Wireless Sensor Networks and Applications
SECTION V Applications
Y. Li, M. Thai and W. Wu (Eds.) pp. 659-700
©2005 Springer

Chapter 18

Maximizing the Lifetime of an Always-On Wireless Sensor Network Application: A Case Study

Santosh Kumar, Anish Arora, and Ten H. Lai
Department of Computer Science and Engineering
The Ohio State University, Columbus, OH 43210
E-mail: {kumars, anish, lai}@cse.ohio-state.edu

1 Introduction

Most applications of wireless sensor networks (WSNs) [2, 4, 10, 19, 22, 24, 25] require long term (several months or even years of) unattended operation from their networks. But, significant challenges still remain in achieving long-term unattended operation from large-scale wireless sensor networks. Critical among those are the problems of power management. An important distinction between wireless sensor networks and most existing systems is the tremendous gap between the energy available to a sensor node and that required for its long-term unattended operation. A typical sensor node such as ones from the Mica family [8, 9] can last 3-5 days on a pair of AA batteries in

its fully active mode. In real life applications, though, we would like a sensor network (comprised of these motes) to last at least a few months. The story is not very different with other platforms. The question then is: How can we fill this huge energy gap? The approaches and techniques applied to fill this energy gap are collectively referred to as *Power Management* in wireless sensor networks.

Always-On and Always-off Applications of WSN: Applications of WSNs are broadly divided into two categories — *always-on* and *always-off*. In an always-on application, it is necessary to monitor the environment continuously because the events of interest can occur at any time. In most of these applications, it is also required to notify a base station of an event as soon as it is observed so that retaliatory actions (or preventive measures) can be taken quickly. Examples of such applications are intrusion detection [2], where intruders can breach a protected region any time, shooter localization [24], where shooting by a sniper can occur anytime, and radiation detection [4], where terrorists can explode a simple radiological dispersion device at any time.

In an always-off application, it is not necessary to monitor the environment continuously; periodic monitoring is enough because the environment does not change very abruptly. Examples of such applications are habitat monitoring [22], where the environment of a bird nest does not change abruptly, and the monitoring of subgalacial bed formation [19], where the glaciers do not change their position or temperature abruptly.

Classification of Power Management Approaches:

An application faces the problem of power management if the active life of the sensor nodes comprising a WSN is less than the desired life of the network. In such a case, we need to find ways to extend the lifetime of the network. We divide the approaches for doing so in two major categories:

- *Fine-Grained Power Management:* This approach extends the active lifetime of individual sensor nodes by exploiting redundancy already existing in the network. For example, in an always-off application it is not necessary to sample the environment continuously. So, all the sensor nodes can be put to sleep, to wakeup only periodically for sampling the environment. As another example, in an always-on application, selected components of sensor nodes (such as the radio) can be put to sleep to be woken up either on demand or periodically, while still meeting the monitoring and notification requirements of the application. Fine-grained power management schemes do not require deploying any more sensor nodes than are absolutely necessary to meet the monitoring requirements of an application.

- *Coarse-grained Power Management:* In this approach, more sensor nodes are assumed to be deployed than are necessary to meet the monitoring and notification requirements of an application. The redundancy in the number of sensor nodes is then exploited to increase the network lifetime. Time is divided into intervals and in each interval, only that many sensor nodes are active as are necessary to provide the desired level of monitoring. The redundant nodes are put into deep

sleep by putting all their components (the processor, the radio, and all the sensors) to sleep. The nodes that remain active are sometimes called *sentries* [15].

These two approaches are complementary in the sense that we can first use fine-grained power management schemes to extend the active lifetime of individual sensor nodes by the maximum extent possible. If this extended active lifetime of the sensor nodes still falls short of the desired lifetime of the network, coarse-grained power management schemes can be used to extend the network lifetime further. Another alternative to using coarse-grained power management schemes (which will require deploying redundant sensor nodes) is to deploy redundant batteries per node or to use more expensive but higher capacity batteries. Which alternative to use (deploying redundant sensor nodes or deploying redundant batteries) will require a careful cost-benefit analysis specific to the application and sensor platforms in consideration, and is out of scope for this chapter. But, using fine-grained power management schemes to extend the lifetime of a WSN is basic. It does not require incurring any additional expenditure (due to deploying redundant batteries or deploying redundant nodes). It comes for free.

