An Energy-Efficient Surveillance System for Wireless Sensor Networks

Tian He, Sudha Krishnamurthy, Jack Stankovic, Tarek Abdelzaher,
Ting Yan, Liqian Luo, Ryan Dickey, Radu Stoleru
Technical Report
Department of Computer Science
University of Virginia, Charlottesville, VA 22903

Abstract

The miniaturization of sensor devices, combined with their ability to gather information about the physical environment in which they are deployed, makes sensor nodes well-suited for unmanned reconnaissance missions. However, the software support required to use a wireless sensor network for this purpose is still lacking. In this paper, we describe a complete framework we have designed that provides the software support necessary for using wireless sensor networks for reconnaissance. The primary goal of this framework is to enable a sensor network to track objects in an energy-efficient and stealthy manner, with minimal false alarms. The framework minimizes false alarms by aggregating the reports from a group of neighboring sensor nodes. It achieves energy-efficiency and stealthiness by placing the sensor nodes in a low power consuming state in the absence of external events, and by minimizing the number of messages transmitted in that state. We have evaluated our implementation by deploying 70 nodes running our software in an open field. Our results show that for the configuration we used, the number of false alarms reduces to zero when the degree of in-network aggregation is 3. Our power management strategy is capable of extending the lifetime of the sensor nodes by up to 900% when the nodes operate for an hour per day. The number of messages in the idle state is nearly zero, confirming the stealthiness of our power management strategy.

1. Motivation

One of the key advantages of wireless sensor networks (WSN) is their ability to bridge the gap between the physical and logical world, by gathering certain useful information from the physical world and communicating that information to more powerful logical devices that can process that information. If the ability of the WSN is suitably harnessed, it is envisioned that the WSNs can reduce or in certain cases, eliminate the need for human effort in mundane information gathering in certain civilian and military applications. The applications range from simple detection to continuously monitoring a spectrum of entities, such as vehicles, weather, intruders, patients, and the habitat [3, 15]. There are several research efforts currently in progress that build hardware and software with the aim of bringing the above envisioned use of WSNs closer to reality. In this paper, we describe one such effort that involves a software architecture we have designed to enable the use of WSNs for special reconnaissance and surveillance in hostile areas.

Special Reconnaissance (SR) involves deploying specially trained forces (called Special Operations Forces (SOF) in military parlance), in hostile or politically sensitive areas of the world. Their mission is to conduct surveillance to acquire or verify information concerning the capabilities, intentions, and activities of an actual or potential enemy. It may also involve securing data about a particular region of interest. The SR teams provide the strategic and operational intelligence that will aid in military planning. Thus, they often serve as the 'eyes and ears' of unconventional warfare, direct action, and counter-terrorism operations. On account of this, their missions often carry an exceptionally high degree of physical risk.

Special operations units have traditionally relied on special reconnaissance personnel, manned aircraft, and satellites for surveillance and reconnaissance [13]. However, these traditional information providers are often too few in number and do not have sufficient real-time capability. Besides, they each have their own individual shortcomings. For example, human personnel are limited in how they can communicate what they observe. They do not have the means to relay back visual imagery, which is a more effective way of communicating information. Their resources have to be frequently replenished and such resupply missions often involve a high element of risk. In addition, their communications equipment are susceptible to jamming and interception. While current satellite communications systems are more secure and difficult to intercept, they tend to be bulky.

Some of the above problems are alleviated by the use of the manned aircrafts and unmanned aerial vehicles. However, operational security becomes more difficult when surveillance aircraft loiter along a hostile border. Moreover, while aerial intelligence can detect presence or absence of large targets on ground, it has to be supplemented with ground intelligence when more detailed information, such as the position or direction of a moving target, is desired. Hence, a miniaturized communications system with better range and effectiveness is desired.

Timely and relevant intelligence is crucial to the success of current day special operations. As a result, surveillance, reconnaissance, and communication assets that deliver near-real-time, full motion imagery for extended periods of time, regardless of the weather conditions, will be required. They will also need communication systems that are secure, dependable, have a low probability-of-intercept, and which extend beyond traditional line-of-sight capabilities. Due to the dynamic nature of modern warfare, the ability to re-program missions and receive corrections from a controller site is also desirable. This has led to the idea of instrumenting militarized zones with distributed sensors for surveillance. Unattended ground sensors (e.g. REMBASS [17]) that can detect, classify, and determine the direction of motion of intruding vehicles and personnel are already in use currently. However, these systems require manual deployment and therefore, endanger human lives. Moreover, they can operate unattended only for 30 days, which may be inadequate for long-term surveillance missions.

In this paper, we describe a software framework that we have developed, which enables the use of WSN for reconnaissance. The primary goal of the framework is to support the ability to track the position of moving targets in an energy-efficient and stealthy manner. Position estimates reported by individual sensor nodes are aggregated in the network and relayed to a video capture device, which then captures the images of the region of interest and transmits them to the base station. The in-network aggregation is based on the spatial and temporal correlation between the individual sensor reports and helps reduce false alarms during tracking. Our framework also has the ability to conserve the power of the deployed sensors when the network is idle. This is done by selecting a small subset of nodes as sentries to detect new events, while allowing the remaining nodes to remain in a low power consuming state. We have built a prototype to demonstrate the capabilities of our software and experimentally evaluated its performance on the field by deploying 70 sensor nodes running our software. Our experimental results show that the probability of false alarms observed reaches a minimum value for certain degrees of aggregation. For our network configuration, we found the degree of aggregation at which that minimum is achieved to be 3. The experimental results we obtained also show that the lifetime extension achieved using our sentry-based power conservation algorithm increases with a smaller duty cycle. With 5% of the deployed nodes serving as sentries and the non-sentries operating at a duty cycle of an hour per day, our algorithm extends the lifetime of a sensor network by up to 900 %.

The remainder of this paper is organized as follows. Section 2 describes the requirements of a typical ground surveillance application. In Section 3, we describe the system setup and hardware components. In Section 4, we provide an overview of our software framework. In Section 5, we elaborate how the individual components of the framework contribute to energy-efficient tracking. We present experimental results in Section 6, and summarize the lessons learned from our experience in Section 7.

2. Application Requirements

The design choices of our software framework are motivated by the requirements of a typical ground surveillance application. The general objective of such an application is to alert the military command and control unit in advance to the occurrence of events of interest in hostile regions. The event of interest for our work is the presence of a moving vehicle or an intruder carrying magnetic objects in the deployed region. The deployed sensor devices must have the ability to detect and track vehicles in the region of interest. Successful detection and tracking requires that the application obtain the current position of a vehicle with acceptable precision and confidence, for as long as the vehicle moves about in the sensor network's field of perception. As and when the information is obtained, it has to be reported to the base station with acceptable latency. The base station then forwards the information to the command and control unit. Additionally, the reports are used to turn on a video capture device, which relays back visual information about the target to the command unit.

The mission of a surveillance application typically lasts from a few days to several months. Due to the confidential nature of the mission and the inaccessibility of the hostile territory, it may not be possible to manually replenish the energy of the power-constrained sensor devices during the course of the mission. Hence, the application requires power management schemes that can extend the lifetime of the sensor devices, so that they remain available for the duration of the mission. One way to achieve that is by conserving the energy of the sensor nodes during the idle periods.

