating intuitive interfaces to the extreme event response systems. Moreover, since these systems can also be useful for enabling rescuers locate survivors under collapsed structures in the aftermath of a collapse, certain ruggedized nodes could be needed.

Software requirements. Real-time and dependability guarantees of software is crucial for extreme event response systems. Extreme event reports should be prioritized traffic and should be handled quickly and in a dependable manner. Grenade timers [3] would be useful for coping with any deadlock and livelock issues that can delay response.

A dual issue is the number of false-positives—premature notifications raised when there is no extreme event. The number of false positives should be kept at an acceptable level for these systems to be acceptable in real-world deployments.

Open research problems. There has been several research papers on algorithms for escape path finding while avoiding hazardous regions [9, 17]. We expect that real-world deployment challenges would be engineering challenges. These systems should be built with real-time and dependability guarantees. Formal verification techniques would be useful for proving real-time and dependability properties of these protocols to be deployed. Additionally, testing strategies should be devised to cover an extensive set of scenarios before a real deployment is attempted. There is also a need for intuitive interfaces to guide evacuees to safe exits, and to help recovery teams locate survivors.

4 Discussion

A key requirement for the monitoring of large public buildings project is an interdisciplinary collaboration between researchers from computer science and structural engineering, two disciplines which have not traditionally interacted with each other. Computer science and engineering part of the effort should be guided by the context provided by structural and environmental engineering that answer the questions "what to sense?", "where to sense?", "when to sense?", and "how to sense?". Depending on the answers to the questions such as "What level of accuracy is acceptable?", "Which events are of interest and which are ignorable?", and "What are the right places for deployment?", computer science and engineering researchers could fine-tune the performance and lifetime of the wireless monitoring systems.

Non-technical challenges should not be overlooked as overcoming them may be crucial for wide-spread adoption of these monitoring systems. For example, network management and maintenance is a major challenge. The software should be packed as an easy to use and configure application. High-level abstract programming languages, such as SQL or scripting

languages might be exported to the user for configuring the application to her customized needs.

Security and privacy issues should also be addressed before real deployments. Security is a problem for extreme-events and control-based applications, e.g., evacuation. Security issues are also important for wireless reprogramming. Privacy issues would be critical for in-door environmental monitoring applications and RFID based inventory and personnel tracking systems.

5 Concluding remarks

In this paper we provided a brief survey of the state-of-the-art in wireless sensor networks with an emphasis on structural and environmental monitoring applications. We outlined the challenges and open research problems—from the computer science and engineering perspective— for these systems to be deployed in real-world scenarios. We believe that to achieve successful real-world deployments, there is a need for collaboration between structural & environmental engineering and computer science & engineering disciplines.

References

[1] H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, B. Shucker, J. Deng, and R. Han. Mantis: System support for multimodal networks of in-situ sensors. In *2nd ACM International Workshop on Wireless Sensor Networks and Applications*, pages 50 – 59, 2003.

[2] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y-R. Choi, T. Herman, S. S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks (Elsevier)*, 46(5):605–634, 2004.

[3] A. Arora and et. al. Exscal: Elements of an extreme scale wireless sensor network. *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2005.

[4] J. Beutel, O. Kasten, F. Mattern, K. Römer, F. Siegemund, and L. Thiele. Prototyping wireless sensor network applications with btnodes. In *1st European Workshop on Wireless Sensor Networks (EWSN)*, number 2920 in LNCS, pages 323–338, Berlin, Germany, January 2004. Springer-Verlag.

[5] J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: Sensor networks in agricultural production. *IEEE Pervasive computing*, 3:38–45, 2003.

[6] Y. Choi, M. Gouda, H. Zhang, and A. Arora. Stabilization of grid routing in sensor networks. *AIAA Journal of Aerospace Computing, Information, and Communication*, 2005.

[7] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, 2002.

[8] P. Enge and P. Misra. Special issue on gps: The global positioning system. In *Proceedings of the IEEE*, pages 3–172, January 1999.

