CHAPTER 1

Introduction to Wireless Sensor Networking

FERNANDO MARTINCIC and LOREN SCHWIEBERT
Wayne State University, Detroit, Michigan

This chapter introduces the topic of wireless sensor networks from the applications perspective. A wireless sensor network consists of a possibly large number of wireless devices able to take environmental measurements such as temperature, light, sound, and humidity. These sensor readings are transmitted over a wireless channel to a running application that makes decisions based on these sensor readings. Authors describe some examples of proposed wireless sensor applications, and consider the following two questions to motivate an application-based viewpoint. What aspects of wireless sensors make the implementation of applications more challenging, or at least different? One widely recognized issue is the limited power available to each wireless sensor node, but there are other challenges such as limited storage or processing. What services are required for a wireless sensor network application to achieve its intended purpose? A number of widely applicable services, such as time synchronization and location determination are briefly discussed in this chapter. Other services are needed to support database requirements, such as message routing, topology management, and data aggregation and storage. As most of these topics are covered in separate chapters, this chapter serves to provide a broad framework to enable the reader to see how these different topics tie together into a cohesive set of capabilities for building wireless sensor network applications.

1.1 INTRODUCTION

A wireless sensor network consists of a possibly large number of wireless devices able to take environmental measurements. Typical examples include temperature,

Handbook of Sensor Networks: Algorithms and Architectures, Edited by Ivan Stojmenović

light, sound, and humidity. These sensor readings are transmitted over a wireless channel to a running application that makes decisions based on these sensor readings. Many applications have been proposed for wireless sensor networks, and many of these applications have specific quality of service (QoS) requirements that offer additional challenges to the application designer. In this chapter, we introduce the topic of wireless sensor networks from the perspective of the application.

Along with some examples of proposed wireless sensor applications, we consider two questions to motivate an application-based viewpoint:

1. *What aspects of wireless sensors make the implementation of applications more challenging, or at least different?*

   One widely recognized issue is the limited power available to each wireless sensor node, but other challenges such as limited storage or processing capabilities play a significant role in constraining the application development.

2. *What services are required for a wireless sensor network application to achieve its intended purpose?*

   A number of widely applicable services, such as time synchronization and location determination are briefly discussed. Other services are needed to support database requirements, such as message routing, topology management, and data aggregation and storage.

Because some of these topics are covered in separate chapters, this discussion serves to provide a broad framework to enable the reader to see how these different topics tie together into a cohesive set of capabilities for building wireless sensor network applications.

1.2 DESIGN CHALLENGES

Several design challenges present themselves to designers of wireless sensor network applications. The limited resources available to individual sensor nodes implies designers must develop highly distributed, fault-tolerant, and energy-efficient applications in a small memory-footprint. Consider the latest-generation MICAz [1,2] sensor node shown in Figure 1.1.

MICAz motes are equipped with an Atmel128L [4] processor capable of a maximum throughput of 8 millions of instructions per second (MIPS) when operating at 8 MHz. It also features an IEEE 802.15.4/Zigbee compliant RF transceiver, operating in the 2.4–2.4835-GHz globally compatible industrial scientific medical (ISM) band, a direct spread-spectrum radio resistant to RF interference, and a 250-kbps data transfer rate. The MICAz runs on TinyOS [5] (v1.1.7 or later) and is compatible with existing sensor boards that are easily mounted onto the mote. A partial list of specifications given by the manufacturers of the MICAz mote is presented in Figure 1.2.

Figure 1.1 MICAz sensor mote hardware. (Image courtesy of Crossbow Technology [3].)

For wireless sensor network applications to have reasonable longevity, an aggressive energy-management policy is mandatory. This is currently the greatest design challenge in any wireless sensor network application. Considering that in the MICAz mote the energy cost associated with transmitting a byte over the transceiver is substantially greater than performing local computation, developers must leverage local processing capabilities to minimize battery-draining radio communication. Several key differences between more traditional ad hoc networks and wireless sensor networks exist [6]:

- Individual nodes in a wireless sensor network have limited computational power and storage capacity. They operate on nonrenewable power sources and employ a short-range transceiver to send and receive messages.
- The number of nodes in a wireless sensor network can be several orders of magnitude higher than in an ad hoc network. Thus, algorithm scalability is an important design criterion for sensor network applications.
- Sensor nodes are generally densely deployed in the area of interest. This dense deployment can be leveraged by the application, since nodes in close proximity can collaborate locally prior to relaying information back to the base station.
- Sensor networks are prone to frequent topology changes. This is due to several reasons, such as hardware failure, depleted batteries, intermittent radio interference, environmental factors, or the addition of sensor nodes. As a result, applications require a degree of inherent fault tolerance and the ability to reconfigure themselves as the network topology evolves over time.

Processor	Atmel ATMega128L @ 8 MHz
Program Flash Memory	128 kilobytes
Measurement Serial Flash	512 kilobytes
Configuration electrically erasable programmable read-only memory (EEPROM)	4 kilobytes
Serial Communications	UART
Analog to Digital Converter	10 bit ADC
Other Interfaces	Digital I/O, I2C, SPI
Processor Current Draw	8 mA in active mode $< 1\,\mu$A in sleep mode
Frequency band	2400MHz to 2483.5MHz
Transmit (TX) data rate	250kbps
RF power	-24dBm to 0dBm
Receive Sensitivity	-90dBm (min), -94dBm (typ)
Adjacent channel rejection	47 dB, +5-MHz channel spacing 38 dB, -5-MHz channel spacing
Outdoor Range	75m to 100m
Indoor Range	20m to 30m
Radio Current Draw	19.7mA in receive mode 11mA (TX -10dBm) 14mA, (TX -5dBm) 17.4mA (TX 0dBm) 20 μA in idle mode (voltage regulator on) 1 μA in sleep mode (voltage regulator off)
Battery	2 AA batteries
User Interface	red, green, and yellow LED
Size	2.25×1.25×0.25in. (w/o battery pack)
Weight	0.7 oz (w/o batteries)
Expansion Connector	51 pin

Figure 1.2 MICAz mote specification [1].

- Wireless sensor networks do not employ a point-to-point communication paradigm because they are usually not aware of the entire size of the network and nodes are not uniquely identifiable. Consequently, it is not possible to individually address a specific node. Paradigms, such as directed diffusion [7,8], employ a data-centric view of generated sensor data. They identify information produced by the sensor network as ⟨attribute, value⟩ pairs. Nodes request data by disseminating interests for this named data throughout the network. Data that matches the criterion are relayed back toward the querying node.

Even with the limitations individual sensor nodes possess and the design challenges application developers face, several advantages exist for instrumenting an area with a wireless sensor network [9]:

- Due to the dense deployment of a greater number of nodes, a higher level of fault tolerance is achievable in wireless sensor networks.
- Coverage of a large area is possible through the union of coverage of several small sensors.
- Coverage of a particular area and terrain can be shaped as needed to overcome any potential barriers or holes in the area under observation.
- It is possible to incrementally extend coverage of the observed area and density by deploying additional sensor nodes within the region of interest.
- An improvement in sensing quality is achieved by combining multiple, independent sensor readings. Local collaboration between nearby sensor nodes achieves a higher level of confidence in observed phenomena.
- Since nodes are deployed in close proximity to the sensed event, this overcomes any ambient environmental factors that might otherwise interfere with observation of the desired phenomenon.

1.3 WIRELESS SENSOR NETWORK APPLICATIONS

Several applications have been envisioned for wireless sensor networks [6]. These range in scope from military applications to environment monitoring to biomedical applications. This section discusses proposed and actual applications that have been implemented by various research groups.

1.3.1 Military Applications

Wireless sensor networks can form a critical part of military command, control, communications, computing, intelligence, surveillance, reconnaissance, and targeting (C4ISRT) systems. Examples of military applications include monitoring of friendly and enemy forces; equipment and ammunition monitoring; targeting; and nuclear, biological, and chemical attack detection.

By equipping or embedding equipment and personnel with sensors, their condition can be monitored more closely. Vehicle-, weapon-, and troop-status information can be gathered and relayed back to a command center to determine the best course of action. Information from military units in separate regions can also be aggregated to give a global snapshot of all military assets.

By deploying wireless sensor networks in critical areas, enemy troop and vehicle movements can be tracked in detail. Sensor nodes can be programmed to send notifications whenever movement through a particular region is detected. Unlike other surveillance techniques, wireless sensor networks can be programmed to be completely passive until a particular phenomenon is detected. Detailed and timely

intelligence about enemy movements can then be relayed, in a proactive manner, to a remote base station.

In fact, some routing protocols have been specifically designed with military applications in mind [10]. Consider the case where a troop of soldiers needs to move through a battlefield. If the area is populated by a wireless sensor network, the soldiers can request the location of enemy tanks, vehicles, and personnel detected by the sensor network (Fig. 1.3). The sensor nodes that detect the presence of a tank can collaborate to determine its position and direction, and disseminate this information throughout the network. The soldiers can use this information to strategically position themselves to minimize any possible casualties.

In chemical and biological warfare, close proximity to ground zero is needed for timely and accurate detection of the agents involved. Sensor networks deployed in friendly regions can be used as early-warning systems to raise an alert whenever the presence of toxic substances is detected. Deployment in an area attacked by chemical or biological weapons can provide detailed analysis, such as concentration levels of the agents involved, without the risk of human exposure.

1.3.2 Environmental Applications

By embedding a wireless sensor network within a natural environment, collection of long-term data on a previously unattainable scale and resolution becomes possible. Applications are able to obtain localized, detailed measurements that are otherwise more difficult to collect. As a result, several environmental applications have been proposed for wireless sensor networks [6,9]. Some of these include habitat monitoring, animal tracking, forest-fire detection, precision farming, and disaster relief applications.

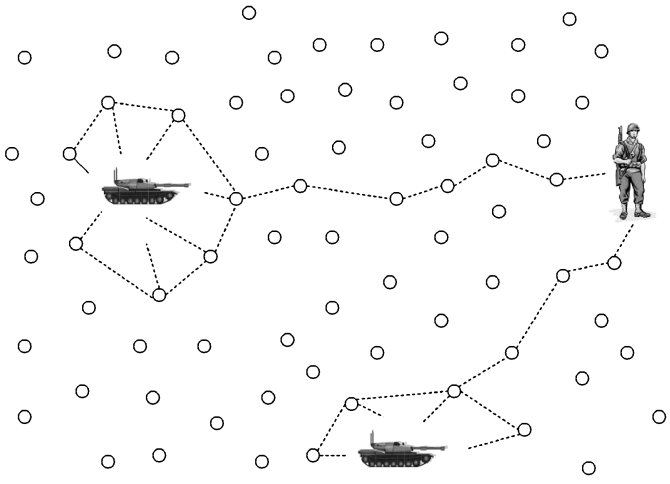

Figure 1.3 Enemy target localization and monitoring.