Maintaining a low probability of false alarms is another important requirement, on account of two reasons. First, false alarms may result in inappropriate decisions being made by the command and control unit. Second, power-constrained components, such as the sensor nodes and camera device, are turned on as a result of false alarms. That causes unnecessary power dissipation and reduction in lifetimes. The

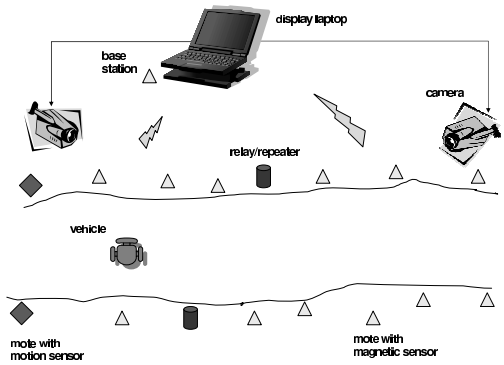

Figure 1. Sensor network deployment

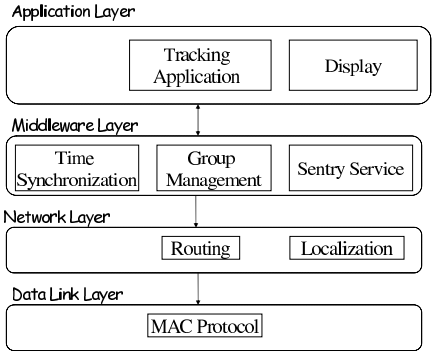

Figure 2. Software framework

false alarm probability, precision in the location estimate, confidence in the detection, and the latency in reporting an event are some of the metrics that we use to evaluate the performance of our software framework.

3. System Description

Figure 1 shows the deployment of sensor nodes for a typical ground surveillance operation. We deployed 70 tiny sensor nodes, called motes [12], along a 500 feet long perimeter in a grassy field. There are several classes of motes; the class of motes we used is called MICA2. The command and control unit consists of a base station, which is a mote that is attached to a portable device, such as a laptop. The laptop is the ultimate destination of the surveillance information and is mainly used for visualization. Each of the motes is equipped with a Chipcon radio operating at a frequency of 433 MHz. While this radio is sufficient to allow the motes deployed in the field to communicate with each other, it is not capable of long-range communication ($>$ 100 ft). Therefore, we assume that in a real system where the base station may be deployed several hundreds of feet away from the sensor field, devices capable of long-range communication, such as repeaters, will be deployed as gateways to assist the sensors and camera devices to relay back information from the motes in the field to the base station.

Each mote is also equipped with a sensor board that has magnetic, acoustic, and photo sensors on it. While the different sensors make it possible for a mote to detect different kinds of targets, only the magnetic sensors are relevant to the application described in this paper. We use the HMC1002 dual-axis magnetometers from Honeywell [11]. These magnetic sensors detect the magnetic field generated by the movement of vehicles and magnetic objects. They have an omni-directional field of view and are therefore, less sensitive to orientation. They have a resolution of 27 μGauss and their sensing range varies with the size of the magnetic object they are sensing. From our experiments, we found that these sensors can sense a small piece of magnet at a distance of approximately 1 ft and passenger vehicles moving at a speed of 5-10 mph at a distance of approximately 8-10 ft.

4. Overview of the Software Framework

The key contribution of this work is a software framework that enables the use of wireless sensor networks for energy-efficient tracking and detection of events. Such a framework is useful for surveillance applications, such as the one outlined in Section 2. The framework we have designed is organized into a layered architecture comprising of higher-level services and lower-level components, as shown in Figure 2. The framework is implemented on top of TinyOS [10]. We first provide an overview of the different software components we have designed and then follow that with a detailed discussion of the role played by those components in the context of a tracking and surveillance application.

Time synchronization, localization, and routing comprise the lower-level components and form the basis for implementing the higher-level services, such as aggregation and power management. Time synchronization and localization are important for a surveillance application, because the collaborative detection and tracking process relies on the spatio-temporal correlation between the tracking reports sent by multiple sensor nodes. The time synchronization module is responsible for synchronizing the local clocks of the sensor nodes with the clock of the base station. The localization module is responsible for ensuring that each node is aware of its location. We currently use a simple localization scheme, which statically assigns nodes their location at the time they are programmed. A static localization scheme is restrictive in that it does not estimate the absolute geograph-

ical coordinates of the event. We chose a static scheme, because such a strategy is sufficient to track the position of a vehicle relative to the sensor field. However, in a real system, such as a battlefield in which it is important to track the absolute geographical coordinates of the hostile tanks, the static scheme can be replaced with dynamic localization schemes (e.g. [7]). The routing component establishes routes through which the nodes exchange information with each other and the base station. All the nodes are linked together by a spanning tree rooted at the base station.

Power management and collaborative detection are the two key higher-level services provided by our framework. The sentry service component is responsible for power management, while the group management component is responsible for collaborative detection and tracking of events. The sentry service conserves energy of the sensor network by selecting a subset of nodes as *sentries*, to watch for events. The remaining nodes are allowed to remain in a low-power state until an event occurs. When an event occurs, the sentries awaken the other nodes in the region and the group management component dynamically organizes the nodes into groups in order to enable collaborative tracking. Together, these two components are responsible for energy-efficient event tracking, which is the primary goal of our framework.

The above framework can be considered as a library that a sensor network application that requires underlying support for energy-efficient tracking can link to. All the deployed nodes are programmed to run the application. Our framework supports the ability to reprogram the nodes with new configuration parameters dynamically. This eliminates the need to download the application code on all the nodes each time the configuration is modified. At the application layer we also have a display module, which we use primarily for visualization and debugging purposes. Optionally, the display software also has the logic to filter out any residual false alarms that have not been filtered out in the network. The display module runs on a portable device and works in conjunction with the base station. It is not part of the software that runs on each node. We now elaborate how the individual components of the above framework interact with each other in the context of a typical tracking application, in order to provide energy-efficient tracking.

5. Time-Driven Tracking Cycle

The sensor nodes prepare for tracking by going through an initialization process. This process is used to synchronize the sensor nodes, set up communication routes, and configure the nodes with the correct control parameters. The initialization process proceeds in a sequence of phases and the transition between phases is time-driven, as shown in Figure 3. Phases 1 through 5 comprise the initialization process, and the nodes are not involved in tracking while the initialization is in progress. At the end of phase 5, the nodes begin the power management and tracking activity. After performing this activity for a certain duration of time, they begin a new tracking cycle. The duration of each phase is a control parameter that can be dynamically configured by the base station. The cyclic process is important for several reasons. First, it allows the nodes to periodically synchronize their clocks to avoid significant clock drifts. Second, it allows the command and control unit to reconfigure the nodes with new control parameters, if required. Third, since node failures may occur anytime during a cycle, a new tracking cycle gives the remaining nodes an opportunity to repair old routes and discover new routes. Finally, a new tracking cycle allows the sentry responsibility to be rotated among other nodes in the network in order to achieve uniform energy dissipation across the network. We now discuss the activities occurring during each phase of the tracking cycle in more detail.

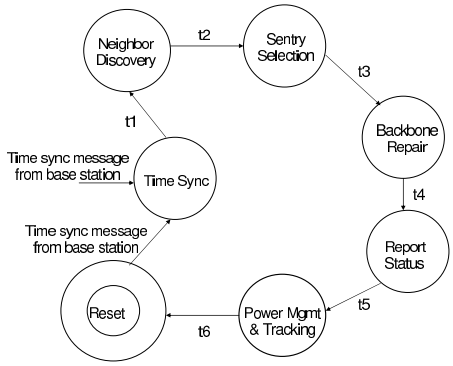

Figure 3. Time-driven phase transition

5.1. Time Synchronization

Each round of initialization begins with the time synchronization phase. Different time synchronization schemes have been proposed for a wireless sensor network. These schemes differ in the overhead incurred for achieving the synchronization, and in the precision, scope, and lifetime they are able to deliver. GPS-based schemes typically achieve persistent synchronization with a precision of about 200 ns. However, GPS devices are expensive and bulky. The post-facto scheme proposed in [5] avoids the need for nodes to be synchronized prior to an event. Instead, it broadcasts a third-party synchronization pulse to all the nodes in a region after an event occurs. The nodes that detected an event use this pulse as a reference to normalize their detection time. The problem with this lazy synchronization scheme is that the

nodes have to regularly wait for a pulse in order to report an event. Such delays are undesirable for reconnaissance applications that require events to be reported in near real-time. The reference broadcast scheme (RBS) proposed in [6] does away with the third-party synchronization pulse, and instead maintains information relating the phase and frequency of each pair of clocks in the neighborhood of a node. The relation is then used to perform time conversion when comparing the timestamps of two different nodes. While RBS achieves a precision of about 1 μs, the message overhead in maintaining the neighborhood information is high and may not be energy-efficient in large systems.