Applying Fine-grained Power Management Schemes: In several always-off applications, it was possible to extend the network lifetime by more than 10 times by using fine-grained power management schemes alone [19, 22]. In the habitat monitoring application [22], sensor nodes were allowed to sleep for 20 minutes, at the end of which they woke up for 70 seconds, collected data, transmitted it to a

gateway, and then went back to sleep. By doing so, it was possible to extend the lifetime of a sensor node from 5 days to 67 days, a factor of more than 13 times.

It is obvious that the strategy of putting all the sensor nodes to sleep and have them all wakeup periodically to sample the environment will not work for always-on applications because of a need to continuously monitor the environment. It has been widely believed that significant lifetime extensions (a factor of 10 or more) for always-on applications can only be achieved by using coarse-grained power management schemes [1, 5, 6, 12, 13, 17, 27, 28] that require deploying more sensor nodes than are absolutely necessary to meet the monitoring and notification requirements of the application. To the best of our knowledge, there have been no study of how much lifetime extension can be achieved for always-on sensor networks by using fine-grained power management schemes alone.

In this chapter, we address the following question: *How long can an always-on sensor network last if only a minimal number of sensor nodes are deployed so that all the deployed nodes are required to be always active to provide the desired level of monitoring?* We show that an always-on sensor network can also achieve comparable lifetime extensions as an always-off sensor network by using only fine-grained power management schemes. We describe several fine-grained power management schemes (e.g. hierarchical sensing, low power listening [21]), where selected components of a sensor node are put to sleep while still meeting the continuous monitoring and instantaneous notification requirements (notifying a base station of an event as soon

as it occurs) of an always-on application. We show by using derivations and concrete numbers the extent of lifetime extensions that can be achieved with these fine-grained power management schemes.

As a case study, we analyze the lifetime of ExScal [2] — a wireless sensor network deployed to detect and classify intruders of different kinds[1]. Our analysis reveals that using low power listening [21] can extend the ExScal lifetime from 3 days to 8 days (Section 4.4). Using hierarchical sensing further extends it to 36 days (Section 4.5). If it were possible to eliminate the periodic control messages, the ExScal network could be made to last 48 days, which represents a lifetime extension by 16 times (Section 4.6). Using other schemes such as in-network data aggregation and reduced reprogramming can increase the lifetime of ExScal further and none of these schemes require deploying any more sensor nodes in the network than are absolutely necessary to provide the desired level of monitoring.

After reading this chapter, the readers will be able to assess the suitability of various fine-grained power management approaches for always-on applications of wireless sensor networks. More importantly, our measured data and lifetime analysis will help readers formulate better (i.e. more accurate) models in their research on power management. For example, the assumption that a mote from the Mica family (a popular sensor platform) [7, 14] lasts 3-5 days on a pair of AA batteries if deployed in an always-on sensor network should be changed to 30-50 days. Also,

[1]The ExScal network (consisting of close to 1000 wireless sensor nodes) was deployed on the ground in December 2004 to demonstrate the proof of concept. Incidentally, this was the largest wireless sensor network deployed on the ground till year 2004.

the claim that in-network data aggregation can extend the network lifetime by more than 50% becomes questionable because our analysis reveals that in-network data aggregation can extend the lifetime of ExScal by at most 8.91%.

Organization of the Chapter. In Section 2, we discuss several fine-grained power management schemes that can be used to extend the lifetime of a wireless sensor network deployed for an always-on application. In Section 3, we provide an overview of the ExScal application, the sensor platform used in ExScal, and major factors that affect the network lifetime of ExScal. In Section 4, we analyze the lifetime of ExScal, illustrating the lifetime extensions achievable by using various fine-grained power management schemes. Section 5 concludes the paper.

2 Fine-grained Power Management Schemes

In this section, we discuss some fine-grained power management techniques that can be applied to extend the lifetime of a WSN deployed for an always-on application. All of these schemes exploit some redundancy already existing in the network to extend network lifetime without affecting the quality of service offered by the network.

2.1 Low Power Listening (LPL)

In an always-on application such as ExScal, most of the time there is no communication in the network. However, the radio can not be turned off on all sensor nodes, because as soon as an event occurs, the event notification message should be quickly propagated to a base station using radios

on other sensor nodes that sit in the path of the sensor detecting an event and the base station. Therefore, ideally, we would like to have a radio that can wake up from the sleep mode by hearing a transmission from a neighbor so that it can be put to sleep when there is no communication in its neighborhood. Such a radio, called radio-triggered wakeup radio, was proposed in [11]. However, it is not yet available on current sensor platforms.