Habitat monitoring permits researchers to obtain detailed measurements of a particular environment in an unobtrusive manner. For example, applications such as the wireless sensor network deployed on Great Duck Island [11] allow researchers to monitor the nesting burrows of Leach's Storm Petrels without disturbing these seabirds during the breeding season. Deployment of the sensor network occurs prior to the arrival of these offshore birds. Monitoring of the birds can then proceed without direct human contact. Similarly, the PODS project [12,13] at the University of Hawaii uses wireless sensor networks to observe the growth of endangered species of plants. Data collected by the sensor network is used to determine the environmental factors that support the growth of these endangered plants. These two applications are discussed in detail in Sections 1.3.4 and 1.3.5.

Consider a scenario where a fire starts in a forest. A wireless sensor network deployed in the forest could immediately notify authorities before it begins to spread uncontrollably (see Fig. 1.4). Accurate location information [14] about the fire can be quickly deduced. Consequently, this timely detection gives firefighters an unprecedented advantage, since they can arrive at the scene before the fire spreads uncontrollably.

Precision farming [15] is another application area that can benefit from wireless sensor network technology. Precision farming requires analysis of spatial data to determine crop response to varying properties such as soil type [16]. The ability to embed sensor nodes in a field at strategic locations could give farmers detailed soil analysis to help maximize crop yield or possibly alert them when soil and crop conditions attain a predefined threshold. Since wireless sensor networks are designed to run unattended, active physical monitoring is not required.

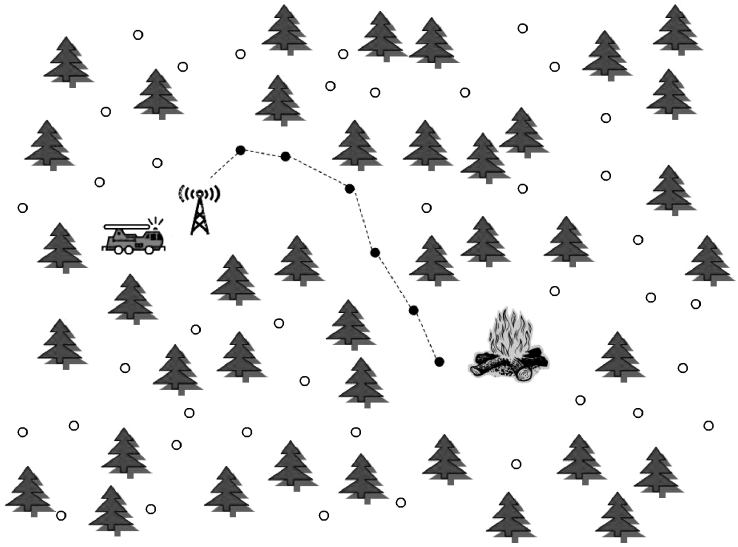

Figure 1.4 Forest-fire monitoring application.

Disaster relief efforts such as the ALERT flood-detection system [17] make use of remote field sensors to relay information to a central computer system in real time. Typically, an ALERT installation comprises several types of sensors, such as rainfall sensors, water-level sensors, and other weather sensors. Data from each set of sensors are gathered and relayed to a central base station.

1.3.3 Health Applications

Potential health applications abound for wireless sensor networks. Conceivably, hospital patients could be equipped with wireless sensor nodes that monitor the patients' vital signs and track their location. Patients could move about more freely while still being under constant supervision. In case of an accident—say, the patient trips and falls—the sensor could alert hospital workers as to the patient's location and condition. A doctor in close proximity, also equipped with a wireless sensor, could be automatically dispatched to respond to the emergency.

Glucose-level monitoring is a potential application suitable for wireless sensor networks [18]. Individuals with diabetes require constant monitoring of blood sugar levels to lead healthy, productive lives. Embedding a glucose meter within a patient with diabetes could allow the patient to monitor trends in blood-sugar levels and also alert the patient whenever a sharp change in blood-sugar levels is detected. Information could be relayed wirelessly from the monitor to a wristwatch display. It would then be possible to take corrective measures to normalize blood-sugar levels in a timely manner before they get to critical levels. This is of particular importance when the individual is asleep and may not be aware that their blood-sugar levels are abnormal.

The Smart Sensors and Integrated Microsystems (SSIM) project at Wayne State University and the Kresge Eye Institute are working on developing an artificial retina [18]. One of the project goals is to build a chronically implanted artificial retina that allows a visually impaired individual to "see" at an acceptable level. Currently, smart sensor chips equipped with 100 microsensors exist that are used in ex vivo retina testing. The smart sensor comprises an integrated circuit (with transmit and receive capabilities) and an array of sensors. Challenges in this application include establishing a communication link between the retinal implant and an external computer to determine if the image is correctly seen. Regulating the amount of power used by the system to avoid damage to the retina and surrounding tissue is also a primary concern.

1.3.4 Habitat Monitoring on Great Duck Island

Leach's Storm Petrel (Fig. 1.5) is a common elusive seabird in the western North Atlantic. Most of their lives are spent off-shore, only to return to land during the breeding season. During this time, they nest in burrows located in soft, peaty soil, and are active predominantly at night. It is believed Great Duck Island, located 15 km off the coast of Maine, has one of the largest petrel breeding colonies in the eastern United States.

Figure 1.5 Leach's Storm Petrel. (U.S. Geological Survey photo by J. A. Splendelow.)

Petrel activity monitoring is a delicate problem, since disturbance or interference on the part of humans can lead to nest abandonment or increased predation on chicks or eggs.

To circumvent this problem, in the spring of 2002, the Intel Research Laboratory at Berkeley initiated a collaboration with the College of the Atlantic in Bar Harbor and the University of California at Berkeley to deploy a series of wireless sensor networks on the island [11,19,20]. By the summer of 2002, 43 sensor nodes were deployed on the island. The primary purpose of the sensor network was to monitor the microclimates in and around nesting burrows used by the petrels. Thus, researchers could take multiple measurements of biological parameters at frequent intervals, with minimal disturbance to the breeding colony. It was necessary to enter the colony only at the beginning of the study to insert sensor nodes into burrows and other areas of interest. Three major issues explored in this experiment included:

1. Determination of the usage pattern of nesting burrows over the cycle when one or both members of the breeding pair may alternate between incubation and feeding.
2. Determination of changes in the environmental conditions of burrows and surface areas throughout the course of the breeding season.
3. Measuring the differences in the microenvironments with and without large numbers of nesting petrels.

By November 2002, 32 sensor nodes had collected over one million sensor readings. For this particular application, the nodes were equipped with a separate weather board that contained sensors to detect temperature, humidity, barometric pressure, and midrange infrared. Motes periodically sampled and relayed their sensor readings to different base stations located throughout the island. These base stations provided researchers access to real-time environmental data gathered by the sensor nodes via the Internet.

In June 2003, a second-generation network comprising 56 nodes was deployed. This network was further augmented in July 2003 with an additional 49 nodes. Finally, in August 2003, over 60 additional burrow nodes and 25 weather-monitoring nodes were deployed on the island.

1.3.4.1 Hardware The system designers employed Mica motes (Fig. 1.6), which are small devices equipped with a microcontroller, low-power radio, memory, and batteries. The motes are designed with a single-channel 916-MHz radio that provides bidirectional communication at 40 kbps, an Atmel Atmega 103 microcontroller operating at 4 MHz, and 512 kB of nonvolatile storage. Power to the mote is supplied by a pair of AA batteries and a DC boost converter.

To allow sampling of the environment, the Mica mote was equipped with a Mica weather board that contains temperature, photoresistor, barometric pressure, humidity, and passive infrared sensors [11]. To protect the motes from adverse weather conditions, the sensor package was sealed in a 10-micron parylene sealant that protected the electrical contacts from water. The sensors themselves remained exposed so as not to hinder their sensitivity. The coated sensor was then encased in a ventilated acrylic enclosure. The acrylic enclosure was radio and infrared transparent and also elevated the mote off the ground.

Due to the longevity of the proposed application, battery life was budgeted carefully. A conservative estimate of 2200 mAh total capacity was utilized. For illustrative purposes, Table 1.1 lists the costs associated with performing basic Mica mote operations and Table 1.2 lists the costs associated with basic sensor operations [21].

For the habitat monitoring application, an application lifetime of 9 months was desired. Thus, with 2200 mAh of total power available, the sensor motes were

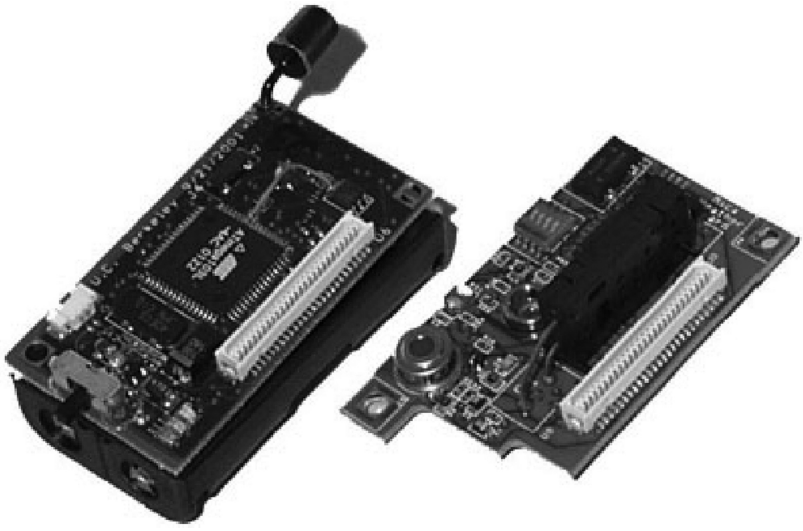

Figure 1.6 Mica sensor node (left) with the Mica Weather Board (right).

TABLE 1.1 Mica Mote Power Requirements for Different Operations

Operation	nAh
30-byte packet transmission	20.000
30-byte packet reception	8.000
1 ms radio listening	1.250
Sensor analog sample	1.080
Sensor digital sample	0.347
Reading sample from ADC	0.011
Flash read data	1.111
4-byte flash write/erase data	83.333

budgeted at 8.148 mAh of power consumption daily. However, sensor motes consume 30 μA in their sleep state [21]. This reduced the daily energy budget to 6.9 mAh available for sensing, communicating, and processing operations. The application was responsible for determining how this energy budget was to be allocated. Without any energy budgeting, a sensor mote operating at a 100% duty cycle can only operate for 7 days [21].

1.3.4.2 Architecture The wireless sensor network architecture is divided into distinct tiers (Fig. 1.7). The lowest level consists of autonomous motes, equipped with various sensors, that perform basic networking, computing, and sensing tasks. They are organized into a local one-hop network and collectively identified as a *sensor patch*. One of the sensor motes within the sensor patch serves as a gateway between the sensor patch and the base station. It differs from other motes in that it is equipped with a high-gain antenna able to transmit data over a 350-foot link to the base station. The gateway node is also equipped with a solar panel and rechargeable battery in order to be able to operate with a 100% duty cycle. Data relayed to the base station are stored in a database and made available over the Internet.

