Embedded Sensor Networks

John Heidemann[1] and Ramesh Govindan[2]

[1] USC/Information Sciences Institute johnh@isi.edu
[2] USC Computer Science Department ramesh@usc.edu

1 Introduction

An *embedded sensor network* is a network of embedded computers placed in the physical world that interacts with the environment. These embedded computers, or *sensor nodes*, are often physically small, relatively inexpensive computers, each with some set of sensors or actuators. These sensor nodes are deployed *in situ*, physically placed in the environment near the objects they are sensing. Sensor nodes are networked, allowing them to communicate and cooperate with each other to monitor the environment and (possibly) effect changes to it. Current sensor networks are usually stationary, although sensors may be attached to moving objects or may even be capable of independent movement. These characteristics: being embedded, being capable of sensing, actuation, and the ability to communicate, define the field of sensor networking and differentiate it from remote sensing, mobile computing with laptop computers, and traditional centralized sensing systems.

Although research in sensor networks dates back to the 1990s or earlier, the field exploded around the year 2000 with the availability of relatively inexpensive (sub-$1000) nodes, sensors, and radios. As of 2004, sensor networking is a very active research area with well established hardware platforms, a growing body of software, and increasing commercial interest. Sensor networks are seeing broader research and commercial deployments in military, scientific, and commercial applications including monitoring of biological habitats, agriculture, and industrial processes.

Sensor networks present challenges in three key areas. First, *energy consumption* is a common problem in sensor network design. Sensors are often battery operated and placed in remote locations, so any activity drains the sensor battery, and bringing the node closer to death.[3] Second, how sensors *sense and interact* with the physical world is of great interest. Sensor networks focus on collaborative signal processing algorithms to exploit multiple,

[3] Greg Pottie, personal communication.

physically separate views on the environment. Finally, with tens, hundreds, or even thousands of sensor nodes, the network and applications as a whole must be *self-configuring*.

This chapter reviews each of these areas of sensor network design, beginning with common hardware platforms, then considering networking software and applications.

2 Hardware Platforms and Sensors

The wide availability of common hardware platforms, radios, and sensors has been an enabler for sensor networking, both in the research and commercial communities.

Node hardware

A typical sensor node contains a general-purpose CPU and working memory, some kind of long-term stable storage such as flash memory or disk, and I/O capabilities to support sensors. Sensor nodes have evolved into two broad categories: *small devices* with 8-bit microcontrollers as CPUs, 10–100KB of working memory, and 100–1000KB of flash secondary storage; and *larger devices* with 32-bit CPUs, megabytes each of working memory and secondary storage.

Motes are representative of the smaller class of devices [22]. The current generation of Mica-2 motes uses an Atmega128 embedded processor running at about 4MHz, providing 128KB flash memory for program code, 4KB working RAM, 8-channels of analog-to-digital converts, 48 digital I/O lines, a UART and SPI serial interface. Motes have evolved significantly over the last several years; originally designed at Berkeley, they are now commercially available from several companies including Crossbow, Dust Networks, and Telos. Similar classes of embedded-controller-based devices are available from other academic and commercial institutions, with examples including the Nymph from University of Colorado [1], and BTnodes from ETH Zurich [35])

The larger class of devices is exemplified by products such as the Stargate (designed by Intel, available from Crossbow Technologies) or the Cerfcube (from Intrinsyc). These devices are used in a variety of embedded applications. In the sensor network context, they are typically used as gateways to a collection of motes, or for applications that require heavier-duty signal processing. Each such device employs an X-Scale or ARM based processor, has upwards of 64 MB working memory and 1GB of flash-based secondary storage. They support many connectivity options including USB and 802.11 wireless, and a 51-pin mote connector allowing use of a mote and its radio.

Power management is a concern in both classes of devices. Individual control of hardware components (CPU, storage, radio, sensors) is necessary for power management. Many systems are battery powered. Energy harvesting is of growing importance, often via solar cells or vibration harvesting.

Sensors

Sensing technology has also kept pace with miniaturization of radios and processors, especially with the proliferation of MEMS sensors. Many such sensors have been incorporated into existing sensor node platforms.

Despite their diversity, the principle of operation behind most of these MEMS sensors is the same. They all rely on environmental factors inducing changes in the electrical properties of appropriately chosen materials. The sensors incorporate sensitive circuitry to detect changes in these electrical properties, and are calibrated to correctly measure the corresponding environmental phenomenon. For example, a temperature sensor relies on changes in the resistivity of certain materials with temperature. The choice of material ranges from metals to semiconductors and is dictated by the required sensing range and sensitivity. Similarly, a light sensor uses photoconductive materials whose electrical characteristics vary with the amount of light falling upon them. Finally, accelerometers measure the voltage induced by structural deformations of piezo-electric materials; these deformations are caused by vibrations or by acceleration.