Fine-grained clock precision is not a primary requirement for our surveillance application. However, it requires energy efficiency and low reporting latency. Therefore, we use a scheme that synchronizes the nodes prior to the occurrence of an event, using a synchronization pulse broadcast by the base station at the beginning of each initialization cycle. Along with the value of the local clock, the base station piggybacks the values of all the dynamically configurable control parameters in the synchronization message. Since the underlying MAC layer we use does not guarantee reliable delivery, the base station retransmits the synchronization message multiple times. At this stage of execution, there are no well-defined routes setup in the network. So the synchronization message is propagated across the network through flooding. However, full-scale flooding is not energy efficient. Instead, we use a restricted form of flooding that allows a node to accept the synchronization messages forwarded only by those nodes that are within a certain neighborhood boundary. Upon accepting a synchronization message, the node adjusts its local clock and broadcasts the message to the other nodes. The propagation delay of the synchronization pulse results in the pulse arriving at different times at different nodes. That could result in an imprecision among the clocks. Our scheme currently does not account for this delay, because the precision we currently obtain is within the threshold required by our application.

5.1.1. Spanning Tree Creation. While the primary purpose of the synchronization message is to coordinate the clocks of the sensor nodes, it also serves as an exploratory message for nodes to setup routes to the base station, like the technique used by directed diffusion [14]. These routes provide an energy-efficient alternative to flooding and enable the nodes to communicate with each other and with the base station. The route that is set up during the propagation of the time synchronization message is essentially a spanning tree rooted at the base station. Initially we used an algorithm in which a node selected the first node from which it received the synchronization message as its parent in the spanning tree. However, we found that if nodes chose their parents without considering the distance separating them, it resulted in asymmetric communication. To solve this problem, we modified our strategy to allow a node to accept the synchronization messages forwarded by only those nodes that 1) are within a certain neighborhood boundary, and 2) are fewer hops away from the base station than itself. A node discards the message if either of these conditions is not true. Among the synchronization messages that a node accepts, it chooses the first node that satisfies the above two conditions as its a parent. The size of the neighborhood boundary is chosen to reduce asymmetric communication, and its value can be configured dynamically from the base station, as we next explain.

5.1.2. Dynamic Reconfiguration. In a typical deployment, it may be necessary to reprogram the sensor nodes in the field remotely from the command and control unit. In most cases, the reprogramming involves changing the values of some of the configuration parameters. Our system supports reprogrammability with the help of the time synchronization message. The base station piggybacks the values of the control parameters in the synchronization message and nodes adopt the new values when they accept the synchronization message. Such a strategy is energy-efficient, because it obviates the need to send separate messages to reprogram the nodes. Examples of control parameters that can be dynamically reconfigured include the duration of each phase shown in Figure 3 and the maximum distance (in hops) between a node and its parent in the spanning tree. In addition, the duration for which a node remains asleep and awake when power management is enabled, the sampling rate, and the degree of in-network aggregation are dynamically reconfigurable parameters. They will be referenced later in this paper.

5.2. Sentry-Based Power Management

After the nodes have synchronized their clocks, they begin the activities necessary for power management. In a typical reconnaissance application, periods of activity are punctuated by long periods of inactivity. Allowing the sensors to continuously monitor events in such an environment results in unnecessary energy dissipation. Current unattended ground sensors, like REMBASS [17], implement simple power management techniques that place the devices in a low power consuming idle state during periods of inactivity. When a target comes into detection range, the sensors detect a change in the ambient energy level and are activated. Such a strategy allows them to operate unattended for about 30 days. While this is a reasonably long lifetime, it may not be sufficient for long-lasting reconnaissance missions.

A Mica2 mote operates on a pair of batteries that approximately supply 2200 mAh at 3V. Different activities of the mote consume different amounts of energy. Studies have

shown that a Mica mote expends 1.25 nAh of power, on an average, when it is idling with its radio turned on for 1 millisecond, while communication operations typically consume higher power [15]. The baseline lifetime of a mote depends on the power consumed during periods of inactivity. Our experimental results showed that a Mica2 mote running an application linked to our software framework survives only for 5 days if its radio remains powered on all the time at the highest power level (10 dBm). Hence, our goal was to extend the lifetime of the network by minimizing the energy consumption during the idle periods. We achieve that using the sentry service component. in Section 6.2.2 shows,

The sentry service leverages the inherent redundancy in a densely deployed sensor network, and selects a subset of nodes as sentries to monitor for activity, while placing the remaining nodes in a low power-consuming snooze state. The sentries awaken the other nodes in the network when an event occurs. The sentry responsibility is rotated in each new tracking cycle, so that the energy dissipation is uniform across the network. The sentry service is responsible for selecting an appropriate number of sentries in each cycle, and ensuring that there is sufficient communication and sensing coverage. Allowing all the sensors in the network to serve as sentries to continuously poll for an event reduces both the latency to detect an event and the likelihood of false alarms. However, it results in higher energy dissipation. On the other hand, if too few sensors are used to monitor the occurrence of an event, the resulting coverage may be insufficient to detect the event in a timely manner. Thus, the sentry selection decisions involve a tradeoff between the amount of lifetime extension possible, and the latency as well as accuracy in detecting an event. We now present our sentry-based power management (SBPM) algorithm.

5.2.1. Neighbor Discovery. After the time synchronization phase, the nodes make a transition to Phase 2, which is the neighbor discovery phase. The purpose of neighbor discovery is to gather information from all the nodes that are within a specific neighborhood radius, so that the information can be used to select sentries to provide adequate sensing coverage. Nodes discover their neighbors by broadcasting HELLO messages within a certain radius. The radius is chosen in such a way that there is at least one sentry within each sensing range. In the HELLO message, a sender sends its identifier, its status indicating whether it is a sentry or not, and the number of sentries that are currently covering it. The sender also identifies the sentry node it reports to, if it is covered by at least one sentry. This local information is used to build a neighborhood table at each node, and forms the basis for sentry selection in Phase 3.

5.2.2. Sentry Declaration. There are several ways to elect sentries. For example, LEACH is a clustering protocol that uses localized coordination to minimize energy dissipation in a sensor network [9]. Each cluster of nodes in LEACH is represented by a cluster-head, which plays a role similar to the sentries in our scheme. Cluster-head responsibility is rotated using a randomized timer and those nodes that have been a cluster-head for fewer number of times in the past are favored as cluster-heads in the new round. LEACH places an upper bound on the percentage of cluster-heads that can be elected in each round for a given network size.

As in LEACH, in our sentry selection scheme also the decision to become a sentry is made locally by each node, using the information gathered from its neighbors. A node decides to become a sentry, if it discovers that none of its neighbors is a sentry or is covered by a sentry. When a node decides to become a sentry, it advertises its intent by broadcasting a SENTRY_DECLARE message to its neighbors. Contention occurs when multiple nodes in the same neighborhood decide to become sentries at the same time. In order to reduce the collision probability, each node uses a random backoff delay to transmit the SENTRY_DECLARE message. If a node receives a SENTRY_DECLARE message from one of its neighbors during the backoff period, it updates its neighborhood table and cancels any pending outgoing SENTRY_DECLARE messages. It then re-evaluates its decision to become a sentry based on the updated neighborhood information. If the node finds that it is still necessary for it to become a sentry, it repeats the sentry declaration process described above.