TABLE 1.2 Individual Sensor Characteristics

Sensor	Accuracy	Changeability	Max Rate (Hz)	Start-Up Time (ms)	Current (mA)
Photoresistor	N/A	10%	2000	10	1.235
I2C temperature	1 K	0.20 K	2	500	0.150
Barometric pressure	1.5 mbar	0.5%	28	35	0.010
Barometric pressure temperature	0.8 K	0.24 K	28	35	0.010
Humidity	2%	3%	500	500–30,000	0.775
Thermopile	3 K	5%	2000	200	0.170
Thermistor	5 K	10%	2000	10	0.126

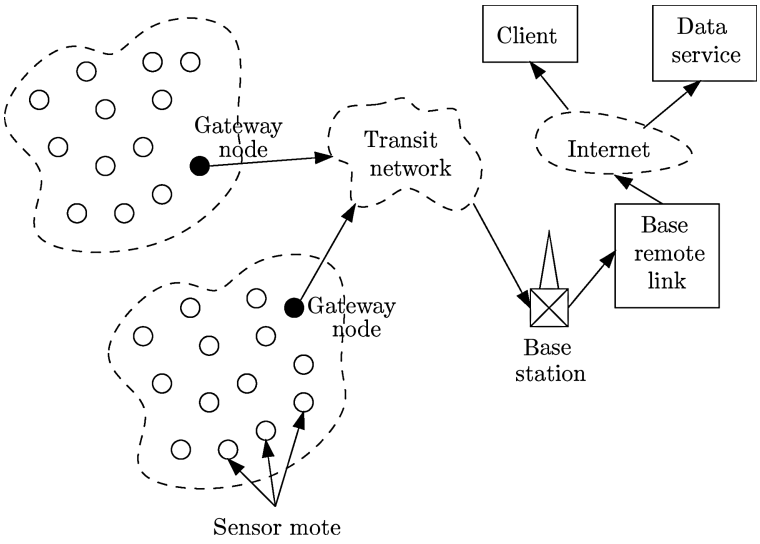

Figure 1.7 System architecture for habitat monitoring.

These collected data are also relayed, via satellite transceiver, to an off-site research facility located in Berkeley, California.

Periodically, motes took readings from each of their sensors. The data were time-stamped and kept in flash memory. Readings were then transmitted in a single 36-byte data packet. After successful transmission, motes entered their lowest power state for the next 70 seconds. The duty cycle was an expected 1.7% for the application. Each sensor mote was powered by two AA batteries with an estimated 2200 mAh capacity.

Several key application requirements identified by the system designers included Internet access, organization of the network as a hierarchy, sensor network longevity, the ability to operate off the grid, remote sensor network management, inconspicuous operation, in situ interaction, sensors and sampling, and data archiving capabilities.

1.3.4.3 Results Since this is one of the first long-term deployments of the Mica mote platform, it was interesting to see how the wireless sensor network performed. Ironically, although the readings collected by the wireless sensor network proved to be unusable to researchers for making scientific conclusions, the fidelity of the acquired sensor readings gave insight into overall network behavior.

Over 1.1 million readings were collected in a time span of 123 days. During this period, abnormal operation was detected among the sensor node population. Typical problems included nodes generating sensor readings that were outside their predefined range, unreliable and erratic packet delivery, and system node failure.

1.3.5 PODS Project

Rare and endangered species of plants are threatened because they grow in limited select locations. Evidently, these locations have special properties that sustain and support their growth. The PODS project [12,13,22], located at Hawaii Volcanoes National Park, consists of a wireless sensor network deployed to perform long-term studies of these rare and endangered species of plants and their environment.

In Hawaii, the weather gradients are very sharp. In fact, regions of the island exist where rain forests and deserts are located less than 10 miles apart. Thus, it is not surprising that endangered species of plants are restricted to very small areas. Unfortunately, weather stations located throughout the island provide insufficient information for the areas where these endangered plants exist. Consequently, deploying a very dense wireless sensor network in the area of interest allows fine-grained temperature, humidity, rainfall, wind, and solar radiation information to be obtained by researchers.

In this particular wireless sensor network application, two types of data are collected: *weather data*, which are collected every 10 minutes, and *high-resolution images*, which are collected every hour. The data repository is a central server located on a different island than where observations are made. Weather measurements are maintained in a database and the high-resolution images are stored as individual files.

Exception reporting is the type of monitoring of interest to the biological problem studied on the island. Baseline information is developed that describes the expected environmental conditions on the island. This baseline information is reported, including periods during which the environment properly reflects it. The other information gathered are the time periods and degree of variance from the baseline model. These are the periods of most interest, because those intervals are when significant changes to the organisms under observation are likely to occur. Data summarization techniques are employed for these categorized data.

The high-resolution images collected every hour have a resolution of 1600×1200 pixels and serve several important interpretive functions. Images permit casual observations during periods where environmental conditions are reported as normal. During exceptional periods, when the environmental conditions deviate from the norm, images provide an important visual check on the conditions and permit a quick analysis of how the various types of vegetation under observation are responding. Most images are taken close to the endangered plant species. This permits observations of flowering, fruit set, fruit disappearance, leaf flushes, leaf loss, and other significant events. Since the images are stored as individual files, it is a simple matter to review them to confirm observations or review periods that were not being monitored. The data measurements collected are generally unfeasible to obtain via conventional monitoring techniques.

The type of deployable equipment allowed for research in the national park is limited. As a minimum criterion, the equipment cannot pose a threat to any species. Furthermore, it must not interfere or be a distraction to visitors. This is of particular concern since some areas of the island are visited by a large number of tourists.

In some parts of the island, little can be done to hide the instrumentation. Therefore, rocks were chosen as containers that camouflage and house the computer, sensing instruments, and batteries. The availability of small trees along other parts of the island expands the options for concealing sensor nodes. In some cases, short hollow structures, designed to look like branches, were also used to house the sensing equipment.

Upon initial deployment, the wireless sensor network engages in a neighborhood discovery process. This gives each node information about which sensor nodes it can communicate with directly. Next, the sensor network executes a routing protocol so that senders are able to send messages to their desired destination. For this particular application, requirements determine the functionality expected of the underlying routing protocol. Since nodes both send and receive messages, the protocol must provide nodes with routing information so that nodes can send messages specifically to other nodes. Adaptability to changing network topologies is required, as sensor nodes may be added, moved, or become depleted. Finally, the routing protocol needs to be designed such that network connectivity is maintained even when nodes are powered down to conserve battery life. As a result, *Geometric Routing Protocol* [13] and *Multi-path On-Demand Routing Protocol* [13] were developed for this particular application.

1.4 SERVICES

Most large-scale wireless sensor network applications share common characteristics. Services such as time synchronization, location discovery, data aggregation, data storage, topology management, and message routing are employed by these applications. Each is briefly described in this section.

1.4.1 Time Synchronization

Time synchronization is an essential service in wireless sensor networks [23]. In order to properly coordinate their operations to achieve complex sensing tasks, sensor nodes must be synchronized. A globally synchronized clock allows sensor nodes to correctly time-stamp detected events. The proper chronology, duration, and time span between these events can then be determined. Incorrect time stamps, due to factors such as hardware clock drift, can cause the reported events relayed back to the base station to be assembled in incorrect chronological order.

Time synchronization is crucial for efficient maintenance of low-duty power cycles. Sensor nodes can conserve battery life by powering down. When properly synchronized, nodes are able to turn themselves on simultaneously. When powered up, sensor nodes can relay messages to the base station and subsequently power down again to conserve energy. Unsynchronized nodes result in increased delays while they wait for neighboring nodes to turn their radios on, and in the worst case, messages transmitted can be lost altogether.

1.4.1.1 Design Challenges

Several common challenges exist for the design of time synchronization protocols [23]. In order to perform synchronization, nodes exchange messages with each other. However, factors in the network can cause delays in message delivery. Four sources of error in network time synchronization can be identified. The first factor is *send time*, which includes the amount of time required to construct and transmit a message from the sender. The second factor is *access time*, which includes the delay experienced at the MAC layer, such as waiting for the channel to become idle. The third factor is *propagation time*, which includes the amount of time spent relaying the message across the various network interfaces between the sender and the receiver. Finally, the fourth source of delay is *receive time*, which includes the amount of time required by the receiver to accept and decode the message and transfer it to the host.

1.4.1.2 Design Metrics

A broad set of design metrics for time synchronization protocols exist [23]. Factors such as energy efficiency, scalability, precision, robustness, lifetime, and scope must all be taken into consideration. As with all protocols designed for wireless sensor networks, energy efficiency is a chief concern. Protocols must be scalable, since sensor networks can potentially contain a very large number of sensor nodes. The precision required may vary depending on the type of sensor network application. For example, in some cases, an ordering of detected events may be required so that a chronology of events can be assembled. In other cases, it may be necessary to time-stamp events at finer resolution. For example, real-time applications, such as target tracking, may require tight synchronization between sensor nodes as they follow the object's movements. Finally, since sensor networks are generally left unattended for long periods of time, time synchronization protocols must be fault-tolerant and adaptive to changing network topologies. For example, as new nodes are introduced and other nodes die, sensor nodes must be able to synchronize themselves seamlessly with their neighbors.

1.4.1.3 Protocols

Much work has gone into solving the problem of time synchronization among sensor nodes. At a rudimentary level, where a simple causality relationship [24,25] between detected events is desired, even traditional approaches employed in other types of distributed systems, such as vector clocks [26,27], are generally not practical for wireless sensor networks.

Vector clocks are not scalable in resource-constrained sensor networks with an unknown or large number of nodes. The additional overhead required to transmit vector time stamps with each message would quickly deplete a node's battery, rendering it useless. Furthermore, vector clocks are abstract in nature and do not indicate the duration of an event in physical time measurements, such as minutes or seconds. Other complex protocols, such as the network time protocol (NTP) [28], are unsuitable for wireless sensor networks because of their computational requirements.

Protocols such as TSync [29] and reference-broadcast system (RBS) [30] exploit the broadcast nature of wireless sensor networks in order to achieve global time synchronization with a high degree of accuracy.

In refs. [31] and [30], Elson et al. propose RBS, a time synchronization technique that uses a *third party* to perform synchronization among nodes. Individual nodes send reference beacons to their neighbors. The beacon's time of arrival is used by receiving nodes as a reference point for comparing local clocks. Since a reference broadcast arrives at all receivers at essentially the same time, propagation error is minimal. In the simplest form of RBS, a node broadcasts a single pulse to two receivers. Upon receiving the reference broadcast, the receivers exchange their receiving times and attempt to estimate their relative phase offsets. Through simulation, it has been shown that 30 reference broadcasts improves the precision from 11 μs to 1.6 μs when synchronizing a pair of nodes.