There is a very large industry devoted to manufacturing small MEMS sensors. This industry is segmented by application (*e.g.*, companies such as Delphi cater to automotive manufacturers) and by sensor type (*e.g.* Silicon Designs focuses entirely on vibration sensors). However, only a handful of companies (examples include Ember, and Millenial Net), focus on applications of wireless networked sensing.

3 Software and Protocols

Sensor networking has seen an enormous amount of research activity in the last five years, making it difficult to do justice to this large body of literature. Our exposition takes a systems approach, describing the components of an emerging general-purpose sensor networking infrastructure.

3.1 Networking

As the name implies, networking is a central component of sensor networks. Networking is important because it provides the glue that allows individual nodes to collaborate. In addition, the radio is a major consumer of energy in small sensor nodes, often 20–40% of the power draw when all components are on. Thus optimizing networking protocols can greatly extend the lifetime of the sensor network as a whole.

This section considers networking in sensor nets at the link layer, with media-access protocols, and at the network layer, with routing protocols. We also consider *topology control*, a service that can be part of either layer, or could be considered in between these two layers.

MAC protocols

Energy conservation is a key concern at the MAC protocols, and so before reviewing protocols we briefly describe MAC-related sources of energy consumption [50]. Packet *collisions* waste energy by forcing packets to be retransmitted, *idle listening* is the cost of actively listening for potential packets, *overhearing* is the cost of receiving packets intended for other destinations, and *control traffic* represents MAC-level maintenance overhead. Since many sensor networks are quiescent between sensor readings, idle listening can easily become the largest energy cost in a sensor net.

The IEEE 802.11 protocol (popularly known as "wi-fi") is contention-based MAC (carrier-sense, multiple-access or CSMA) now seeing wide commercial deployment. Intended for laptop computers, it provides good high-speed communication (up to 54Mb/s in some versions) for larger sensor network nodes. Unfortunately, the ad hoc mode of 802.11 that is required for peer-to-peer communications in a sensor network has very little support for energy conservation, and many sensor networks require bitrates less than 100kb/s, so the protocol is unsuitable for smaller and more power-constrained nodes.

Time-division multiplexing (TDMA) protocols were used in early sensor networks [43]. By scheduling media access they can largely avoid collisions, idle listening, overhearing, thus greatly reducing energy consumption. Their disadvantage is that they often assume clustering, taxing the cluster head and making mobile operation more difficult.

Small sensor networks generally require relatively low speed (20–40kb/s), simple protocols, precluding high-speed 802.11 and more complex TDMA protocols. Recent research has proposed 802.11-like MAC protocols designed to conserve energy by avoiding overhearing (PAMAS [42]), and idle listening (S-MAC [50, 51]). As an example protocol, S-MAC synchronizes most nodes into a sleep-schedule. Nodes regularly wake up, contend for the media if they have data to send, then either transmit data or go to sleep. By adjusting the sleep duration, duty cycles of 1–50% are possible to reduce the cost of idle listening. Adaptive listen [51] and future-Request-to-Send of T-MAC [46] extended these ideas to provide better throughput when there are multiple packets to send or when data travels over multiple hops.

IEEE 802.15.4 (also known as "Zigbee") is a recently standardized protocol targeted at sensor network and home automation applications. It includes an optional fixed duty cycle to avoid idle listening similar to these research protocols. Although it is still too early see how this protocol compares to current research, a standard protocol in this domain should spur commercial developments.

Network layer

As with MAC protocols, overhead is an important concern for sensor network routing protocols. Here the major source of overhead is control traffic: the number of routing update- or request-messages that are required. We next

review routing protocols, both Internet Protocol-based ad hoc networking and non-IP-based schemes.

The IETF (Internet Engineering Task Force) is standardizing *ad hoc* routing protocols for wireless, IP-based networks. Ad hoc routing protocols are usually grouped into *pro-active* protocols (for example, DSDV) which precompute routes to some or all destinations, and *reactive* protocols (for example, AODV and DSR), which compute routes to specific destinations only when prompted by traffic. The control traffic overhead of these protocols is proportional to the rate at which links change and, for reactive protocols, the rate traffic is sent to new destinations. Reactive protocols are a good match for very dynamic networks, such as those with many mobile nodes, since they only maintain routes to active destinations. Many sensor networks today have only stationary nodes; in these cases pro-active protocols may be preferred because links change relatively infrequently and such protocols are simpler and have no delay searching for a route when traffic is sent to a new destination.

In addition to IP-based routing protocols, geographic routing and directed diffusion are protocols more specific to sensor networks. Although originally proposed for wired networks, geographic routing protocols such as GPSR and similar protocols [3, 26] exploit the spatial nature of sensor networks and the spatial-dominated nature of radio propagation.