The backoff delay of a node is an important parameter in the sentry selection process. It allows us to tune the tradeoffs between energy dissipation and sensing coverage. We consider different factors, such as the residual energy of a node and the desired sensing coverage, when choosing the backoff delay. The backoff delay of a node is inversely proportional to its residual energy. Thus, a node with higher residual energy has a greater likelihood of being selected as a sentry, thereby balancing the energy dissipation uniformly across the network. On the other hand, the backoff delay of a node is proportional to the number of neighbors that are not covered by a sentry. Thus, nodes in regions where there is insufficient sensing coverage are favored for being selected as sentries. The key feature of this sentry selection algorithm is that it provides an adaptive, self-configuring technique for choosing the sentries purely based on local information. However, the lack of global knowledge may result in a non-optimal number of sentries. One way to improve our scheme is by bounding the total number of sentries selected in each round based on the network size and density, as done in LEACH.

5.2.3. Sentry Backbone Completion. The focus of the selection algorithm in Phase 3 is to choose sentries primarily to provide sufficient sensing coverage for the entire network. When power management is enabled, the sentries provide

the only means to route messages between the sensor field and the base station, because the sentries are the only nodes that remain awake. However, since the linkage factor was not considered during the sentry selection in Phase 3, the set of sentries selected may not be able to link every node to the base station. The purpose of Phase 4 is to select additional sentries, if required, to bridge that communication gap. This is done by designating all the parent nodes in the spanning tree that was created at the end of the time synchronization phase, as sentries. Thus the sentry nodes chosen in Phase 3, along with the parent nodes in the spanning tree, together provide sensing coverage and form the routing backbone that is used when power management is enabled. After the routing backbone is finalized, all the nodes use the backbone to report their status to the base station in Phase 5. The base station forwards those reports to the display module, which can then be used to visualize the routes and detect any failed nodes.

5.2.4. Power Management. The selection of sentries sets the stage for the power management phase. In this phase, the non-sentry nodes alternate between sleep and wakeup states. A node in the sleep state conserves power by disabling all processing, including those that are related to communication and sensing. We now examine different ways to control the sleep-wakeup cycle.

In the first approach, which we call *proactive* control, a non-sentry node stays awake until it receives a beacon from its sentry node, signalling the non-sentry node to sleep for a certain duration of time. Upon receiving the sleep beacon, the non-sentry node makes a transition to the sleep state and remains in that state for the specified amount of time. It wakes up when the timer expires and repeats the process by waiting for the next sleep beacon. Since neighboring non-sentry nodes are likely to receive the same sleep beacon, their sleep-wakeup cycle proceeds in a lock-step fashion. This regular synchronization of the non-sentry nodes with their respective sentries is beneficial in two ways. First, it allows multiple nodes to receive the same beacon, and obviates the need to send out individual sleep beacons to put each non-sentry node to sleep. This reduces the message overhead. Second, since nodes in a neighborhood are all awake at the same time, the correlated sleep-wakeup cycle helps to improve the tracking efficiency.

The second approach to control the sleep-wakeup cycle is called the *reactive* control. In this approach, the sentries are not required to send out explicit beacons to put the non-sentry nodes to sleep. Instead, the transition between sleep and wakeup states is timer-driven. Each non-sentry node remains awake for *awakeDuration* amount of time and then sleeps for *sleepDuration* amount of time. A node breaks out of this cycle and remains awake for a longer duration only on demand. This demand arises when either the connectivity is poor and additional nodes need to be awakened to fill the communication holes, as done in ASCENT [4], or additional sensing coverage is required to detect an event, as in our case. If such a need arises, a sentry sends out beacon messages to awaken its neighboring non-sentries.

The reactive scheme is more stealthy compared to the proactive scheme, because no unnecessary beacons are sent unless an event occurs. Hence, the reactive approach is more appropriate for a surveillance application. However, since the non-sentries do not periodically synchronize their clocks with the clocks of their sentries, the clocks of the non-sentry nodes may drift in course of time. As a result, neighboring non-sentry nodes may no longer have a sleep-wakeup cycle that is strictly in lock-step. Due to the reduced coordination, the non-sentry nodes may be slower to respond when an event initially appears. Furthermore, since a sentry no longer knows for certain which of its neighbors are awake, it has to retransmit the awake beacon multiple times in order to improve the chances of awakening as many of its neighbors as possible when an event occurs. We compare the message overhead between the proactive and reactive schemes in Section 6.2.1.

5.3. Event Tracking and Reporting

After the sentry backbone has been created and power management is enabled, the nodes are ready for tracking. A simple way to track events is by allowing each node that has sensed an event to report its location and other relevant information about the event to the base station. The base station can then filter out the false alarms and infer the location of the event from the genuine reports. The advantage of this approach is that it allows all of the complex processing of the sensor readings to be deferred to the more powerful base station. However, the main drawback is that, if the nodes are densely deployed, then multiple nodes may sense the event at the same time and send their individual reports to the base station. This results in higher traffic and wasteful expenditure of energy, which can be reduced by aggregating multiple reports about the same event and sending a digest, instead of the individual reports to the base station. In-network aggregation has been addressed in other related work. For example, the LEACH clustering protocol that was referenced earlier, organizes the nodes into clusters. Each cluster is represented by a head, which is responsible for fusing data within its cluster. The framework we have designed also performs in-network aggregation by organizing the nodes into groups. However, unlike LEACH, the groups in our work are transient and are formed only in response to an external event. A group represents an event uniquely and exists only as long as the event is in the scope of the sensor field. We now describe the key activities of the group management component, which is responsible for aggregation.

A more detailed description of this component is provided in [1].

5.3.1. Event Detection. Tracking an event involves continuously monitoring the current position of the event. For simplicity, we assume that the location of the event is described by a two-dimensional coordinate, although it would be simple to extend this to other location attributes. In order to help a node detect an event of interest, the node is programmed with the event's signature. A signature is an attribute-based name for an event [16], that succinctly describes the key attributes of the event and enables a sensor node to distinguish between different types of events. For instance, in order to detect a moving vehicle, a node can be supplied with a signature of the type (`MagSensorReading` $\geq$ `VEHICLE_THRESHOLD`), where the value of `VEHICLE_THRESHOLD` depends on the vehicle being detected. This allows the sensor node to signal the presence of a moving vehicle, if its magnetic sensor reading is above the specified threshold. Similarly, a signature of the type (`MagSensorReading` $\geq$ `VEHICLE_THRESHOLD or MotionSensorReading` $\geq$ `HUMAN_THRESHOLD`) allows a node to detect the presence of either a moving vehicle or a moving person. The more precise the signature, the more accurately it allows a node to distinguish between different types of events.

5.3.2. Group Formation. When a node or a set of nodes detect an event, the group management component reacts by creating a group. All the nodes that detect the same event join the same group. A group is an abstraction of a physical event and its formation is transparent to the end user. The group management component establishes a one-one mapping between a group and a physical event and maintains the mapping as long as the nodes continue to detect the event. Each group is represented by a *leader*, which plays an important role in aggregation. There are different ways to choose a leader. For example, the node that first detects an event can choose the sentry to which it reports as the leader of the group. However, this requires that the group management be coupled with the sentry service. Another option is to choose the node located at the center of the group as the leader, so that the average distance over which nodes transmit their reports is small. However, the delay in identifying the node at the center could result in a late report. In a typical surveillance application, the target moves quickly and the tracking report has to arrive at the base station in near real-time. So any overhead in the group formation and leader election has to be minimal and transparent to the user. Therefore, we use a simple leader election in which the first node that detects an event elects itself as the leader of its group.