In ref. [32], the authors propose a networkwide time synchronization protocol called Timing-Sync Protocol for Sensor Networks (TPSN). The protocol has two phases: *level discovery* and *synchronization*. The level-discovery phase is initiated when the sensor network is deployed. A node is elected as the root node (level 0) and initiates the level-discovery phase by transmitting a level-discovery message, which contains the node ID and level of the sender. Upon receiving this message, a node assigns itself a level that is one level higher than the incoming level-discovery message. Subsequent level-discovery messages received are discarded. This broadcast phase continues until all nodes are assigned a level. The synchronization phase of the algorithm involves a two-way message exchange between a pair of nodes. The authors assume that clock drift and propagation delay (in both directions) between a pair of nodes is constant in the period of time between a single message exchange.

A node initiates synchronization by sending a pulse message that includes the node's level and local time. A node that receives the pulse message responds with an acknowledgment that includes the original time stamp received, the relative clock drift between both nodes, and the propagation delay. The node that initiated the pulse calculates the *actual* ensuing clock drift and propagation delay, and synchronizes itself with the receiving node. The synchronization phase is initiated by the root node. Nodes at the level below the root node exchange messages with the root node and adjust their clocks accordingly. Other nodes at lower levels, upon overhearing that nodes at levels above them are performing time synchronization, also initiate time synchronization. The authors report that their time synchronization protocol is precise within 6.5 μs when implemented on Compaq IPAQs running the Linux operating system. On Mica motes, they report their time synchronization protocol achieves an accuracy of 29.13 μs.

In ref. [33] the authors describe two lightweight synchronization algorithms called Tiny-Sync and Mini-Sync. Both techniques employ the conventional two-way messaging scheme to determine the relative clock drift and offset between the clocks of two sensor nodes.

In ref. [34], the authors describe lightweight tree-based synchronization (LTS), which attempts to minimize the underlying complexity of the time synchronization process, rather than attempting to maximize accuracy. Two approaches are presented in LTS. Both of them require sensor nodes to synchronize their clocks to a reference point. The first approach given is a centralized algorithm that uses the

edges of the spanning broadcast to perform pairwise synchronization. The root of the spanning tree is responsible for initiating synchronization. Under the assumption that clock drift is bounded and given the required degree of precision, the reference node calculates the time period a synchronization step is valid.

The second approach presented by the authors is completely distributed. Individual sensor nodes request synchronization with other nodes as needed. When a node decides it is necessary to synchronize its clock with another node, it sends a synchronization request to the closest reference node. As a result, all nodes along the path from the reference node and the node requesting synchronization must have their clocks synchronized for the requesting node to synchronize its local clock properly.

1.4.2 Location Discovery

Location discovery involves sensor nodes deriving their positional information, expressed as global coordinates or within an application-defined local coordinate system. The importance of location discovery is widely recognized [35–40]. It serves as a fundamental basis for additional wireless sensor network services where location awareness is required, such as message routing. Furthermore, in applications such as fire detection, it is generally not sufficient to determine *if* a fire is present, but more importantly, *where*. A brief review of three proposed solutions to location discovery are presented.

1.4.2.1 Multilateration by Distance Measurements Meguerdichian et al. [35] describe a localized algorithm that uses multilateration for solving the problem of location discovery. A node determines its location based on its distance from neighboring nodes that serve as beacons. Beacons are nodes that are location-aware and broadcast their location information periodically. They acquire their location from multilateration procedures or other sources such as GPS. Distances between neighboring nodes are estimated using received signal strength indication (RSSI) or ultrasound techniques. Thus, a node requires only local neighbor information to determine its position.

1.4.2.2 Ad Hoc Positioning System Niculescu and Nath [38] propose their ad hoc positioning system (APS), whereby nodes determine their location in reference to *landmarks* that are location aware. Landmarks can be other sensor nodes, base stations, or beacons that have positional information. Unlike GPS, where direct line of sight is required with a series of satellites in order to triangulate a location, landmark information is propagated through the wireless sensor network in a multihop fashion.

When an arbitrary node in the wireless sensor network has distance estimates to three or more landmarks, it computes its own position in the plane. The node utilizes the centroid of the landmarks as its location estimate. Nodes in direct communication with a landmark infer their distance from it based on the received signal strength of the landmark.

Through message propagation, nodes two hops away from a landmark estimate their distance based on the distance estimates of nodes located next to the landmark. The propagation schemes proposed by the authors eventually flood the entire network until all nodes are able to determine their coordinates.

1.4.2.3 APS using Angle of Arrival

In ref. [39], Niculescu and Nath present two algorithms, *DV-Bearing* and *DV-Radial*, that allow sensor nodes to get a bearing and a radial in relation to a landmark using angle of arrival (AoA) to derive position information. The term "bearing" refers to an angle measurement with respect to another object. A "radial" refers to a reverse bearing which is simply the angle at which an object is seen from another location. The term "heading" refers to the sensor node's bearing with respect to true north and represents its absolute orientation.

AoA sensing requires sensor nodes to be equipped with an antenna array or several ultrasound receivers. This equipment is currently available in small package formats for wireless sensor network nodes such as the one developed for the Cricket Compass Project [41,42]. The theory of operation is based on time difference of arrival (TDoA) and phase difference of arrival. If a node sends an RF signal and an ultrasound signal at about the same time, the receiving node can infer the distance between the sender and itself by measuring the time difference between the arrival of the RF signal and the ultrasound signal. To derive the angle of arrival of the signal, the receiving sensor node uses two ultrasound receivers placed at a known distance from each other.

1.4.3 Data Aggregation

Data aggregation and query dissemination are important issues in wireless sensor networks [43]. Sensor nodes are typically energy constrained. Therefore, it is desirable to minimize the number of messages relayed, because radio transmissions can quickly consume battery power. A naive approach to reporting sensed phenomena is one where all (raw) sensor readings are relayed to a base station for off-line analysis and processing. However, since sensor nodes within the same vicinity often detect the same, common phenomena, it is likely some redundancy in sensor readings will occur [44]. Local collaboration allows nearby sensor nodes to filter and process sensor readings before transmitting them to a base station. Consequently, this process can reduce the number of messages relayed to the base station.

Figure 1.8 represents an animal-tracking application where several sensor nodes are randomly deployed in a forest. When an animal, represented by the solid square, passes through the area being monitored, individual sensor nodes detect the presence of the animal and relay their findings, in a multihop fashion, to the base station located some distance away. In sufficiently dense sensor networks, overlapping areas of coverage are possible. Thus, the animal may be detected by several sensors.

In the scenario presented in Figure 1.8, nodes A, B, C, D, and E sense the presence of a nearby animal. Nodes $B-E$ each send a message to node A with their observed sensor data. Node A forwards the received messages, along with its own set of sensor readings, to the next node along the path to the base station. Thus, node A sends a total

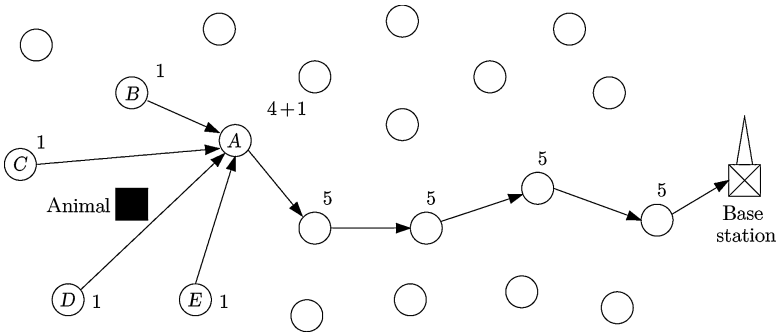

Figure 1.8 Event detection and reporting without data aggregation.

of 5 messages, which are all subsequently relayed from node to node, until they reach the base station. In total, 29 messages are transmitted throughout the network.

A reduction in communication and energy costs is possible if collected sensor data is aggregated prior to relaying. Figure 1.9 is similar to Figure 1.8, except that node A collects sensor readings from nodes $B-E$ and itself, applies an aggregation function ϕ, and then relays the aggregated data. Results are compressed into a single message, which is subsequently transmitted, in a multihop fashion, for further analysis by the base station.

Various types of data aggregation are possible, depending on the level of refinement desired. In-network processing can be designed to perform one or more of the following operations:

- *Aggregate the data into a single binary value.* A Boolean (i.e., true or false) value would be sufficient to indicate if an animal was detected or not.
- *Aggregate the data readings into an area.* Coordinates of a bounding box can be given that defines the area where the sensor readings are observed. Nodes,

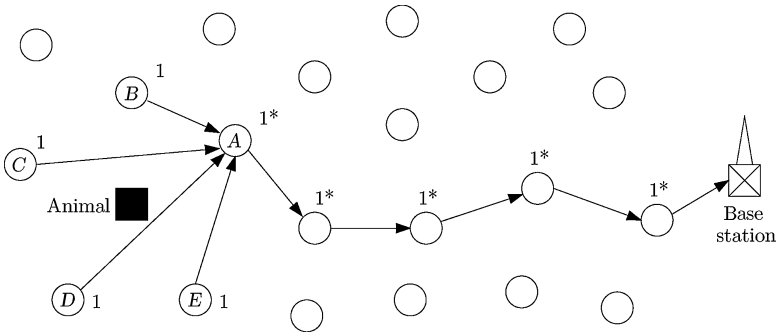

Figure 1.9 Event detection and reporting with data aggregation.

upon receiving this area information, dynamically adjust the size of the bounding box to accommodate their sensor readings before retransmitting.

- *Aggregate the collected data by applying an application-specific aggregation or filtering function*. As an example, the average, maximum, minimum, or sum of sensor values could be calculated en route prior to forwarding any received information.

Energy conservation, as a result of data aggregation, is of particular concern for sensor nodes close to the base station. Without any form of data aggregation, a greater number of messages are transmitted. As a result, their batteries are depleted quickly. Eventually, when nodes that communicate directly with the base station die, the sensor network is rendered unusable, regardless of the remaining power of other nodes (see Fig. 1.10), since no messages can reach the base station.

Data aggregation seeks to combine data arriving from different sources en route. In [44], the authors study the energy savings and latency trade-offs caused by data aggregation and how factors such as source (i.e., event) and sink (i.e., base station) placements and network density affect this trade-off. A complexity analysis of optimal data aggregation in sensor networks is also performed, and although it is shown that optimal data aggregation is NP-hard, polynomial-time solutions exist for certain cases.

The work presented in [45] continuously computes aggregates of wireless sensor network monitoring functions. Aggregates computed include sums, averages, and counts. Network properties considered include loss rates, energy levels, and packet counts. A novel tree construction algorithm is proposed to enable energy-efficient computation of some classes of aggregates, and it is demonstrated, through actual implementation and experiments, that wireless communication artifacts and packet loss significantly impact the computation of these aggregate properties. During experiments conducted on a test bed of 26 sensor nodes, packet loss for each link was measured every minute for two hours under various topology settings.