Low Power Listening (LPL) proposed in [21] is an approximation to the radio-triggered wakeup. It allows a sensor node to put its radio (and the processor) to sleep mode for a certain interval and wake it up periodically to sample the channel. If the radio detects preamble bytes, it stays awake and extracts the entire packet. Otherwise, it returns to sleep. This feature was implemented on the sensor platform used in ExScal.

One downside of using the low power listening feature is that the sender has to send a preamble at least as long as the sleep period of the radio. So, there is a trade-off in choosing the sleep period, as was pointed out in [21]. We will analyze the effect of this trade-off in Section 4. Despite this trade-off, low power listening feature can significantly extend the lifetime of WSNs deployed for always-on applications because communication is rare (less than 10 packets every minute) in most of them. *In Section 4.4, we show that by using LPL we can extend the lifetime of ExScal by 2.6 times.*

2.2 Hierarchical Sensing

The concept of hierarchical sensing was originally introduced in [9] under the name of energy-quality hierarchy. Here, we provide a more general definition of the concept and identify its defining characteristics so that it can be used in other always-on applications.

In most always-on applications, the environment needs to be monitored continuously. However, keeping all the sensors and the processor continuously active consumes significant energy. For example, in ExScal, if all the sensors and the processor were left continuously active, a sensor node would have lasted less than 5 days, even if the radio was always turned off. If a sensor can sense the environment without the processor being active, then we can significantly extend the network lifetime by putting the processor to sleep until an event is detected. Further lifetime extensions are possible if the following holds for an always-on application:

- The sensing platform used is intended to detect multiple types of events. (In ExScal, the network is required to detect persons on foot and vehicles.)
- All event types are accompanied by a common simple event. (In ExScal, every intruder is a moving object so that the simple event is the movement.)
- A subset of sensors (called wakeup sensors) can detect the simple event. (In ExScal, PIR sensor detects all moving objects.)

A sensor (or a set of sensors) qualifies as a *wakeup sensor*,

if it has the following features:

1. It does not need the processor to be active to perform an event detection. Sampling of the environment, signal processing of the sampled data, and thresholding (for detecting an abrupt change in the environment due to an event) can be done in the sensor hardware without involving the processor.

2. It has the hardware circuitry to raise an interrupt that can wakeup a sleeping processor.

3. It can detect the common simple event. (This feature is necessary because otherwise some events can be missed by the network.)

4. It has the longest sensing range of all the sensors mounted on a sensor node. (Again, this feature is necessary because otherwise some events can be missed by the network.)

A sensor platform is said to have the *Hierarchical Sensing* feature if it has at least one wakeup sensor. If a platform has the hierarchical sensing feature, it can just keep the wakeup sensor active continuously and put the processor and all the other sensors to sleep. In case an event is detected by the wakeup sensor, it will wake up the processor, which will further process the sensor data to determine if a real event has occurred, and if so, it will wake up all the sleeping sensors to detect other properties of the event using different sensing modalities. For example, in ExScal, the wakeup sensor can detect the presence of an intruder

and the sleeping sensors can help classify the type of the intruder.

For the hierarchical sensor scheme to work, the choice of wakeup sensor is critical. The wakeup sensor should not wakeup the processor very frequently due to false alarms. The best wakeup sensor is the one that draws a small current. For the sleeping sensors, it is important to have a low startup time so that when they are woken up, they do not miss an event. *In Section 4.5, we show that by using hierarchical sensing (with PIR sensor as the wakeup sensor) together with LPL we can extend the lifetime of ExScal by 12 times.*

2.3 Other Fine-grained Power Management Schemes

There are several other fine-grained power management schemes that can be used in an always-on application to extend the network lifetime. Some of these are:

1. **Reducing Periodic Messages:** Reducing periodic control messages can result in significant lifetime extensions. *The lifetime of ExScal can be increased by 31.6% if there were no periodic messages.*
2. **In-Network Data Aggregation:** Aggregating the event detection messages as it flows towards the base station can also extend network lifetime. *In ExScal, using data aggregation can result in a lifetime extension of upto 8.91%.*
3. **Reduced Control Operations:** Reducing the number of control operations such as wireless reprogramming results in further lifetime extensions.