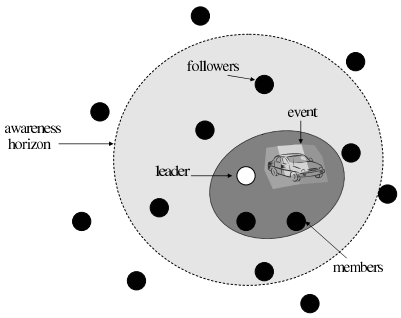

Figure 4. Group migration

5.3.3. In-Network Aggregation. The formation of a group is the first step toward in-network aggregation. After the node has joined a group upon detecting an event, it periodically reports its position to the group leader, as long as it continues to detect the event. Sending the reports to a local leader instead of a remote base station contributes to energy-efficient tracking. The leader records each report it receives in a database. The database stores the most recent report from each member in a group. Reports that are older than a certain threshold period of time, t_r, are purged from the database, in order to better estimate the current position of the moving vehicle. More the number of reports about an event, the higher is the confidence in the detection. We define the confidence level of an event detection as the number of distinct nodes that have reported the event in the last t_r units of time. When the confidence level of detecting an event is at least as high as the threshold required by the application, the leader aggregates the reports from all of its group members and reports the aggregated position to the base station. The confidence threshold can be tuned to minimize the number of false alarms, as we show in Section 6.1.1. There are different choices for the aggregation function. One option is to take a simple average of the two-dimensional coordinates reported by the group members. Alternatively, the leader can use a weighted average function to estimate the location of the event.

5.3.4. Group Migration. One of the key challenges of the group management component is to ensure that there is exactly one sensor group for each external event that is detected. Maintaining the unique mapping between a sensor group and an external event is trivial, if the events are stationary. However, in our application, while the sensor nodes are stationary, the events are mobile. Hence, different sets of nodes may detect the same event in different locations and at different points in time. When nodes in a group stop detect-

ing the event, they resign from the group. On the other hand, new nodes that detect the event join the group as the event propagates through the sensor field. Thus, the group management component must be able to maintain the unique mapping between a group and an event as long as the event lasts, while allowing the group membership to be dynamic. That requires the ability to distinguish between an event that has been previously detected and an event that has newly occurred. The group management layer achieves this using the following approach, which is based on the assumption that a typical tracking event progresses continuously in space and time.

Upon detecting an event, the leader of a group advertises the presence of the event to all the nodes within a certain distance, called the *awareness horizon*. The awareness horizon is chosen to encompass nodes that are within the communication range of the leader and represents the nodes that are spatially correlated. Since the sensing range is usually smaller than the communication range, the awareness horizon includes the nodes that are within the sensing range of the leader. Some of these nodes may sense the same event that the leader senses, and therefore, they are members of the group that the leader belongs to. The remaining nodes in the awareness horizon that cannot sense the event are called the *followers*. Figure 4 shows the distinction between group members and followers.

A node becomes a follower when it receives the periodic advertisement broadcast by a leader. Upon receiving the broadcast, the node sets a timer. If the node senses the event before the timer expires, it makes the transition from being a follower to a group member. A node is allowed to spawn a new group upon sensing an event, only if it is neither a follower nor already a member of the group. This restriction is based on the assumption that nodes that are spatially correlated are likely to detect the same event at nearly the same event. Hence, by restricting the formation of a new group to nodes outside the current awareness horizon, we prevent multiple groups from being formed for the same event. As the event propagates through the sensor field, more nodes are awakened, and the awareness horizon shifts. When the target propagates beyond the sensing range of the current leader, the leadership of the group is handed over to a node in the new awareness horizon. Finally, when the nodes no longer detect the event, the group is disbanded and the nodes resume their normal sleep-wakeup cycle.

6. Performance Evaluation

We now present experimental results that evaluate the performance of the software framework described in the previous section. We obtained most of the experimental results through an actual deployment of Mica2 motes in a field, using the setup described in Section 3. We first analyze the performance of event detection and tracking and then evaluate the sentry service and power management features of our framework.

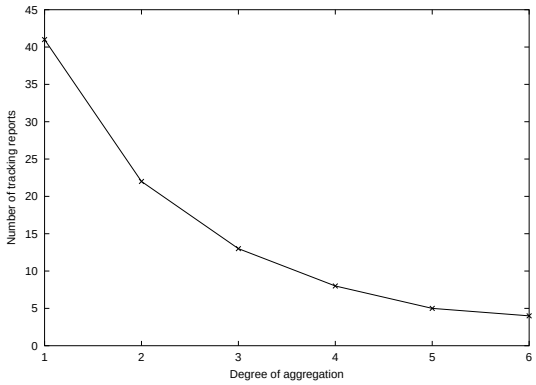

Figure 5. Impact of degree of aggregation on the message overhead

6.1. Evaluation of In-Network Aggregation

In our experimental setup, we deployed 70 Mica2 motes along two sides of a road at a distance of 4-5 ft from each other. They were deployed densely in order to improve the degree of aggregation. The non-sentry motes were programmed to sleep for 2 seconds out of every 3 seconds, when the power management was enabled.

Our goal was to track the movement of a car being driven along the stretch of road and study the impact of the degree of in-network aggregation (DOA) on the tracking performance. The degree of aggregation is the minimum number of reports about an event that a leader of a group waits to receive from its group members, before reporting the event's location to the base station. In our implementation, the value of the DOA is dynamically configurable from the base station. We were interested in studying the impact of the degree of aggregation on the following metrics:

- the number of tracking reports received by the base station,
- the number of false alarms generated, and
- the latency in reporting an event.

We classify false alarms into *false negatives* and *false positives*. A false positive occurs when a group of motes report the presence of the moving car in their neighborhood, when in reality, the car is not in their vicinity. In other words, if the position of the car as it appears on the display does not correspond to its actual position in the field, we treat it as a false positive. A false negative occurs, if the

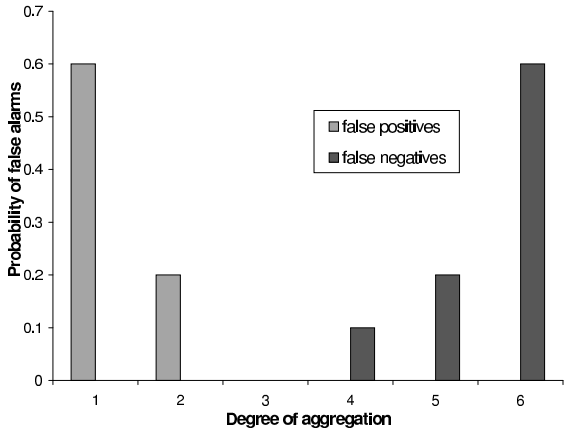

Figure 6. Impact of degree of aggregation on false alarms

base station does not receive any reports of the car, although in reality, there is a car moving though the sensor field. In other words, if the car never appears on the display as it moves from one end of the sensor field to the other, we treat it as a false negative. It is important to emphasize that we do not consider a delayed report as a false negative.

In our tracking experiments we drove a car at a speed varying between 5-10 mph. We varied the degree of aggregation from 1 to 6 and repeated the tracking experiment for each value of DOA ten times. Figure 5 shows how the number of the tracking reports received by the base station varies with the DOA. From the figure, we see that when the value of DOA increases from 1 to 2, the number of tracking reports reduces by almost 50%. As the value of DOA increases even further, we observe that there is a steady drop in the number of tracking reports generated. These results verify the fact that the in-network aggregation, resulting from organizing the sensor nodes into groups, significantly reduces the message overhead during tracking.