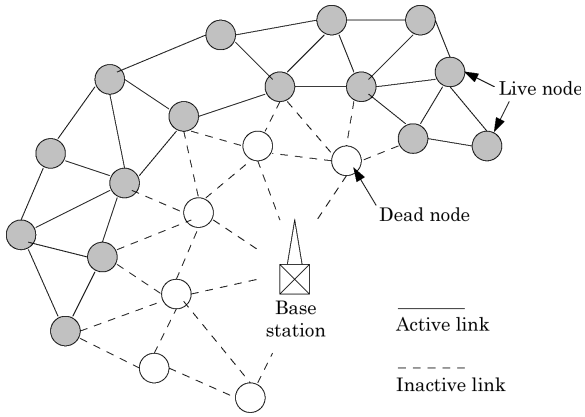

Figure 1.10 Event detection and reporting with data aggregation.

Although the majority of links were good, 10% of the nodes exhibited a packet rate loss greater than 50%. As a result, the value of the COUNT aggregate, which reports the total number of active sensor nodes in the network, fluctuated greatly over time.

The infrastructure presented for wireless sensor network monitoring consists of three classes of software [45]:

1. The first component consists of a tool such as *dump* that is used to collect detailed information about the system state. This is used to provide debugging information about the sensor nodes and also report any logged information kept by the nodes over a period of time.
2. The second category of tool is referred to as *scans*. These constitute a global, albeit aggregated view, of the wireless sensor network and report metrics such as overall resource consumption. An example of a scan is an *escan* whereby a special user-gateway node initiates state information collection from the entire system. However, instead of all nodes relaying their power level information, data collected is aggregated en route in order to minimize the amount of information propagated throughout the network.
3. The final category of tool is referred to as *digests*, which are simply aggregates of some network property. Digests span the entire network, but unlike scans, they are computed continuously. Computed information is propagated throughout the network by piggybacking digests onto regular messages transmitted throughout the sensor network. Clearly, the energy savings achieved is offset by the increased latency.

The second contribution entails the design of protocols to enable computation of network digests. Values such as node energy level, degree of connectivity, and volume of traffic, are considered. Decomposable functions (i.e., functions that can be expressed in terms of another function) such as *sum*, *min*, *max*, *average*, and *count* are applied to these analyzed values. Digest computation is accomplished using *digest diffusion*, which implicitly builds a broadcast tree where computed partial results of decomposable functions are propagated toward the root.

For example, assume a connected homogeneous wireless sensor network exists where all nodes are equipped with thermal sensors that record the ambient temperature. Initially, every node assumes it has observed the highest temperature reading and exchanges this information with its immediate neighbors. A node, upon receiving a temperature measurement from a neighboring node, adjusts the source of the highest reading, if necessary, and propagates this information throughout the broadcast tree. Eventually, all nodes converge on the same maximal temperature reading.

Broadcast tree maintenance is required as nodes fail over the lifetime of the wireless sensor network. Thus, a node periodically broadcasts messages to maintain the digest. Nodes use a time-out value to determine if a neighboring node is no longer transmitting messages to it. Thus, a node may switch to a different parent node when it is no longer receiving messages from its existing parent node.

1.4.4 Data Storage

Data storage presents a unique challenge to developers. Event information collected by individual nodes must be stored at some location, either in situ or externally. In some cases, where an off-line storage area is not available, data must be stored within the wireless sensor network. Ratnasamy et al. [46,47] describe three data-storage paradigms employable in wireless sensor networks:

1. *External Storage*. In this model, when a node detects an event, the corresponding data are relayed to some external storage located outside the network, such as a base station. The advantage of this approach is that queries posed to the network incur no energy expenditure since all data are already stored off-line.
2. *Local Storage*. In this model, when a node detects an event, event information is stored locally at the node. The advantage of this approach is that no initial communication costs are incurred. Queries posed to the wireless sensor network are flooded to all nodes. The nodes with the desired information relay their data back to the base station for further processing.
3. *Data-Centric Storage*. In this model, event information is routed to a predefined location, specified by a geographic hash function (GHT), within the wireless sensor network. Queries are directed to the node that contains the relevant information, which relays the reply to the base station for further processing.

For wireless sensor network applications that are envisioned to be long-lived, even optimized communication schedules can deplete a node's battery within a relatively short period of time (i.e., a couple of months). Consider an environmental application, such as microclimate monitoring, where individual sensor nodes periodically sample their local environment to measure temperature, light, precipitation, pressure, and humidity levels. Over time, the amount of data generated by the sensor network can be substantial. This is particularly true if individual sensor nodes take samples at short regular intervals, such as every 30 minutes.

Ganesan et al. [48] look to provide a distributed, progressively degrading storage model. This is achieved by constructing local, multiresolution summaries of observed sensor data stored hierarchically throughout the wireless sensor network. Queries on summary information are performed in a drill-down fashion: coarse, highly compressed data are stored in nodes at the highest levels in the hierarchy. As more detailed information is required, nodes at lower levels in the hierarchy, with more detailed event information, are queried.

Summary information is created by employing a wavelet-based compression technique, which offers the following advantages:

- A compact representation of data is produced that highlights interesting features in the accumulated data, such as long-term trends, edges, and significant anomalies.

- Spatiotemporal queries can be satisfied with little communication overhead by employing drill-down querying. Basic information from the wireless sensor network is gathered from nodes at the highest level in the hierarchy. As more detailed spatiotemporal information is required, nodes further down the hierarchy are queried for the relevant data.
- Aging, and subsequently discarding summaries selectively, gracefully degrades query performance over time. Since wireless sensor networks are typically resource-constrained, nodes discard older data in favor of newly gathered sensor readings.

1.4.5 Topology Management and Message Routing

Wireless sensor networks can possibly contain hundreds or thousands of nodes. Routing protocols must be designed to achieve an acceptable degree of fault tolerance in the presence of sensor node failures, while minimizing energy consumption. Furthermore, since channel bandwidth is limited, routing protocols should be designed to allow for local collaboration to reduce bandwidth requirements.

Observations made in ref. [49] show that, although intuitively it appears a denser deployment of sensor nodes renders a more effective wireless sensor network, if the topology is not carefully managed, this can lead to a greater number of collisions and potentially congest the network. As a result, there is an increased amount of latency when reporting results and a reduction in the overall energy efficiency of the network. Furthermore, as the number of reported data measurements increases, the accuracy requirements of the application may be surpassed. This increase in the reporting rate by the deployed sensor nodes can actually harm the wireless sensor network performance, rather than prove beneficial.

Message-routing algorithms in ad hoc networks can be separated into two broad categories: greedy algorithms and flooding algorithms [50]. Greedy algorithms apply a greedy path-finding heuristic that may not guarantee a message reaches its intended receiver. One example of greedy routing, proposed by Finn in 1987, is forwarding to a neighbor that is closest to the destination. Additional steps are required to ensure the message is received by its intended recipient. Flooding algorithms employ a controlled packet duplication mechanism to ensure every node receives at least one copy of the message. For these algorithms to terminate, nodes in the sensor network must remember which messages have been previously received.

In ref. [50], the authors present two distributed routing protocols, *face routing* and *greedy-face-greedy* (GFG). Both algorithms guarantee packet delivery as long as the wireless sensor network remains connected and static while the message is relayed from sender to receiver. The medium access is ideal since it guarantees message transmission between two neighbors in a finite time. The communication graph is the unit graph where two nodes can communicate if and only if the distance between them is at most R, where R is the transmission radius of all nodes.

Both algorithms require messages to carry some overhead information. However, sensor nodes themselves do not need to maintain additional routing information.

The algorithms first construct a connected planar subgraph, called a Gabriel graph, of the underlying wireless sensor network in a distributed fashion. Edge e is in the Gabriel graph if and only if the circle with edge e as the diameter contains no other nodes inside it. The Gabriel graph partitions the graph into faces that are bound by polygons and make up the edges of the graph.

In the face-routing algorithm [50], the boundary of the face is traversed in a counterclockwise fashion until an edge is found that intersects with the line that connects the source and destination. The algorithm then continues to scan the next adjoining face in a similar manner. The entire process iterates until the destination is reached.

In the GFG algorithm [50], greedy routing (i.e., forwarding to the neighbor node closest to the destination) is applied as long as the node currently holding the packet has a neighbor closer to the destination node than itself. When current node A does not have such a neighbor, face routing is applied until a node B, closer to the destination node than node A, is encountered. Node B then reverts back to greedy forwarding. This reversal of modes can be repeated until the packet is delivered to its intended destination. Greedy perimeter stateless routing (GPSR) [51] is a routing protocol similar to GFG [50] that incorporates medium-access-layer and mobility considerations.

Greedy routing algorithms have been found to work well in wireless sensor networks due to their efficiency and scalability [52]. Greedy forwarding techniques offer several advantages over naive routing techniques (i.e., flooding):

- Nodes need to maintain only local topology information. This makes the protocol highly scalable, since routing information to all destinations is not maintained locally. Such a routing table would quickly grow in size, consuming the node's limited memory.
- The protocol is adaptable to frequent topology changes, since the routing path can be dynamically adjusted based on the current one-hop neighborhood of a node.
- Since only local information is used, nodes need not be aware of the topology of the entire wireless sensor network.

Network self-organization can be extended further than simple topology management. Assigning *roles* to sensor nodes based on their physical connectivity and sensing capabilities is proposed in [53]. Metrics, such as sensing proximity value, cumulative sensing degree, and other intermediate sensing parameters, allow the wireless sensor network to be partitioned into distinct *sensing zones*. Sensing zones are a collection of sensor nodes with a common sensing objective and a specific sensing quality of service (sQoS). Coordinators are elected to act as leaders within a sensing zone and are responsible for coordinating sensing-zone members and performing network reorganization maintenance. This approach is an improvement over other types of topology management schemes, such as hierarchical topologies, since they may be too rigid for a particular wireless sensor network application.

1.5 WIRELESS SENSOR AND ACTOR NETWORKS

Wireless sensor and actor networks (WSANs) [54] can be considered as an extension of traditional wireless sensor networks. They consist of two major components: *sensor nodes* and *actor nodes*. Sensor nodes are low-cost, low-power devices with limited sensing, computational, and communication capabilities. Actor nodes are resource-rich nodes equipped with more powerful processors, longer-range radio transceivers, and longer-lasting, or possibly renewable, power sources. They may also be able to navigate throughout the area covered by the sensor nodes. The number of sensor nodes generally outnumbers the number of actor nodes by a sizable quantity.