Directed diffusion combines a distance-vector-like, reactive routing protocol with an attribute-based routing scheme and an emphasis on processing data in the network [24]. Variants of directed diffusion provide several different routing mechanisms under the same interface [18]. Diffusion uses attribute-based routing instead of address-based or geographic routing diffusion combines attribute-based resource discovery with routing, and it allows applications to focus on the desired data rather than specific sensors. Diffusion suggests that processing data in the network is important for efficiency. Examples of in-network processing are duplicate suppression, data aggregation, and statistical filtering. When data is generated from multiple sensors, in-network processing can merge this data near those sites rather than sending all data out, thus greatly reducing energy consumption. (This approach has been adopted by several non-diffusion systems as well, for example TinyDB, described below.)

3.2 Systems Services

Beyond the lower-level networking primitives, a usable sensor networking system must provide several additional services. Many of these services, such as operating systems, security, time synchronization, and resource discovery, are also found in traditional wired and wireless networks. However, some services such as localization are unique to sensor networks. In this subsection, we briefly discuss some of the services that sensor networks will implement, and describe challenges in the design of these services.

Operating Systems and Code Development Tools

The sensor networking community typically uses embedded (and, possibly, real-time) versions of existing operating systems such as Linux for the larger devices discussed above. These embedded versions provide largely the same programming support as their regular counterparts, but with additional device-level support for embedded controllers, flash memory, and other peripherals specific to these devices. As such, not much research has been required on new operating systems support for these larger devices.

By contrast, the smaller devices (such as the motes) have required novel directions in operating system design. One such direction has been the development of a POSIX-compliant multi-threaded OS for these small devices [1]. TinyOS [21], an operating system for the motes and widely used by many research groups as well as in some segments of industry, departs significantly from the traditional multi-threaded model of modern operating systems. Rather, TinyOS relies on the observation that most sensor networking applications will be *event-driven*: *i.e.*, that applications will react to external sensed events. It is structured such that *components* (software modules that provide a distinct abstraction, either of a hardware device or of some software functionality such as a send-receive networking interface) can invoke *events* in other components. Typically, in response to a sensed event or some hardware interrupt, a chain of such component invocations can be used to process the event. In addition, TinyOS also provides two other abstractions: a *task* enables components to effectively use the idle processor for computations, and a *command* which is invoked in order to get a component to perform an action (such as sending a network message, or setting some device parameters).

Taken together, these abstractions allow a programmer to write an application as a component graph: nodes in this graph are components, and links signify event and command invocation from components or their associated tasks. In addition, the emphasis on event-driven programming rather than multi-threading avoids the memory cost of run-time stacks for each thread. This powerful functionality promotes code re-use, improves modularity (easily reuse of components), and minimizes object code size for memory-constrained devices.

Associated with TinyOS is a programming language called nesC, which contains language-level constructs for TinyOS's abstractions: components, tasks, events and commands. This approach promotes compile-time program checking, as well as automated construction and management of the component graph. In addition, it provides support for parameterized, compile-time memory allocation to avoid the memory costs of dynamic memory allocation.

Finally, the community uses several simulation and emulation tools that enable code development, debugging, and system evaluation. One is *ns-2*, a general purpose networking simulator, together with its wireless extensions, which was widely used to develop and evaluate sensor networking routing protocols. More recently, sensor network specific simulators have seen increased

use. Among these, TOSSIM [29] enables developers to simulate a network of motes, and run actual application and protocol code on this network. Emstar [14] is a flexible programming environment for larger sensor nodes. As a simulation environment, it includes support for simulation of hybrid networks of large and small devices and several radio propagation and sensor models.

Node Localization

Localization is the functionality by which nodes autonomously determine their *position* in two or three dimensions. This is a crucial service for sensor networks since location provides invaluable context in interpreting sensed data. In recent years, there has been significant work in localization for sensor networks and networks of embedded devices.

An important focus of the localization literature has been robust techniques for estimating distances between nodes (*ranging*). The networking community has focused on two classes of ranging techniques: RF-based ranging and acoustic ranging. RF-based ranging, as exemplified by the SpotON [20] and Calamari [47] systems, is based on the premise that by measuring received signal strength a receiver can determine its distance to a transmitter. This presumes that RF propagation in an environment can be accurately characterized by a simple path loss model with known parameters. Using this technique, nodes can estimate distances to all neighbors within radio range. Range errors upwards of 10% of the nominal radio range have been reported in the literature [47], usually after a fairly involved calibration step that estimates the path loss parameters and adjusts for variations in transceiver characteristics. As an alternative to modeling radio propagation, the RADAR system has proposed building a database of receive signal strength as a function of location [2]. This approach requires surveying the environment to precompute the database and assumes propagation is relatively time invariant. A second class of ranging schemes is based on measuring the time-of-flight of an acoustic or ultrasound signal [13, 38]. More precisely, these techniques measure the difference in arrival times of simultaneously transmitted radio and ultrasound signals, then estimate distances knowing the speed of sound. Some approaches in acoustic ranging use spread spectrum approaches for resilience to multipath, and employ techniques to correct for latencies induced by other system components [13]. Such techniques provide an order of magnitude better accuracy (1–2% error) than simple time-of-flight over distances of 3–6 meters.