6.1.1. Impact of Aggregation on False Alarms. Our next experimental result shows how the degree of in-network aggregation affects the false alarms generated when tracking an event. We determined the probability of false alarms for each value of DOA by counting the number of false positives and false negatives we observed on the display during a set of 10 tracking rounds. Figure 6 shows how the probability of false positives and the probability of false negatives are each affected by the degree of aggregation. From Figure 6 we see that as the value of DOA increases from 1 to 6, the probability of false positives drops from 0.6 to 0, while the probability of false negatives increases from 0 to 0.6. These results can be explained as follows.

When the DOA = 1, the leader of a group reports the event to the base station, as soon as at least one member of the group detects the event. In an ideal scenario in which the sensing is perfect, even a single sensor reading should generate a high level of confidence. However, in practice, the sensor boards are sometimes faulty. This could result in an event being reported when it is not actually present. Hence, a single sensor reading may not be very reliable. One way to improve the reliability in the event detection is to increase the redundancy, by either waiting for multiple reports from the same sensor node (temporal redundancy), or by waiting for reports from multiple neighboring sensor nodes (spatial redundancy). We chose to experiment with the latter option, because we assumed that the faults in the sensor boards are independently distributed. Therefore, the probability that multiple neighboring sensor nodes are simultaneously faulty is lower than the probability that a single sensor node is faulty. From Figure 6, we see that our assumption is validated. The figure shows that if the leader waits until at least 3 different sensor nodes have detected the event, before reporting the event to the base station, the number of false positives drops to 0.

However, if the sensing range and the density of deployment is not sufficiently high, it is harder to achieve a higher degree of aggregation. This results either in more false negatives, as shown in Figure 6, or in higher reporting latency. Figure 7 shows how the reporting latency increases with the degree of aggregation for a car moving at 5 mph through a sensor field where the nodes are deployed 4-5 ft apart. We define the reporting latency as the time elapsed from the instant at which the car enters the sensor field until the instant at which the base station receives the first *genuine* report about the location of the car. In addition to the density, the increase in the latency and false negatives depends on the sleep cycle of the sensor nodes and the speed of the moving vehicle. For our configuration, we found that we were able to reduce the latency and false negatives for higher degree of aggregation (DOA $\geq$ 4), by increasing the speed of the vehicle from about 5 mph to about 10 mph. However, increasing the speed beyond that value resulted in more false negatives. The reason is that when nodes are some distance apart, a higher speed allows the vehicle to be in the sensing range of more nodes at the same time. Hence, the vehicle can be detected even at a higher degree of aggregation. However, the sensors have a non-negligible reaction time, which further increases if the nodes are sleeping. Hence, if the speed is increased beyond a certain threshold, the vehicle may move past the sensing range of the nodes before they have a chance to react. That could result in more false negatives.

We must emphasize that the performance numbers we have presented above exhibit some degree of variance across different experimental runs and in different environments. Therefore, instead of using the above experimental results to

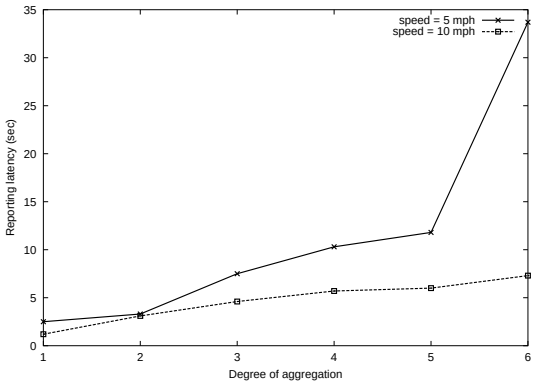

Figure 7. Impact of degree of aggregation on reporting latency

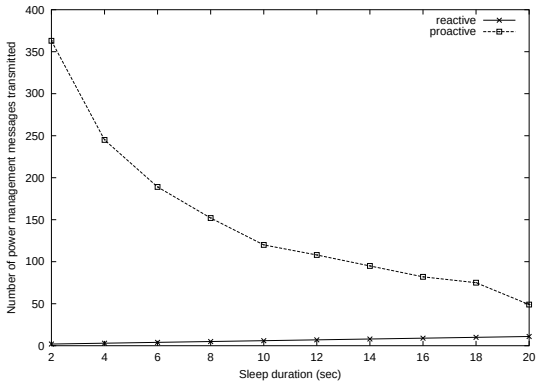

Figure 8. Message overhead due to power management

deduce absolute performance numbers, we use them to draw some general conclusions about choosing the degree of in-network aggregation. First, a higher DOA certainly helps reduce the message overhead and the number of false positives. However, if the density with which the sensor nodes are deployed is not sufficiently high, a higher degree of aggregation may adversely affect the tracking performance. This effect is more pronounced in the case of slow-moving events. Even if the nodes are densely packed and the events are fast-moving, it is harder to achieve a high degree of aggregation, if the motes sleep for a long duration and their sleep-wakeup cycles are not in lock-step. Thus, we see that the degree of aggregation represents a tradeoff between different parameters. The recommendation we follow based on our results is to choose a value of DOA that is large enough to maintain the probability of false negatives within a certain threshold. Our experiments show that a value of 2 or 3 for the degree of in-network aggregation is reasonable for our configuration. If this value is not large enough to maintain the false positives within the desired threshold, then we recommend using a second tier of filtering at the base station.

The above discussion motivates the need for an analytical model that captures the tradeoff between the key parameters, such as the degree of aggregation, density of node deployment, sleep duration, and the maximum probability of false alarms that a user can tolerate. Such a model can then be used to choose the appropriate degree of aggregation, when the values of the other parameters are known. Such a model is also valuable in estimating the probability of false alarms that a user can expect for a specific design and configuration.

6.2. Evaluation of Sentry Service

In this section, we analyze the key features of the sentry service component. We first analyze the stealthiness of the power management scheme, and then assess the extension in lifetime achieved for different sentry distributions and for different periods of the sleep-wakeup cycle of the non-sentries.

6.2.1. Stealthiness of Power Management Component. In Section 5.2.4, we compared and contrasted the proactive and reactive schemes for controlling the sleep-wakeup cycle of the non-sentry nodes when power management is enabled. The proactive scheme provides better responsiveness when an event occurs, at the cost of transmitting more messages in the absence of an event. In contrast, the reactive scheme provides better stealthiness during the idle periods, at the cost of retransmitting multiple messages in order to awaken the non-sentries when an event occurs. A sentry chooses the interval between successive retransmissions in such a way that the beacon transmission coincides with the wakeup period of the neighboring non-sentry nodes. We use the following equation to control the number of retransmissions of the awake beacon (n_r).

$$n_r = \frac{sleepDuration + awakeDuration}{awakeDuration + 1} \qquad (1)$$

A larger value of *awakeDuration* results in fewer retransmissions of the awake beacon when a sentry detects an event. However, if the motes are awake longer, more energy is consumed and therefore, the lifetime of the sensor network reduces.

Higher message overhead also translates to higher energy consumption. In order to compare the message overhead between the reactive and proactive schemes, we implemented both the schemes and conducted simulation experiments using the Nido simulator [18]. We simulated a simple scenario in which a tank moved across a sensor field in which 10 nodes capable of magnetic sensing were deployed. The duration of each simulation run was 600 seconds. The *awakeDuration* of the motes was fixed at 2 seconds for each run. Figure 8 compares the number of messages sent out by the

proactive and reactive schemes during the tracking phase when power management is enabled.