There are several defining characteristics of WSANs. These include:

- *Real-Time Requirements*. Depending on the application, it may be necessary for nodes within the sensor network to respond quickly to detected events. For example, in an environmental monitoring application, if a fire is detected, some sort of corrective action should be initiated as quickly as possible. The data collected by the wireless sensor and actor network must be timely and current when the corrective action is taken.
- *Coordination*. In a wireless sensor network, the process of data collection is coordinated by a central entity, such as a base station. In a wireless sensor and actor network, sensor–sensor coordination, actor–sensor coordination, and actor–actor coordination are required. Sensor nodes report detected events to actor nodes, which in turn, take some appropriate action. This may include coordinating response activities with other actor nodes, providing additional instructions to nearby sensor nodes, or processing sensed event information to relay back to a central base station.

The roles of sensor nodes and actor nodes are to collect data from the environment and react appropriately to sensed events. The *sensor–actor field* defines the area where sensor nodes and actor nodes are distributed. A central base station, sometimes referred to as a *sink*, monitors and coordinates overall network activity.

When a sensor node observes a particular phenomenon, it transmits its findings to a nearby actor node. The actor node processes all incoming data and initiates an appropriate response or processes and relays the information to the sink. The sink can then further process the received information and subsequently issue additional commands to the actor nodes to gather more information, or react to, the detected event.

1.5.1 Architecture

There are two possible types of architectures possible in WSANs:

1. *Semiautomated Architecture*. This architecture bears similarities to the architecture in most wireless sensor networks. A central base station is used

to coordinate the efforts of the actor node and sensor nodes. Queries are issued to the network and results are relayed to the base station for further processing.

2. *Automated Architecture*. This architecture does not require a central base station to coordinate efforts. Actors are programmed to work autonomously and respond to detected events appropriately. This architecture has a few advantages over the semiautomated architecture: it exhibits a lower latency, since sensed information is only relayed to actor nodes; and it has a longer overall network lifetime, since event information is only relayed to the actor node within one hop of the sensor nodes that detected the phenomenon.

Aside from communication between actor nodes and sensor nodes, communication between actor nodes must be coordinated as well in order to achieve the application objectives. Actor nodes, being resource-rich nodes with high transmission power, can transmit information over long distances, unlike sensor nodes. Furthermore, since the number of actor nodes in a wireless sensor and actor network is typically small, communication among actor nodes is analogous to an ad hoc sensor network.

The most crucial aspect of sensor–actor communication is low communication delay due to the proximity between sensor nodes and actor nodes. Other issues to consider include:

- *What are the communication requirements between actor nodes and sensor nodes*? These requirements include factors such as ensuring communication between actor nodes and sensor nodes consume minimal energy, the latency in reporting sensed event information to the actor node(s), and ensuring a proper ordering of event information.
- *Which sensors transmit to which actors*? If an event is detected by multiple sensor nodes, the sensor node may decided to relay information to a single actor node, or perhaps, to a series of actor nodes. Both approaches have their advantages and disadvantages. For example, information sent to a single actor node consumes less overall energy since fewer messages are relayed throughout the wireless sensor and actor network. However, relaying sensed event information to multiple actor nodes provides an increased level of redundancy. This may be a necessity if the network is deployed in a hostile environment where nodes are prone to failure.
- *What is the arrival time of messages*? Consider a hypothetical security application whereby actor nodes are deployed to monitor and patrol an art gallery. If an intruder is detected, one objective of the actor nodes may be to surround and immobilize the intruder. This requires that actor nodes receive notification from sensor nodes that detect the intruder in a timely (i.e., relatively simultaneously) fashion in order to coordinate their movements.

As a consequence, the set of communication protocols for wireless sensor and actor networks should provide real-time services within a specified upper bound

for delay, relay messages in an energy-efficient manner among sensor nodes and actor nodes, ensure the proper ordering of events, provide synchronization between sensor nodes reporting an activity to multiple actor nodes, and allow messages to be routed to arbitrary actor nodes.

Depending on the quality of service requirements of the wireless sensor and actor network application, coverage of a sensed event is partitioned into four cases:

1. A minimal set of actor nodes cover the event region
2. A minimum set of sensor nodes cover the event region
3. A minimum set of actor nodes and sensor nodes cover the event region
4. The entire set of actor nodes and sensor nodes in the event region monitor the phenomenon

The first three cases are aimed at reducing the level of redundancy, while the last case aims to provide maximal coverage of a detected event. There are trade-offs with both approaches. The amount of energy consumed in the network is reduced in the first three cases, at the expense of more intense coverage. The last case affords maximal coverage of the detected phenomenon, but at the expense of higher energy consumption.

Aside from communicating with sensor nodes, actor nodes can communicate directly with each other. Communication between actors can occur under various circumstances. For example, an actor node that receives information from a nearby sensor node requires the assistance of additional actor nodes in order to complete its task. Similarly, if multiple actors receive the same event information, the actor nodes can communicate with each other to coordinate their efforts.

1.5.2 Protocol Stack

As of the time of this writing, a de facto protocol stack for wireless sensor networks or wireless sensor and actor networks did not exist [54]. Unfortunately, there is no general consensus within the wireless sensor network research community about the layer structure in wireless sensor networks. It is argued that strict layering guarantees controlled interaction among layers, whereas a cross-layer design can produce spaghetti-like code that is difficult to maintain because modifications must be propagated across all protocols [55]. Furthermore, cross-layer designs can produce unintended interactions among protocols that result in performance degradation.

Other researchers are in favor of adopting a cross-layer design to overcome potential performance problems. The authors in [55] introduce a layered architecture where protocols in different layers cooperate by sharing network-status information while still maintaining separation between various layers. Despite the potential ptifalls, several motivations for employing a cross-layering approach exist [56]:

- *Optimization can be achieved in several layers.* The optimization goals at a particular layer can be designed to work with the optimization goals of other layers above and below.

- *Optimization in one level can require cooperation from other levels to show its effects*. Consider the case where the underlying routing protocol is designed to select the shortest route possible. Although this optimization results in smaller hop distances requiring less energy to transmit message packets, the larger number of messages transmitted can result in a greater amount of contention. If the medium-access control (MAC) layer is not optimized accordingly, the routing protocol may suffer as a consequence.
- *There are possible conflicts between optimization goals in distinct layers*. Some optimization solutions at distinct layers are orthogonal in design. For example, at the network layer, it may be desirable to reduce the amount of overhead maintained at individual nodes. However, this may result in a lower quality of service at the transport layer since less information is broadcast with individual packets. Similarly, employing data-compression techniques may interfere with latency requirements imposed by the application, as the nodes must wait to accumulate and aggregate received information.
- *Some scenarios do not require support from all layers*. Consider a multihop local positioning system (LPS) based on hop-by-hop distance measurements to estimate the relative distance between an arbitrary node and an anchor node. The network layer and transport layer, used to handle the end-to-end data transmissions, are not required in this application. Consequently, these layers can be omitted.

The authors in [54] suggest the protocol stack for sensor nodes and actor nodes consist of three planes:

1. *Communication Plane*. This plane enables the exchange of information between the various nodes within the wireless sensor and actor network. It receives commands from the coordination plane and provides the appropriate link relations between various nodes. The functionality of the communication plane is contained within the constituent transport layer, routing layer, and MAC layer.
   (a) *Transport Layer*. Aside from providing the traditional reliability requirements, the transport-layer protocol is responsible for providing the real-time requirements of the WSAN. For example, if the transport protocol utilized in sensor–actor communication detects a low level of reliability, the transport protocol employed in communication can notify other actors of this situation.
   (b) *Routing Layer*. Sensor nodes that detect an event have to select which actor node(s) will receive the gathered sensor information. This poses a challenge due to the existence of several actor nodes in the network. Once a decision is made, the data are relayed to the appropriate node. The routing protocol is responsible for determining the path messages will take, performing any in-network data aggregation to reduce the

number of messages relayed throughout the network, and supporting any real-time communication requirements imposed.

(c) *MAC Layer*. To effectively transmit event information from a large number of sensor nodes to actor nodes, a MAC protocol is essential. In some applications, actor nodes may be mobile. Consequently, actor nodes may leave the transmission area of some sensor nodes. One of the functions of the MAC layer is to ensure connectivity between sensor nodes and actor nodes. Contention-based protocols are generally not suitable for real-time communication between sensor nodes and actor nodes due to the latency imposed by handshaking. Exploiting the periodic nature of sensor network traffic allows for the development of collision-free real-time scheduling algorithms. These are more suitable for wireless sensor and actor networks, since they can reduce the overall delay and provide real-time guarantees.

2. *Coordination Plane*. Data received along the communication plane is forwarded to the coordination plane, which processes the received information and decides on an appropriate action. This enables nodes to collaborate and achieve a higher-level objective. Issues such as sensor–sensor coordination are addressed. These include decisions as to which sensor nodes will relay information to the corresponding actor nodes, how routing of messages in a multihop fashion is handled, how in-network data aggregation is performed, and actor node selection.
3. *Management Plane*. This plane is responsible for monitoring and controlling node functions. This includes functions such as node power management features, node mobility management, and node fault management.

1.6 SENSOR QUERYING AND DATABASE SYSTEMS

Users of wireless sensor network applications are typically interested in continuous streams of information [17] that represent the evolving status of the area under observation as time progresses [57]. Query processing systems such as TinyDB [58], Directed Diffusion [7,8], and Cougar [59] provide users of wireless sensor network applications with a high-level interface for performing queries. This relieves the user from writing complex code to gather information from the sensor network.

Part of the ongoing research into sensor database systems includes distributed query processing [60] and storage mechanisms [48] in sensor networks. The need for scalable self-organized data retrieval and in-network processing is clear. A unified query processing/networking system involves an additional challenge to designers of wireless sensor networks. Different applications have varying requirements in terms of information transfer rates, latency, coverage, and storage. The trade-off between optimizing the network topology and performing efficient query processing is an issue that needs to be resolved.

In TinyDB [58], users specify a set of declarative queries that define the information to be gathered from the wireless sensor network. Queries indicate the type of readings to be obtained, including the subset of nodes the user is interested in, and any simple transformations to be performed over the collected data. They are specified using a language like a structured query language (SQL). A sample query could be expressed as follows:

```
SELECT AVG(temp)
FROM sensors
WHERE location in (0,0,100,100) AND light > 1000 lux
SAMPLE_PERIOD 10 seconds
```

TinyDB queries are generally specified on a PC and then distributed throughout the sensor network by a query executor. The query is disseminated and results are returned in an energy-efficient manner using a variety of in-network processing techniques and cross-layer optimizations. For example, in the preceding sample query, the query executor is responsible for determining which predicate to evaluate first in the sensor network: the *temp* predicate or *light* predicate.

Queries in TinyDB are disseminated through the entire network and collected via a routing tree. The root node of the routing tree is end point of the query, which is generally where the user that issued the query is located. Nodes within the routing tree maintain a parent–child relationship in order to properly propagate results to the root. Research into query processing techniques include the design of an acquisitional query processor for data collection in wireless sensor networks. Information such as where, when, and how often data are physically collected and delivered, can be leveraged to significantly reduce the overall power consumption in the sensor network [61].