Ranging is a component of a *localization system* of which there are, broadly speaking, two kinds: infrastructure-based and ad-hoc. Systems in the former class fix node positions by assuming the existence of some external infrastructure (typically beacons with known positions and with known or predetermined deployments). In this class, there has been extensive work on distributed position inference using specialized beacons [5], in-building localization systems that enable position for context-aware applications ([17], among others), and systems for estimating orientation of handheld devices [34].

Similarly, a large body of work has examined algorithms for ad-hoc localization schemes. Perhaps the earliest pieces of work in the area of sensor network localization can be attributed to Bulusu *et al.* [5], Niculescu *et al.* [32] and Savvides *et al.* [39, 38]. Niculescu *et al.* propose that nodes first estimate their distances to anchors using one of several techniques (DV-hop, DV-distance, and a Euclidean scheme), then fix their own position using these distances. Savvides *et al.* proposes an N-hop multilateration scheme. That work also discusses a Kalman filtering based position refinement phase to improve position estimates. In later work, they discuss the error characteristics and the dependence on network size and anchor density of their schemes [37]. Finally, Langendoen *et al.* [27] discuss a fairly detailed comparison of the above schemes in the face of ranging errors, different node densities and anchor fractions.

Time Synchronization

Sensor networks are predicated on the ability of sensor nodes to *collaboratively* detect events. Time synchronization is often a crucial requirement for collaborative detection—collaborating nodes may need to temporally correlate their sensor readings. The problem of node time synchronization has received extensive attention in the sensor networks literature. Much of this literature borrows heavily from early work on Internet time synchronization.

Networked time synchronization relies on time stamping a message at both the sender and receiver, and reconciling their clocks based one or more such message exchanges. Between sending a message and receiving it there are several kinds of delays introduced: sender-side processing delay, message propagation delay, message transmission delay, and receiver side processing delays. Techniques for time synchronization differ in how they estimate, or eliminate various sources of delay.

For example, the simplest approach time stamps messages close to the radio hardware (thereby eliminating processing delays). Thus, a single message is sufficient to reconcile clocks if it is assumed that propagation delay is negligible. Two messages are sufficient to account for propagation delay as well [11]. Reference Broadcast Synchronization (RBS) avoids error introduced by variance in sender-side timing by comparing the same broadcast message received at the two *receivers*, and estimating receiver processing by averaging over several received packets [10]. As an aside, many of these techniques can be used to synchronize nodes after the occurrence of an event, an approach called *post-facto* synchronization [9].

Thus far, we have discussed how two nodes synchronize their clocks with each other. Some research has looked at synchronizing clocks network-wide to a reference [11, 10]. Most of these techniques rely on using one of the above methods to synchronize all the clocks to a reference clock hop-by-hop. Using such techniques, clock synchronization error increases linearly with network diameter. The existence of techniques that have better error characteristics is known theoretically, but no practical implementations of such techniques exist.

Resource Discovery

Resource discovery is a problem growing out of the field of ubiquitous computing: when a device enters an area, how can it identify relevant local resources. In ubiquitous computing, relevant resources are network services such as printers and mail servers, to be used by people. Sun Microsystems' Jini includes resource discovery services for a local network, while web search engines such as Google can be thought of as Internet-wide resource discovery services.

Sensor networks today are typically more application-specific and homogeneous, and so today there is little need to locate shared services. In many sensor networks today the only service not present at all nodes is a wide-area network connection; discovery of an Internet connection is easily integrated with routing. As sensor networks become more complex we expect that individual nodes will become more heterogeneous. As cameras, additional storage, and other services are deployed, service discovery will become more important. In sensor networks today, Directed Diffusion combines resource discovery with routing [18]. Other protocols such as ReOrg support service heterogeneity (in its case, wall-powered nodes) integrated with the topology configuration protocol [8].

Databases and Storage Services

Individual sensor nodes in a sensor network produce many sensor readings. Nodes may also collaboratively detect events by exchanging these readings. We will collectively refer to readings and events as sensor data. An important systems challenge for sensor networking is the design of mechanisms for retrieving sensor data.

Protocols such as directed diffusion suggested data-centric communication as an architectural principle that governs the design of low-level mechanisms for accessing sensor data. At a higher level, a natural paradigm for accessing sensor data is to treat the sensor network as a distributed relational database. The Cougar [49] and TinyDB [31] systems allow users to specify sensor data of interest in a declarative fashion, using an SQL query of the form:

```
SELECT AVG(temp)
  FROM sensors
  WHERE loc in (35,40,100,120) and light > 1525 lux
  SAMPLE PERIOD 35 seconds
```

Such a query allows the user to obtain the average temperature seen at all nodes observing a sufficiently high light intensity which are located within a specified region. These systems can be seen as providing a powerful, yet widely used, *programming* paradigm for sensor networks.