[9] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Mobile Computing and Networking*, pages 263–270, 1999.

[10] S. Goel and T. Imielinski. Prediction-based monitoring in sensor networks: taking lessons from mpeg. *SIGCOMM Comput. Commun. Rev.*, 31(5):82–98, 2001.

[11] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher. Range-free localization schemes in large scale sensor networks. *Mobicom*, 2003.

[12] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. Application specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Networking*, 2002.

[13] J. Hill. *System Architecture for Wireless Sensor Networks*. PhD thesis, UC Berkeley, 2003.

[14] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. *ASPLOS*, pages 93–104, 2000.

[15] P. Kinney. Zigbee technology: Wireless control that simply works. www.zigbee.org/resources, 2003. Whitepaper.

[16] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. *In First Symposium on Network Systems Design and Implementation (NSDI)*, 2004.

[17] Q. Li, M. Rosa, and D. Rus. Distributed algorithms for guiding navigation across a sensor network. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 313–325, New York, NY, USA, 2003. ACM Press.

[18] J. Lynch, A. Sundararajan, K. Law, A. Kiremidjian, and E. Carryer. Embedding damage detection algorithms in a wireless sensing unit for attainment of operational power efficiency. *Smart Materials and Structures*, 13(4):800–810, 2004.

[19] S. Madden, M. Franklin, J. Hellerstein, and Wei Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

[20] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. *SenSys*, 2004.

[21] J. Melton. Sql language summary. *ACM Comput. Surv.*, 28(1):141–143, 1996.

[22] Crossbow technology, mica2 platform. `www.xbow.com/Products/Wireless_Sensor_Networks.htm`.

[23] Microstrain. `http://www.microstrain.com/`.

[24] Moteiv. `http://www.moteiv.com/`.

[25] T. Nagayama, M. Ruiz-Sandoval, B. Spencer, K. Mechitov, and G. Agha. Wireless strain sensor development for civil infrastructure. *Proceedings of First International Workshop on Networked Sensing Systems*, 2004.

[26] S. Pakzad, S. Kim, G. Fenves, S. Glaser, D. Culler, and J. Demmel. Multi-purpose wireless accelerometers for civil infrastructure monitoring. *5th International Workshop on Structural Health Monitoring (IWSHM)*, 2005.

[27] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, 2004.

[28] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proceedings of the 6th Annual ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, August 2000.

[29] Rfid. `http://en.wikipedia.org/wiki/RFID`.

[30] M. Ruiz-Sandoval, B. Spencer, and N. Kurata. Development of a high sensitivity accelerometer for the mica platform. *Proceedings of International Workshop on Advanced Sensors, Structural Health Monitoring, and Smart Structures*, 2003.

[31] G. Simon, M. Maroti, A. Ledeczi, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. *Sensys*, 2004.

[32] Stargate projects. `http://platformx.sourceforge.net/Links/project.html`.

[33] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. An analysis of a large scale habitat monitoring application. *In Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[34] Tinyos project. `http://sourceforge.net/projects/tinyos/`.

[35] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, P. Buonadonna, S. Burgess, D. Gay, W. Hong, T. Dawson, and D. Culler. A macroscope in the redwoods. *In Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[36] Y. Wang, J. Lynch, and K. Law. Validation of a large-scale wireless structural monitoring system on the geumdang bridge. *9th International Conference on Structural Safety and Reliability*, 2005.

[37] Y. Wang, J. Lynch, and K. Law. Wireless structural sensors using reliable communication protocols for data acquisition and interrogation. *Proceedings of the 23rd International Modal Analysis Conference*, 2005.

[38] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 14–27. ACM Press, 2003.

[39] N. Xiu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. *In Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[40] Y. Xu, S. Bien, Y. Mori, J. Heidemann, and D. Estrin. Topology control protocols to conserve energy in wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2003.

[41] H. Zhang, A. Arora, Y. Choi, and M. G. Gouda. Reliable bursty convergecast in wireless sensor networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 266–276, New York, NY, USA, 2005. ACM Press.