Directed diffusion [7,8] employs a different approach to query processing. Rather than utilizing a specific query language, an application specifies a *named interest*, which is used to query the sensor network. Interests contain the query particulars, expressed through a sequence of attribute/value pairs. For example, an interest expressed as:

```
location = [(100,100), (10,200)]
temperature = [10,20]
```

would report the temperature readings from all nodes located within the specified location whose temperature is within the specified limits. The interest is initiated by a *sink* node and flooded throughout the sensor network. A node that lacks data matching an interest forwards it to its neighbor node. The decision as to which node to forward the interest to is based on the contents of the interest. The notion that cues can be embedded within the query itself is one of the core principles behind *data-centric routing*. As the interest is propagated, nodes build routing tables that are used to return matching data to the sink.

1.7 SENSOR NETWORK RELIABILITY

Several applications of wireless sensor networks exist where reliability of data delivery is critical. For example, consider a security application where sensors are required to detect and identify the presence of intruders. Given the critical nature of the application, when an intruder is detected, messages must reach the base station in a timely and reliable manner. Three unique issues must be addressed when discussing data-delivery reliability in wireless sensor networks [62]:

1. *Environmental Considerations*. Wireless sensor networks can be deployed in harsh environments. However, the limited lifetime of individual sensor nodes, low bandwidth, and the size of the sensor network must be considered.
2. *Message Considerations*. Messages relayed throughout a wireless sensor network are generally small compared to ad hoc networks. For example, a simple query that requests information from a specific region of interest might be flooded throughout the sensor network. The reduced message size affects the type of loss–recovery scheme employed in the wireless sensor network.
3. *Reliability Considerations*. Traditional notions of reliability are concerned with reception of 100% of all messages transmitted. However, in a wireless sensor network, reliability may be expressed in terms of data gathered from a particular subregion within the network, or as the fidelity of partial, aggregated results.

1.7.1 PicoRadio Network

The authors in [63] present experimental measurements of radio energy consumption and packet reliability for their prototype PicoRadio network that is composed of PicoNodes [64]. Energy consumption is categorized by the energy consumed when the radio is in different states (i.e., idle, transmitting, or receiving). Packet delivery reliability is measured from a network and link perspective.

1.7.1.1 Hardware The prototype PicoNode consists of a StrongARM SA-1100 microprocessor, a Xilinx C4929XKA field-programmable gate array (FPGA), an Ericsson PBA-313-01/2 Bluetooth radio, 4 MB of DRAM, 4 MB of flash memory, and one of two possible custom sensor boards. The first board is configured with sensors that obtain light, sound, temperature, and humidity measurements. The second possible sensor board is configured with an accelerometer and magnetometer.

1.7.1.2 Protocol Stack The protocol stack utilized by each PicoNode in the sensor network test bed includes [63]:

1. *Physical Layer*. Each PicoNode employs a 100-mW Bluetooth radio that supports 79 channels in the 2.4-GHz ISM frequency band with a maximum data

rate of 1 Mbps. The radios employ Gaussian frequency shift keying modulation with 1 MHz channel spacing.

2. *Data Link Layer*. The data-link layer consists of three major components: the transmit controller and data path (TCD), the receive controller and data path (RCD), and the medium-access control (MAC). The TCD and RCD are responsible for packet buffering, serialization, deserialization, cyclic redundancy checking, and line balancing.

   The MAC uses carrier sense multiple access (CSMA) with preamble sampling (PS) for infrequent message broadcasts. For unicast traffic, a variant of spatial time-division multiple-access (S-TDMA), referred to as *on-demand S-TDMA*, is employed. Packet headers and payloads use an 8-bit cyclic redundancy check (CRC) and a data acknowledgment retransmission scheme with time-outs to help ensure packet reliability.

3. *Network Layer*. The network layer consists of four major components: energy-aware routing (EAR) protocol, location service, neighbour list service (NLS), and queuing service.

   (a) *EAR* is a destination-initiated reactive routing protocol designed to increase the survivability of the sensor network. Routing paths are chosen in a probabilistic fashion where the probability of selecting a route is inversely proportional to the average energy cost of that particular route. This achieves an even energy depletion of the sensor network.

   (b) The *location service* is called *hop-terrain* and makes use of a combination of RSSI and hop counts from reference nodes in order to triangulate a location.

   (c) The NLS maintains a table that maps neighbor-node MAC IDs to network addresses. Each entry in the table contains a link cost metric and a status indicator. The cost metric indicates the average energy required to perform a unicast transmission along a particular link.

   (d) Finally, the *queuing service* manages the timing of events during node initialization, neighbor discovery, location discovery, and MAC ID assignment.

4. *Application Layer*. The application layer consists of a standard sensor board, an optional sensor board, and the required application drivers that provide the interface between adjacent layers. The initial target application for the PicoRadio project was indoor building monitoring. The test bed comprises of three different types of nodes. The first type is sensor nodes that obtain measurements. The second are controller nodes that issue queries to the network. Finally, anchor nodes provide a location reference by periodically broadcasting their locations to other nodes in the sensor network.

1.7.1.3 Packet Reliability Empirical data about energy consumption and packet reliability of the PicoRadio network was gathered. Three configurations with varying parameters were executed and the results were collected. The first

TABLE 1.3 PicoNode Experiment Configurations

System	Description
Baseline	CSMA and on-demand S-TDMA with $T_f = T$, $S = 20$ ms, and $N_s = 9$
Case 1	CSMA-PS with $T_p = 512$ μs and $T_s = 5$ μs On-demand S-TDMA with $T_f = 256$ ms, $S = 20$ ms, and $N_s = 9$
Case 2	CSMA-PS with $T_p = 512$ μs and $T_s = 5$ μs On-demand S-TDMA with $T_f = 90$ ms, $S = 10$ ms, and $N_s = 9$

configuration is a *baseline* configuration. The subsequent two configurations, denoted *case 1* and *case 2*, have varying parameters. The configurations are summarized in Table 1.3.

The sensor network consisted of 25 PicoNodes placed in an approximately rectangular grid. Spacing between nodes varied from 3 to 7 m, with all nodes placed at roughly the same elevation. At the beginning of each experiment, a controller node broadcasted a query requesting all sensor nodes to relay 200 temperature measurements at intervals of 5 s. T denotes the time period between samples, and N denotes the total number of samples to be taken. These parameters are specified in the query disseminated to each node.

The baseline configuration utilizes CSMA without any preamble sampling. The radio is constantly on, even when the node is not transmitting or the channel is idle. The size of the frame is denoted by T_f, the number of slots within the frame is denoted by N_s, and the size of the slot spacing is denoted by S.

Nodes transmit their data packets during their designated time slots, and data packets acquired from neighboring nodes are forwarded during the designated frame using CSMA. Both case 1 and case 2 utilize CSMA with preamble sampling. Nodes wake up every T_p seconds to sense the channel. If no preamble is detected within the time period denoted by T_s, the node goes back to sleep.

1.7.1.4 Results For the baseline configuration, the end-to-end packet loss ratio (PLR) of individual sensor nodes varied from 0 to 0.2, with an overall average PLR of 0.04 for the entire sensor network. The nodes with the best reliability were those placed closest to the controller. Nodes located farthest from the controller and along the edges of the sensor network exhibited the most packet loss. The hop count for messages to reach the destination varied from a minimum of 1 hop to a maximum of 8 hops.

For case 1, the variation in the PLR was lower, but the overall PLR for the network remained the same. This is because for a given slot spacing, the preamble sampling had a negligible impact on end-to-end packet reliability. In case 2, the PLR ranged from 0 to 0.88, with an overall network average of 0.36. The higher PLR was caused by more packet collisions due to the smaller frame size and slot spacing.

1.8 SENSOR OPERATING SYSTEMS

TinyOS is an open-source operating system designed for wireless embedded sensor networks [5,65]. It features a component-based architecture that enables implementation of sensor network applications. TinyOS features a component library that includes network protocols, distributed services, sensor drivers, and data-acquisition tools. TinyOS features an event-driven execution model and enables fine-grained power management. It has been ported to several platforms with support for various sensor boards.

Currently, over 500 research groups and companies use TinyOS and the sensor motes developed by Crossbow [66]. A partial list of research projects [67] currently under way is presented in Table 1.4. A partial list of companies [67] that use TinyOS in commercial developments is provided in Table 1.5.

TABLE 1.4 TinyOS Research Projects

Project	Description
Calamari [68]	Localization solutions for sensor networks
CotsBots [69]	Inexpensive and modular mobile robots built using off-the-shelf components to investigate distributed sensing and cooperation algorithms in large (>50) robot networks
Firebug [70]	Berkeley civil engineering project for the design and construction of a wildfire instrumentation system using networked sensors
galsC [71]	Language and compiler designed for use with the TinyGALS [72] programming model
Great Duck Island [19]	Remote habitat monitoring of Leach's Storm Petrel
Mate [73]	Application-specific virtual machines for TinyOS networks
PicoRadio [74]	Development of mesoscale low-cost transceivers for ubiquitous wireless data acquisition that minimizes power/energy dissipation
Sensing Structural Integrity [75]	Reporting the location and kinematics of damage during and after an earthquake
Telegraph [76]	Study of various technologies for adaptive data flow such as streaming data from sensors, logs, and peer-to-peer systems
TinyDB [77]	Query processing system for extracting information from a network of TinyOS sensors
TinyGALS [72]	Globally asynchronous and locally synchronous model for programming event-driven embedded systems
XYZ On A Chip [78]	Research focused on airflow measurement technology and the use of sensor networks for controlling indoor temperature

TABLE 1.5 TinyOS Commercial Research Projects

Project	Description
Digital Sun's S. Sense [79]	Soil-moisture sensor system for sprinkler systems to keep grass green while conserving water
Dust Networks [80]	Manufacturers of resilient, self-healing wireless mesh networks optimized for low data-rate applications
Crossbow [66]	Manufacturer of wireless sensor networks and wireless data loggers that use TinyOS
Ember [81]	Developer of wireless semiconductor systems that consist of chips embedded with networking software and low-frequency radio transmitter technology that support wireless mesh monitoring and low-power autohealing management networks
Sensicast [82]	Provider of end-to-end intelligent wireless sensor network solutions to original equipment manufacturers (OEMs) and system integrators
Sensit [83]	Developers of the most highly used wind-eroding mass sensor worldwide

1.9 SUMMARY

This chapter outlined some envisioned, as well as implemented, wireless sensor network applications. A brief overview of the various types of services required by wireless sensor network applications was also presented. Although advances in technology have increased the processing, storage, and communication capabilities of sensor nodes, the main obstacle yet to be overcome is the limited power available to sensor nodes. As battery technology and energy-harvesting techniques improve, wireless sensor network applications will continue to flourish.