Both systems are implemented using data-centric communication primitives. For example, in TinyDB a query is flooded throughout the network, and nodes whose sensor data match the query respond. In principle, this is similar to interests and data messages in Directed Diffusion. Indeed, TinyDB can be implemented using Directed Diffusion.

While TinyDB and Cougar work well for continuous queries (those for which responses stream back continuously), the high overhead of flooding makes it unsuitable for one-shot queries. Researchers have focused on a class of systems that more efficiently support one-shot queries. These systems are built on an efficient rendezvous mechanism called *data-centric storage* [40]. In data-centric storage, a hash function is used to map a key associated with a data item to a geographic location. A geographic routing protocol called GPSR [26] is used to store the item at that location. A node wishing to retrieve items matching that key would use the same hash function and route a query to that node. Such a mechanism avoids flooding the query throughout the network. Depending on how the hash function is constructed, this mechanism can be used to construct a variety of storage structures that support sophisticated queries. These storage structures include distributed hash tables [36], distributed multi-dimensional indices [30], and storage hierarchies [12].

Remote Programming

Reprogrammability is a key characteristic of software systems. While simple sensor networks may be configured "in the factory" and then discarded, sensors deployed for longer periods of time in remote locations motivate in-situ reprogrammability to meet changing application requirements or just to fix bugs. Several flavors of reprogramming have been reconsidered by the community. *Retasking* usually refers to reconfiguring an application's parameters to match some pre-anticipated needs. *Scripting* and *virtual-machines* represent the ability to reconfigure a sensor network at a high-level, often by recombining pre-deployed lower-level components. Finally, true *reprogramming* is reserved for replacing the complete operating image of the sensor node.

Retasking has been explored in several environments. One example of retasking is directed diffusion [24] and filters [19], where application-specific attributes can be used to tune a fielded application. Another widely used example is TinyDB [31] where new queries are downloaded into the network. In both of these cases run-time information is distributed through the network to reconfigure pre-deployed filters or database operators.

A more generic facility is possible with scripting or virtual machines. In SensorWare [4], pre-configured components are reconfigured on-the-fly by commands distributed over the network. Scripting is distinguished from retasking because the configuration information is provided by a script in a high-level language (Tcl, in the case of SensorWare), allowing more sophisticated reconfiguration than is possible with the static data structures of simple retasking. Virtual machines were popularized with Java; Maté is an example of virtual machines applied to sensor networks [28]. While traditional virtual machines provide a primitive, low-level instruction set, Maté emphasizes very high-level, application-specific instructions to maximize code-density.

Reprogramming the complete sensor node is a delicate process where a software image is transfered over one or more hops, verified, and the sensor

node carefully reboots to run the new code. Two recent systems have described mote-level reprogrammability: Hui and Culler's system [23] and MOAP [44]. Both carefully segment and transfer a relatively large (multi-kilobyte) software image, with Hui's system emphasizing rapidly propagating the image throughout the entire network with pipelining, while MOAP strives to transfer a complete image to nearby neighbors before forwarding data further.

Many of the above approaches assume the entire sensor network is to be reprogrammed. Since messages in diffusion are addressed to nodes identified by particular attributes it is easy to retask part of a sensor network. SensorWare has also used scripts to reprogram parts of a network. In principle, similar techniques could be applied to the other approaches.

Security

Security issues have not received as much attention as some of the other research areas in sensor networks. This is understandable, since often interesting security research is spurred by vulnerabilities learned from large-scale deployments, of which there are relatively few. There is, of course, a general acceptance that security is of paramount importance in sensor networks. They are vulnerable to a wide variety of denial of service attacks at all levels, ranging from the physical to the application layer [48].

Existing sensor network security research has mostly focused on adapting security mechanisms to the computational and messaging constraints imposed by tiny sensor devices [33, 25]. This line of research attempts to implement encryption and message authentication mechanisms by relying on shared symmetric or group keys augmented with message counters. Some of these mechanisms have been implemented on the motes in order to provide secure communication at the link layer.

3.3 Application Primitives

Sensor networks will, in general, be used to sense phenomena of different kinds. Broadly speaking, phenomena may be of two kinds: *diffuse* phenomena like fires, clouds, contaminants *etc.*, and *point* phenomena like animals, tanks and other targets. Generic techniques for sensing these kinds of phenomena might form useful application-level primitives which ease the task of developing new applications. Some sensor networks research has focused on primitives for these two kinds of phenomena.

Diffuse phenomena are distinguished by their spatial extent, which is generally larger than the average inter-sensor spacing. Thus, an important primitive for such phenomena is one that detects and tracks the *boundary* of the phenomenon. Not much research attention has been bestowed on this class of problems, aside from isolated pieces of work that have discussed a technique to compute the approximate boundary of a phenomenon along a hierarchical structure, and techniques to robustly, yet locally, estimate whether a node lies on the boundary of a phenomenon or not.

Rather more attention has been devoted to application-level primitives for point phenomena. One can decompose the problem of sensing such phenomena into two smaller problems: *target localization*, which determines *where* the point phenomenon or target is, and *tracking*, which updates the path of the target as it moves. Both of these problems have been examined in the sensor network context.