Figure 8 shows that the number of power management messages in the reactive scheme increases from 2 to 11 as the sleep duration increases from 2 seconds to 20 seconds. This is justified by Equation 1, which indicates that a longer sleep duration requires more retransmissions of the awake beacon, in order to increase the probability that it will be received by the non-sentry nodes. In contrast, the message overhead in the case of the proactive scheme reduces as the sleep duration increases. This is because the periodicity with which a sentry sends out the sleep beacon is equal to *sleepDuration + awakeDuration*. As the sleep duration increases, the sleep beacons are sent out less frequently, thereby reducing the message overhead.

The results in Figure 8 also show that the message overhead due to power management is significantly lower in the reactive scheme compared to its proactive counterpart. This suggests that the reactive scheme is more stealthy compared to the proactive scheme. While this is true for the 2 second awake period we have chosen, it may not be true for smaller values of *awakeDuration*. In our experiment, we chose a relatively high value of 2 seconds for *awakeDuration*, in order to compensate for the high rate of drift in the software timers in the current TinyOS implementation. If the timer drift is smaller in future implementations of TinyOS, we would choose a smaller awake duration for the motes, so that the overall energy consumption of the network can be reduced. However, a smaller value of *awakeDuration* would increase the message overhead for the reactive scheme. We have currently adopted the reactive scheme for our surveillance application, because it provides better stealthiness for the duration of the sleep-wakeup cycle we have chosen. However, an investigation into a hybrid scheme that combines the advantages of both the proactive and reactive schemes would be worthwhile to pursue as future work.

6.2.2. Power Savings. One of the main goals of the sentry service module is the extension of the lifetime of the sensor network. The sentry service extends the lifetime by conserving the energy consumption of the nodes when the network is idle. Non-sentry nodes alternate between sleep and wakeup states, and in Section 6.2.1, we justified our choice of a timer-driven, reactive approach to control the sleep-wakeup cycle. When a mote is in the sleep state, its radio is turned off, all of its I/O ports are configured appropriately to minimize the current consumption, the ADC module is turned off to disable any sampling, and the controller is placed in a power-save state. When the sleep timer expires, the controller is awakened by a timer interrupt, and all of the modules resume activity. The extent to which our power management approach increases the lifetime of a mote depends on the fraction of time the mote spends in the sleep state. We now use the current consumed in the sleep and wakeup states using the above power management scheme to predict how the expected lifetime of a sensor network varies with the fraction of sentries selected.

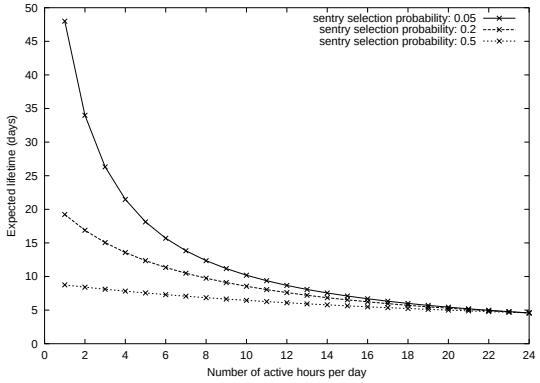

Figure 9. Expected lifetime of a sensor network using sentry-based power management

A Mica2 mote is powered by a pair of AA batteries, supplying a combined voltage of 3V. Assuming that a pair of batteries will supply 2200 mAh at 3V [15], we can estimate the lifetime of a mote, if we know the current consumed in the sleep and wakeup states and the duty cycle of the mote. The duty cycle of a mote is the number of hours per day it remains awake polling for events. Based on our measurements, we found that a Mica2 mote equipped with a magnetic sensor board and running our sentry-based power management software consumes 20 mA in the wakeup state. The wakeup current includes the current consumed by the magnetometer to sample at a rate of 10 samples per second. On the other hand, we measured the sleep current of the mote to vary between 50 μA to 130 μA, which results in a 99% reduction in the current consumption. We use a sleep current of 130 μA for the discussion in this section.

From the above data, we can determine the lifetime of a sensor network that uses our sentry-based power management scheme. The lifetime of a sensor network depends on the fraction of sentries selected and the fraction of time the non-sentry nodes remain awake. Let $P(s)$ denote the probability that a node is selected as a sentry, and $P(a)$ denote the probability that a non-sentry node is awake. The total current (C) consumed by a mote in the baseline case, when there are no events in the network, is given by Equation 2. The lifetime of the motes, L, is the ratio of the battery capacity to the total current consumed. Assuming a battery capacity of 2200 mAh, the lifetime of the motes in hours is simply $2200/C$.

$$C = P(s)*20+(1-P(s))*(P(a)*20+(1-P(a))*0.13) \quad (2)$$

Figure 9 uses the above equation to predict the expected lifetime of the motes for different percentages of their duty

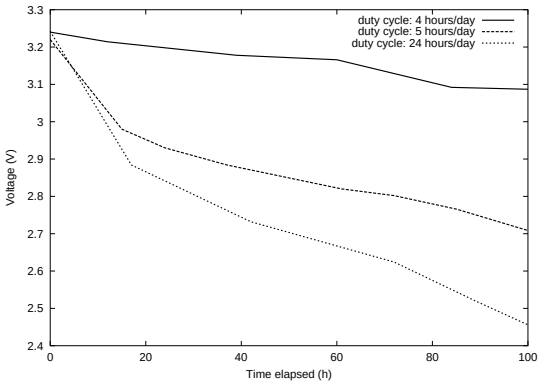

Figure 10. Impact of sleep duration on power consumption

cycle. A mote that is always asleep is expected to survive for 2 years, whereas a mote that is always awake (i.e. always remains a sentry), can survive only up to 5 days. The exponential curves show that the lifetime greatly improves when the duty cycle is low. For example, when the probability that a node is selected as a sentry is 0.5, and its duty cycle is reduced from 24 hours per day to one hour per day, its lifetime extends by nearly 100%. The graphs also show that the lifetime improves significantly when the probability that a node is selected as a sentry is low. For example, when the probability that a node is selected as a sentry is reduced to 0.05, and its duty cycle is reduced from 24 hours per day to one hour per day, its lifetime extends by nearly 900%. The probability of selecting a node as a sentry involves a tradeoff between the sensing coverage that can be achieved and the required network lifetime. A higher probability results in more sentries and provides better sensing coverage. However, it also reduces the lifetime of the network, as Figure 9 shows. In order to reduce the number of sentries without adversely affecting the sensing coverage, we can either choose magnetometers with higher sensing range or increase the density with which the nodes are deployed. For example, in our experiments we found that when the motes were placed at a distance of 8 ft from each other, the probability that a node was selected as a sentry was nearly 0.5. However, in a more dense deployment in which the motes were placed within a few inches from each other, the probability of selecting a node as a sentry dropped to 0.2. The reason is that a dense deployment results in a larger number of neighbors for each node. Therefore, a single sentry is able to cover more neighbors, and that gives fewer nodes a chance to elect themselves as a sentry.

In addition to predicting the lifetime of the network using a simple model, we also conducted experiments to compare the rate at which energy is dissipated for different duty cycles in an actual deployment. In each of our experiments we deployed 6 motes, all equipped with magnetic sensor boards, inside an office building. Sentry rotation occurred once every 4 hours. Since there is no direct way to measure the energy consumed by the motes, we used the voltage drop across the batteries supplying power to the motes as an indirect way to measure the energy dissipation. We measured the voltage for each mote at regular intervals over a period of 100 hours and found that the voltage drop was reasonably uniform across the motes. Figure 10 shows the voltage drop during the observation period for one of the 6 motes for different values of duty cycle. From the figure, we see that the battery voltage for a mote does not drop uniformly with time. One of the reasons for the non-uniform energy dissipation is the periodic rotation of the sentry responsibility. The voltage drop of a mote is higher during an interval in which it is serving as a sentry than when it is serving as a non-sentry, because the periodic sampling operation performed by a sentry consumes significant energy. The results also confirm that a higher duty cycle results in a higher energy dissipation. We see that when the mote is always awake, it loses most of its capacity within 100 hours (about 4 days). This reasonably matches the result in Figure 9, which predicted that a mote operating 100% of the time will last only 5 days.