As wireless sensor network applications become increasingly more powerful and proliferate, additional services that support their increased functionality will also be required. Several research groups have begun to develop middleware to provide needed services to support wireless sensor networks. Ideally, deployed wireless sensor networks should configure, adjust, and heal themselves automatically with minimal user intervention. Information sharing among independent sensor networks, deployed within the same region, even though they are distinct, is another desirable quality. However, before these scenarios become a reality, much research remains to be done.

ACKNOWLEDGMENTS

This material is based on work supported by the National Science Foundation under Grant ANI-0086020.

REFERENCES

1. Crossbow Technology MPR2400 MICAz, from http://www.xbow.com/products/product_pdf_files/wireless_pdf/6020-0060-01_a_micaz.pdf/, December 2004.
2. See at http://www.xbow.com/products/productsdetails.aspx?sid=101.
3. Crossbow Technology's MicaZ sensor mote, from http://gyro.xbow.com/other/micaz_new.jpg, December 2004.
4. See at http://www.atmel.com.
5. J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, pages 93–104, Cambridge, Massachusetts, November 2000.
6. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, March 2002.
7. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 56–67, ACM Press, 2000.
8. C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, **11**(1):2–16, 2003.
9. J. Agre and L. Clare. An integrated architecture for cooperative sensing networks. *IEEE Computer*, pages 106–108, May 2000.
10. F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, pages 148–159, ACM Press, 2002.
11. A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 88–97, ACM Press, 2002.
12. E. Biagioni. PODS: Interpreting spatial and temporal environmental information. In *Usability Evaluation and Interface Design: Cognitive Engineering, Intelligent Agents, and Virtual Reality*, Volume I of the *Proceedings of HCI International 2001, the 9th International Conference on Human-Computer Interaction*, pages 317–321, New Orleans, Louisiana, August 2001.
13. E. Biagioni and K. Bridges. The application of remote sensor technology to assist the recovery of rare and endangered species special issue on distributed sensor networks. *International Journal of High Performance Computing Applications*, **16**(3), August 2002.
14. D. Niculescu and B. Nath. Ad hoc positioning system (APS), In *Proceedings of GLOBECOM'01 (IEEE)*, pages 2926–2931, San Antonio, Texas, November 2001.
15. K. A. Sudduth. Engineering technologies for precision farming. Presented at the International Seminar on Agricultural Mechanization Technology for Precision Farming, Suwon, Korea, May 1999.
16. C. R. Locke, G. J. Carbone, A. M. Filippi, E. J. Sadler, B. K. Gerwig, and D. E. Evans. Using remote sensing and modeling to measure crop biophysical variability. In *Proceedings of the 5th International Precision Agriculture Conference*, Minneapolis, Minnesota, July 2000.

17. P. Bonnet, J. Gehrke, and P. Seshadri. Querying the physical world. *IEEE Personal Communications*, **7**:10–15, October 2000.
18. L. Schwiebert, S. Gupta, and J. Weinmann. Research challenges in wireless networks of biomedical sensors. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 151–165, ACM Press, 2001.
19. Habitat monitoring on great duck island, from http://www.greatduckisland.net/, November 2004.
20. R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a sensor network expedition. In *Proceedings of the 1st European Workshop on Wireless Sensor Networks* (*EWSN '04*), January 2004.
21. J. Polastre. Design and Implementation of Wireless Sensor Networks for Habitat Monitoring. Master's thesis, University of California, Berkeley, May 2003.
22. See at http://www.botany.hawaii.edu/pods/.
23. F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *IEEE Network*, **18**(4):45–50, July/August 2004.
24. L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, July 1978.
25. K. M. Chandy and L. Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*, February 1985.
26. C. J. Fidge. Partial orders for parallel debugging. In *ACM SIGPLAN/SIGOPS Workshop on Parallel 4 Distributed Debugging*, 1985.
27. F. Mattern. Virtual time and global states of distributed systems. In *International Workshop on Parallel and Distributed Algorithms*, 1989.
28. D. L. Mills. Internet time synchronization: The network time protocol. In *Global States and Time in Distributed Systems*, Zhonghua Yang and T. Anthony Marsland (eds.), pages 91–102, IEEE Computer Society Press, 1994.
29. H. Dai and R. Han. Tsync: A lightweight bidirectional time synchronization service for wireless sensor networks. *Mobile Computing and Communications Review*, **8**(1):125–139, 2004.
30. J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of 5th Symposium on Operating Systems Design and Implementation (OSDI)*, pages 147–163, December 2002.
31. J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *Proceedings of the 2001 International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless and Mobile Computing*, April 2001.
32. S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems* (*SenSys*), pages 138–149, ACM Press, 2003.
33. M. L. Sichitiu and C. Veerarittiphan. Simple, accurate time synchronization for wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2003)*, Volume 2, pages 1266–1273, New Orleans, Louisiana, March 2003.
34. J. van Greunen and J. Rabaey. Lightweight time synchronization for sensor networks. In *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications*, pages 11–19, ACM Press, 2003.

35. S. Meguerdichian, S. Slijepcevic, V. Karayan, and M. Potkonjak. Localized algorithms in wireless ad-hoc networks: Location discovery and sensor exposure. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 106–116, ACM Press, 2001.
36. A. Savvides, C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 166–179, ACM Press, 2001.
37. A. Savvides, H. Park, and M. B. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 112–121, ACM Press, 2002.
38. D. Niculescu and B. Nath. Ad hoc positioning system (APS). In *Proceedings of GLOBECOM'01 (IEEE)*, pages 2926–2931, San Antonio, Texas, November 2001.
39. D. Niculescu and B. Nath. Ad hoc positioning system (APS) using AOA. In *Proceedings of IEEE INFOCOM 2003—The Conference on Computer Communications*, **22**(1): 1734–1743, March 2003.
40. D. Niculescu and B. Nath. Localized positioning in ad hoc networks. In *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications*, Anchorage, Alaska, April 2003.
41. N. B. Priyantha, A. Miu, H. Balakrishnan, and S. Teller. The cricket compass for context-aware mobile applications. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 1–14, ACM Press, 2001.
42. Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 32–43, ACM Press, 2000.
43. J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, pages 146–159, ACM Press, 2001.
44. B. Krishnamachari, D. Estrin, and S. Wicker. Impact of data aggregation in wireless sensor networks. In *International Workshop of Distributed Event Based Systems (DEBS)*, July 2002.
45. J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks. In *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications*, May 2003.
46. S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. Ght: A Geographic hash table for data-centric storage. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 78–87, ACM Press, 2002.
47. S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu. Data-centric storage in sensornets with ght, a geographic hash table. *Mobile Networks and Applications*, **8**(4):427–442, 2003.
48. D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multi-resolution storage for sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pages 89–102, ACM Press, 2003.

49. S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman. Infrastructure tradeoffs for sensor networks. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 49–58, ACM Press, 2002.
50. P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networking*, **7**(6):609–616, 2001.
51. B. Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 243–254, ACM Press, 2000.
52. G. Xing, C. Lu, R. Pless, and Q. Huang. On greedy geographic routing algorithms in sensing-covered networks. In *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 31–42, ACM Press, 2004.
53. M. Kochhal, L. Schwiebert, and S. Gupta. Role-based hierarchical self organization for wireless ad hoc sensor networks. In *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications*, pages 98–107, ACM Press, 2003.
54. I. F. Akyildiz and I. H. Kasimoglu. Wireless sensor and actor networks: Research challenges. *Ad Hoc Networks*, **2**(4):351–367, October 2004.
55. M. Conti, G. Maselli, G. Turi, and S. Giordano. Cross-layering in mobile ad hoc network design. *Computer (IEEE)*, **37**(2):48–51, February 2004.
56. Y. Zhang and L. Cheng. Cross-layer optimization for sensor networks. *New York Metro Area Networking Workshop 2003*, New York, New York, September 2003.
57. A. Woo, S. Madden, and R. Govindan. Networking support for query processing in sensor networks. *Communications of the ACM*, **47**(6):47–52, 2004.
58. S. Madden, W. Hong, J. Hellerstein, and M. Franklin. Tinydb: A declarative database for sensor networks, from http://telegraph.cs.berkeley.edu/tinydb.
59. Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *ACM SIGMOD Record*, **31**(3):9–18, 2002.
60. X. Li, Y. J. Kim, R. Govindan, and W. Hong. Multi-dimensional range queries in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pages 63–75, ACM Press, 2003.
61. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 491–502, ACM Press, 2003.
62. S. J. Park, R. Vedantham, R. Sivakumar, and I. F. Akyildiz. A scalable approach for reliable downstream data delivery in wireless sensor networks. In *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 78–89, ACM Press, 2004.
63. J. M. Reason and J. M. Rabaey. A study of energy consumption and reliability in a multi-hop sensor network. *Mobile Computing and Communications Review*, **8**(1): 84–97, 2004.
64. J. M. Rabaey, M. J. Ammer, J. L. da Silva, D. Patel, and S. Roundy. Picoradio supports ad hoc ultra-low power wireless networking. *Computer (IEEE)*, **33**(7):42–48, July 2000.
65. TinyOS Community Forum, from http://www.tinyos.net/, November 2004.
66. Crossbow Technology Inc., from http://www.xbow.com, November 2004.
67. TinyOS Community Forum related work, from http://www.tinyos.net/related.html, November 2004.

68. Calamari: a sensor field localization system, from http://www.cs.berkeley.edu/kamin/calamari/, November 2004.
69. CotsBots, from http://www-bsac.eecs.berkeley.edu/projects/cotsbots/, November 2004.
70. FireBug, from http://firebug.sourceforge.net/, November 2004.
71. galsC: A language for event-driven embedded systems, from http://galsc.sourceforge.net/, November 2004.
72. TinyGALS: A programming model for event driven embedded systems, from http://ptolemy.eecs.berkeley.edu/papers/03/tinygals/, November 2004.
73. Mate, from http://www.cs.berkeley.edu/pal/mate-web/, November 2004.
74. PicoRadio, from http://bwrc.eecs.berkeley.edu/research/pico radio/, November 2004.
75. S. D. Glaser, from http://www.ce.berkeley.edu/glaser/curee.pdf, November 2004.
76. The Telegraph Project at UC Berkeley, from http://telegraph.cs.berkeley.edu/, November 2004.
77. TinyDB: A declarative database for sensor networks, from http://telegraph.cs.berkeley.edu/tinydb/, November 2004.
78. XYZ on a chip: Integrated wireless sensor networks for the control of the indoor environment in buildings, from http://www.cbe.berkeley.edu/research/briefs-wirelessxyz.htm, November 2004.
79. Digital Sun, from http://www.digitalsun.com/, November 2004.
80. Dust networks, from http://www.dust-inc.com/products/main.shtml, November 2004.
81. Ember, from http://www.ember.com/index.html, November 2004.
82. Sensicast, from http://www.sensicast.com/, November 2004.
83. Sensit Company, from http://www.sensit.com/, November 2004.