For target localization, a simple technique would be to use the "closest point of approach", *i.e.*, say that the location of the target is the location of the sensor which detects the target with the highest intensity. Variants of this algorithm might pinpoint the target as being located at the weighted centroid of all sensors that sense the target. The accuracy of such techniques depends heavily on deployment density. A more sophisticated approach, and one that has been studied fairly extensively in the signal processing literature, relies on *triangulating* the target position based on the observed delay differences in the received signal at a cluster of sensors within the network. This technique has been shown to work quite well [7].

Having localized the target, the next challenge is to track the target as it moves through the sensor field. A simple representation of the target's track is the sequence of its locations over time, and this may be computed by sending the target's location periodically to a base station. This approach can incur significant communication cost, so more sophisticated techniques rely on handing off the track to sensors along the target's path. Furthermore, it is possible to be more energy-efficient by *waking up* sensors in advance of a target's arrival. However, this requires techniques than can predict the target's track. A body of work has focused on *information-directed* approaches to solve this problem [52]. These approaches maintain a continuously updated belief state about target location that allows them to probabilistically determine to which sensor the target's track should be handed off.

4 Applications

The sensor network community is investigating several disciplines in which sensor networks might be applicable for various purposes. The following paragraphs discuss these potential applications briefly, sketching applications in the military, the sciences and environmental monitoring, and civil and industrial areas. For many of these applications, sensor networks will enable in-situ sensing at unprecedented spatial scales.

Military applications:

Military applications supported much early work in sensor networks. Securing an area to detect intruders and monitoring vehicle traffic on a road or in open terrain were a focus of the DARPA SensIT program. More recently researchers have demonstrated a sensor-network-based sniper localization system [41].

Environmental monitoring applications:

Many current applications for sensor networks are in areas of biology and life sciences, where a common theme is the ability of sensors to take observations in much more detail and for much longer than is possible today. We briefly evaluate habitat monitoring, marine microorganism monitoring, contaminant transport and precision farming.

Habitat monitoring has been the focus of great interest in the sensor network community [45]. Examples include micro-climate monitoring at James Reserve, and nest monitoring at Great Duck Island (see [45] for details and additional references). These applications provide an ideal testing ground for sensor networks because they require fairly simple monitoring (light, temperature, sound, perhaps presence or absence of an animal) at tens of stations. This level of monitoring is not possible without sensor networks because human observations would be too invasive to the environment and centralized or wired monitors cannot span the physical area.

Marine biologists envision using sensor networks to obtain data at fine spatial scales (a few meters to tens of meters). There is need for such data in their application domain, and current instrumentation technology is inadequate or too expensive to fulfill this need. The time evolution of red tides (rapidly formed colonies of algae that are harmful to fish and birds) is poorly understood, and appears to be triggered by small scale temperature, light, and nutrient variations. Sensor networks can be deployed at this scale, and have been used in a laboratory setting to gather data. An interesting twist is the addition of limited actuation to such networks where the sensors may move (*e.g.*, in a small boat) a little in order to obtain better quality data or to vary spatial coverage.

A similar use is envisioned by environmental engineers, who see sensor networks helping them build accurate models of contaminant seepage in soil. Data at fine spatial scales can be used to more precisely model contaminant flow [16] and thus predict contamination of scarce groundwater resources. In the longer term, such networks can be used for monitoring the compliance of industries to regulations that govern the release of contaminants into the soil. A closely related area is precision farming, where detailed monitoring enabled by dense sensor deployment could allow more effective use of fertilizers.

Civil and commercial applications:

Finally, there is growing interest in sensor networks in civil engineering and industrial applications.

Seismologists envision using sensor networks to understand the propagation of earthquakes at fine spatial scales. This propagation is critically affected by soil conditions, and can impact how much earthquakes affect buildings and other structures. A related application is structural monitoring [6]: sensor networks can be used to measure the response of a building to vibrations, and

variation of these responses over time can be used to detect and localize damage in a variety of structures (buildings, bridges, ships).

Transportation networks are an important economic part of all cities, and it is not surprising that there is a fairly large investment in traditional fixed sensors and centralized Traffic Monitoring Systems. Researchers are exploring how sensor networks can augment this infrastructure in two different ways. Rapidly deployable traffic-monitoring sensor networks may be useful to temporarily collect data for development or pollution-related traffic studies in areas that do not warrant long-term monitoring [15] More radically, several research groups have proposed a future where each car has its own sensors that can communicate with nearby cars, avoiding centralized management and enabling new applications.

Finally, there is growing interest in industrial applications of sensor networks to closely monitor manufacturing and safety conditions. Although these applications are just now emerging, promising areas include industrial monitoring in the oil industry (Ember), environmental monitoring in semiconductor processing facilities (Intel), and even monitoring of art in museums (Sensicast).