The experimental results we obtained are promising in that they show that the sentry-based power management algorithm is adaptive and that it is successful in extending the lifetime of the sensor network. While our current sentry selection algorithm does not choose the minimal number of sentries, by knowing the lifetime of the mission in advance, we can choose the density of deployment and the duty cycle in such a way that the lifetime requirement can be met.

7. Lessons Learned

Our earlier work in the area of sensor networks mainly focused on designing individual protocols and adopted a simulation-based approach for evaluating them [1, 7, 8]. The work described in this paper is our first experience in building a complete system for using sensor networks for a practical application, and evaluating it through an actual deployment of motes. This practical experience has been valuable, because it has taught us that some of the simplified assumptions we made about the hardware platform and operating system in our earlier simulation-based approach do not hold well in practice. The lessons we learned have greatly impacted some of the design choices we had to make in building our system. We now share some of those experiences here with the intent that they will enable future system implementors to account for these issues from ground up, when they build systems for sensor networks.

Variance in sensor readings: In a simulation-based approach, all sensor devices of the same type generate

the same readings under identical conditions. However, in practice, we found that the same type of sensors are capable of generating different sensor readings under identical conditions. Such a phenomenon may occur because of differences in the way the devices are manufactured, and it is often hard to capture those differences in a simulator. The variance in the sensor readings can be accounted for at the very outset through software calibration of the sensors.

False alarms: A simulation-based approach assumes that sensors behave according to their specifications. However, in practice, sensors are capable of generating false alarms. False alarms can be suppressed by aggregating the readings of neighboring sensors and by using algorithms that filter noise.

Application-specific reliability: We found that the packet loss in the Mica2 platform can be as large as 20%. A well-known approach to counter message loss is to retransmit the message multiple times, in order to improve the probability of delivery. Such retransmissions can be initiated either in the lower layers of the protocol stack or at the application layer. Since retransmitting a message consumes significant energy, it is important that the messages are retransmitted selectively, based on application-specific knowledge. For instance, applications that transmit ephemeral sensor readings, such as the instantaneous temperature, may not require reliability. Lower layers, such as the MAC layer, often lack domain-specific knowledge. So implementing reliability guarantees in the lower layers makes it harder to provide application-specific reliability. Hence, for a system that strives to achieve energy efficiency, providing reliability guarantees at the application layer is a better option.

Race conditions: Contention occurs not only when different nodes try to transmit simultaneously, but also when different software modules on the same node transmit simultaneously. Due to the limited support of the operating system, the latter can lead to race conditions. Race conditions are another example of a phenomenon that we effectively ignored in our earlier simulation-based approach, but had to address when building the system we have described. Race conditions can be avoided, if the OS can make use of synchronization structures, like atomic blocks, in order to coordinate the shared resources among the contending modules.

Asymmetric channels: Another issue we had to address in an actual deployment of motes was to account for the effect of asymmetric channels. Asymmetric channels result in different reception rates along different directions between the same pair of nodes. Communication in low power devices, such as the motes, is asymmetric due to differences in hardware, signal attenuation, and residual battery capacity [2]. It is hard to capture the effects of asymmetric communication in a simulation. As a result, we had ignored the effects of asymmetric communication in our earlier simulation-based work. However, in practice, we were able to reduce the effect of asymmetric channels by restricting a node to communicate with only those neighbors that are well within its communication range.

Software timer drift: The drift in the software timers in TinyOS presents another practical issue, especially when nodes transit into sleep state. In order to compensate for the drift in the soft timers, we had to increase the duration for which a mote remains awake, and design appropriate strategies to control the sleep-wakeup cycle, as described in Section 6.2.1.

Built-in debug capability: Another practical challenge we were faced with in building a system with multiple components was the lack of appropriate tools for debugging the mote platform. More sophisticated debugging tools, such as SCALE [2], for assessing connectivity and studying interactions between modules will greatly ease the burden on the programmer in the future.

Acknowledgments: We thank Jonathan Hui and Bruce Krogh of Carnegie-Mellon University for designing and implementing the sentry selection algorithm. We also thank Gary Zhou, Yong Chen, Lin Gu, and Qing Cao for their help in testing and deploying the system.

References

[1] B. M. Blum, P. Nagaraddi, A. Wood, T. F. Abdelzaher, S. Son, and J. A. Stankovic. An Entity Maintenance and Connection Service for Sensor Networks. In *The First Intl. Conference on Mobile Systems, Applications, and Services (MobiSys)*, May 2003.

[2] A. Cerpa, N. Busek, and D. Estrin. SCALE: A Tool for Simple Connectivity Assessment in Lossy Environments. Technical Report 0021, CENS, September 2003.

[3] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat Monitoring: Application Driver for Wireless Communications Technology. In *Proc. of the 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, April 2001.

[4] A. Cerpa and D. Estrin. ASCENT: Adaptive Self-Confi guring Sensor Networks Topologies. In *Proc. of the IEEE Computer and Communications Societies (INFOCOM)*, June 2002.

[5] J. Elson and D. Estrin. Time Synchronization for Wireless Sensor Networks. In *Proc. of Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, April 2001.

[6] J. Elson and K. Romer. Wireless Sensor Networks: A New Regime for Time Synchronization. In *Proc. of the Workshop on Hot Topics in Networks (HotNets)*, October 2002.

[7] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-Free Localization Schemes in Large-Scale Sensor Networks. In *Proc. of the Intl. Conference on Mobile Computing and Networking (MOBICOM)*, September 2003.

[8] T. He, J. Stankovic, C. Lu, and T. Abdelzaher. SPEED: A Stateless Protocol for Real-Time Communication in Ad Hoc Sensor Networks. In *Proc. of International Conference on Distributed Computing Systems (ICDCS)*, May 2003.

[9] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Effi cient Communication Protocol for Wireless Microsensor Networks. In *Proc. of the Intl. Conference on System Sciences*, January 2000.

[10] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System Architecture Directions for Networked Sensors. In *Proc. of Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 93–104, 2000.

[11] Honeywell. *1- and 2-Axis Magnetic Sensors.* Available at `www.ssec.honeywell.com/magnetic/datasheets/hmc1001-2_1021-2.pdf`.

[12] M. Horton, D. E. Culler, K. Pister, J. Hill, R. Szewczyk, and A. Woo. MICA: The Commercialization of Microsensor Motes. *Sensors Online*, April 2002. www.sensorsmag.com/articles/0402/40.

[13] S. Howard. Special Operations Forces and Unmanned Aerial Vehicles: Sooner or Later? Master's thesis, School of Advanced Airpower Studies, June 1995. Available at `www.fas.org/irp/eprint/howard.htm`.

[14] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proc. of Mobile Computing and Networking (MOBICOM)*, pages 56–67, August 2000.

[15] A. Mainwaring, J. Polastre, R. Szewczyk, D. E. Culler, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *Proc. of the ACM Workshop on Sensor Networks and Application (WSNA)*, September 2002.

[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proc. of the ACM SIGCOMM*, August 2001.

[17] Remote Battlefi eld Sensor System. Available at `www.fas.org/man/dod-101/sys/land/rembass.htm`.

[18] University of California, Berkeley. *TOSSIM: A Simulator for TinyOS Networks*. Available at `webs.cs.berkeley.edu/tos/tinyos-1.x/doc/nido.pdf`.