5 Conclusions

This chapter has surveyed embedded sensor networks. With recent hardware advances for small, inexpensive, networked sensors, a growing body of software components to link them together into a whole, and applications in many areas, embedded sensor networks are an active and growing area of embedded computing.

References

1. H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, and R. Han B. Shucker, J. Deng. MANTIS: System support for MultimodAl NeTworks of In-situ Sensors. In *Proceedings of the 2nd ACM Workshop on Sensor Networks and Applications*, pages 50–59, San Diego, CA, USA, September 2003. ACM.
2. Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of the IEEE Infocom*, pages 775–784, Tel Aviv, Israel, March 2000. IEEE.
3. Prosenjit Bose, Pat Morin, Ivan Stojmenovic, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proceedings of the Third ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (Dial M)*, pages 48–55, Seattle, Washigton, USA, August 1999. ACM.
4. Athanassios Boulis, Chih-Chieh Han, and Mani B. Srivastava. Design and implementation of a framework for efficient and programmable sensor networks. In

Proceedings of the ACM/Usenix International Conference on Mobile Systems, Applications, and Services (MobiSys), pages 187–200, San Francisco, CA, USA, May 2003. ACM.

5. Nirupama Bulusu, John Heidemann, and Deborah Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, October 2000.
6. J. Caffrey, R. Govindan, E. Johnson, B. Krishnamachari, S. Masri, and G. Sukhatme. Networked Sensing for Structural Health Monitoring. In *Proceedings of the Fourth International Workshop on Structural Monitoring and Control*, June 2004.
7. J.C. Chen, L. Yip, J. Elson, H. Wang, D. Maniezzoand R.E. Hudson, K. Yao, and D. Estrin. Coherent Acoustic Array Processing and Localization in Wireless Sensor Networks. *Proceedings of the IEEE*, 2003.
8. W. Steven Conner, Jasmeet Chhabra, Mark Yarvis, and Lakshman Krishnamurthy. Experimental evaluation of synchronization and topology control for in-building sensor network applications. In *Proceedings of the Second ACM Workshop on Sensor Networks and Applications*, pages 38–49, San Diego, CA, USA, September 2003. ACM.
9. Jeremy Elson and Deborah Estrin. Time synchronization for wireless sensor networks. In *Proceedings of the 15th IEEE International Parallel and Distributed Processing Symposium*, pages 1965–1970, San Francisco, CA, USA, April 2001. IEEE.
10. Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA, December 2002.
11. Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 138–149. ACM Press, 2003.
12. Deepak Ganesan, Deborah Estrin, and John Heidemann. DIMENSIONS: Why do we need a new data handling architecture for sensor networks? In *Proceedings of the ACM Workshop on Hot Topics in Networks*, pages 143–148, Princeton, NJ, USA, October 2002. ACM.
13. L. Girod and D. Estrin. Robust Range Estimation Using Acoustic and Multimodal Sensing. In *Proc. IEEE International Conference on Intelligent Robots and Systems*, Maui, Hawaii, USA, October 2001. IEEE.
14. Lewis Girod, Jeremy Elson, Alberto Cerpa, Thanos Stathopoulos, Nithya Ramanathan, and Deborah Estrin. Emstar: a software environment for developing and deploying wireless sensor networks. In *Proceedings of the USENIX Conference Proceedings*, pages 283–296, Boston, Mass., USA, June 2004. USENIX.
15. Genevieve Giuliano and John Heidemann. Rapidly deployable sensors for vehicle counting and classification. `http://www.isi.edu/ilense/metrans/`, 2004. Work in progress.
16. T. Harmon. Networked Sensing in Support of Real-time Parameter Estimation. Association of Environmental Engineering and Science Professors, 2003.
17. Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. In *Proceedings of the ACM International Conference on Mobile Computing and Networking*, pages 59–67, Seattle, WA, USA, August 1999. ACM.

18. John Heidemann, Fabio Silva, and Deborah Estrin. Matching data dissemination algorithms to application requirements. In *Proceedings of the ACM SenSys Conference*, pages 218–229, Los Angeles, California, USA, November 2003. ACM.
19. John Heidemann, Fabio Silva, Yan Yu, Deborah Estrin, and Padmaparma Haldar. Diffusion filters as a flexible architecture for event notification in wireless sensor networks. Technical Report ISI-TR-556, USC/Information Sciences Institute, April 2002.
20. J. Hightower, C. Vakili, G. Borriello, and R. Want. Design and Calibration of the SpotON Ad-Hoc Location Sensing System, 2001. Unpublished manuscript.
21. Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System archtecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, 2000.
22. Jason L. Hill and David E. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, Nov/Dec 2002.
23. Jonathan W. Hui and David Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd ACM SenSys Conference*, page to appear, Baltimore, Maryland, USA, November 2004. ACM.
24. Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking*, pages 56–67, Boston, MA, USA, August 2000. ACM.
25. C. Karlof, N. Sastry, and D. Wagner. TinySec: Link-Layer Encryption for Tiny Devices. In *Proceedings of the 2nd ACM SenSys Conference*, page to appear, 2004.
26. Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking*, pages 243–254, Boston, Mass., USA, August 2000. ACM.
27. K. Langendoen and N. Reijers. Distributed Localization in Wireless Sensor Networks: A Quantitative Comparison. Technical Report PDS-2002-003, Technical University, Delft, November 2002.
28. Philip Levis and David Culler. Maté: A tiny virtual machine for sensor networks. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 85–95, San Jose, CA, USA, October 2002. ACM.
29. Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: accurate and scalable simulation of entire tinyOS applications. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 126–137, Los Angeles, California, USA, 2003. ACM Press.
30. Xin Li, Young Jin Kim, Ramesh Govindan, and Wei Hong. Multi-dimensional Range Queries in Sensor Networks. In *Proceedings of the ACM SenSys Conference*, pages 63–75, Los Angeles, California, USA, November 2003.
31. Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: Tiny AGgregate queries in ad-hoc sensor networks. In *Proceedings of the Usenix Symposium on Operating Systems Design and Implementation*, pages 131–146, Boston, Massachusetts, USA, December 2002. USENIX.
32. D. Niculescu and B. Nath. Ad-hoc Positioning System. In *Proceedings of IEEE Globecom*, 2001.

33. Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 189–199. ACM Press, 2001.
34. Nissanka B Priyantha, Alen K. L. Miu, Hari Balakrishnan, and Seth Teller. The Cricket Compass for Context-Aware Mobile Applications. In *Proc. of Sixth ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Rome, Italy, July 2001.
35. The Smart-Its Project. Smart-its home page. `http://www.smart-its.org/`, 2003.
36. Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, and Scott Shenker. GHT: A geographic hash table for data-centric storage. In *Proceedings of the ACM Workshop on Sensor Networks and Applications*, pages 78–87, Atlanta, Georgia, USA, September 2002. ACM.
37. A. Savvides, W. Garber, S. Adlakha, R. Moses, and M. Srivastava. On the Error Characteristics of Multihop Node Localization in Wireless Sensor Networks. In *Proceedings of First International Workshop on Information Processing in Sensor Networks*, 2003.
38. A. Savvides, H. Park, and M. Srivastava. The Bits and Flops of the N-hop Multilateration Primitive for Node Localization Problems. In *Proceedings of the First International Workshop for Wireless Sensor Networks and Applications (WSNA)*, 2002.
39. Andreas Savvides, Chih-Chien Han, and Mani Srivastava. Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors. In *Proc. of Seventh ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pages 166–179, Rome, Italy, July 2001. ACM.
40. Scott Shenker, Sylvia Ratnasamy, Brad Karp, Ramesh Govindan, and Deborah Estrin. Data-Centric Storage in Sensornets. In *Proc. ACM SIGCOMM Workshop on Hot Topics In Networks*, pages 137–144, Princeton, NJ, 2002.
41. Gyula Simon, Akos Ledeczi, and Miklos Maroti. Sensor network-based countersniper system. In *Proceedings of the 2nd ACM SenSys Conference*, page to appear, Baltimore, Maryland, USA, November 2004. ACM.
42. S. Singh and C.S. Raghavendra. PAMAS: Power aware multi-access protocol with signalling for ad hoc networks. *ACM Computer Communication Review*, 28(3):5–26, July 1998.
43. K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie. A self-organizing sensor network. In *Proceedings of the 37th Allerton Conference on Communication, Control, and Computing*, Monticello, Ill., USA, September 1999.
44. Thanos Stathopoulos, John Heidemann, and Deborah Estrin. A remote code update mechanism for wireless sensor networks. Technical Report CENS-TR-30, University of California, Los Angeles, Center for Embedded Networked Sensing, November 2003.
45. Robert Szewczyk, Eric Osterweil, Joseph Polastre, Michael Hamilton, Alan Mainwaring, and Deborah Estrin. Application driven systems research: Habitat monitoring with sensor networks. *Communications of the ACM*, 47(6):34–40, June 2004.
46. Tijs van Dam and Koen Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the First ACM SenSys Conference*, pages 171–180, Los Angeles, California, USA, November 2003. ACM.

47. K. Whitehouse and D. Culler. Calibration as a Parameter Estimation Problem in Sensor Network. In *Proc. ACM Workshop on Sensor Networks and Applications*, Atlanta, GA, 2002.
48. Anthony D. Wood and John A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, October 2002.
49. Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. In *SIGMOD Record*, September 2002.
50. Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*, pages 1567–1576, New York, NY, USA, June 2002. USC/Information Sciences Institute, IEEE.
51. Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated, adaptive sleeping for wireless sensor networks. *ACM/IEEE Transactions on Networking*, 12(3):493–506, June 2004. A preprint of this paper was available as ISI-TR-2003-567.
52. Feng Zhao, Jaewon Shin, and James Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine*, 19(2):61–72, March 2002.