4. **Reduced Actuations:** Actuations such as blinking LEDs and sounding buzzers can consume significant energy. *For example, keeping even one LED continuously active will reduce ExScal's lifetime by more than half.*

We provide the details of calculation on how to analyze the effect of the above schemes on the lifetime of a WSN in Section 4.

Finally, we discuss two more fine-grained power management techniques without analyzing their effects on the lifetime of ExScal.

- **Duty Cycling the Wakeup Sensor:** In some sensors such as the acoustic, it is possible to reduce the energy consumption of this sensor by letting the sensor sleep in between its samplings. In ExScal, acoustic sensor collected samples at the rate of 4 kHz for 30 ms, after every 300 ms. Since the startup time of the acoustic sensor is less than 1 ms, it can be put to sleep in between its samplings to save energy. After it collects one set of samples, it can be put to sleep for the next 269 ms, at the end of which it will wakeup, collect another set of samples for 30 ms and go back to sleep. If its sampling frequency is reduced (so that it sleeps for more than 269 ms in every cycle), in order to conserve even more energy, its sensing range may get reduced. It may still be possible to meet the monitoring requirements of the application with this reduced sensing range. A careful analysis needs to be performed before reducing the sensing range of a sen-

sor in order to ensure that the application requirements with respect to coverage can still be met with the reduced sensing range. If the acoustic sensor was used as a wakeup sensor, significant energy savings could have been achieved with this duty cycling. Unfortunately, duty cycling could not be used to reduce the energy consumption of the PIR sensor (which was the wakeup sensor in ExScal that remained continuously active) because of its high startup time (more than 1 second).

- **Reducing the Transmitter Power Level:** With the radio used in XSM and in motes from the Mica family, the energy consumed in transmission depends on the power level used. Using a lower power level means lower energy consumption at the expense of a reduced transmission range. Using a lower transmission range also means lower interference in the network. On the other hand, using a lower transmission range can result in more hops in a multi-hop network. The reliability of transmitting a packet across multiple hops decreases as the number of hops increases in the path of the packet. Therefore, a careful analysis should be performed before reducing the transmission range to ensure that the connectivity and packet reliability requirements of the application are still met with a reduced transmission range.

There may be more fine-grained approaches to extend the lifetime of a WSN than what we have discussed in this chapter. An application developer should explore all these options before deciding to deploy redundant sensor nodes

or redundant batteries in order to get a desired lifetime from the network.

3 The ExScal Application and the XSM Platform

In this section, we provide an overview of the ExScal application [2], an overview of the sensor platform used in ExScal (called XSM) and the major requirements (or features) of ExScal that have a significant impact on its lifetime.

The goal in the ExScal application was to deploy a wireless sensor network over a large region to monitor intrusion activities. The network was required to detect different types of intruders breaching the perimeter of the protected region, classify them into some predetermined categories (e.g. person, soldier, car, tank), and track their trajectories of intrusion. The network was also required to notify the nearest base station of an intrusion event in less than 2 seconds.

The key issues in ExScal were to minimize the cost of coverage, minimize the power consumption to maximize the network lifetime, provide accurate (i.e. low false alarm rate) and timely (i.e. less than 2 seconds from the occurrence of the event) detection of intrusion events in the face of unavoidable hardware and software failures, and do all of this with low human involvement.

Minimizing the cost of coverage required minimizing the number of sensor nodes needed (which for our purpose means not deploying any more sensor nodes than are absolutely necessary to meet the monitoring and notification requirements of the application). This, in turn, required de-

ploying nodes in an optimal topology[2]. We refer the readers to [16] for details on the layout of sensor nodes that was used in ExScal. Minimizing the cost of coverage also required finding off-the-shelf sensors with the largest sensing ranges and using radios that provided the largest communication range with the lowest energy consumption, while keeping the cost low.

Accurate and timely detection of intruders were critical to ExScal. Accuracy had priority over timeliness. What good is a network that gives false detections quickly? Therefore, the network was required to have a low false alarm rate. If the detection message does not reach the base station quickly, an intruder may compromise the asset being protected by the sensor network. Therefore, low latency of event notification was also critical to ExScal. ExScal was required to achieve both low false alarm rate and low latency of detection in the face of unavoidable failures (both hardware and software). We refer the readers to [3] for a list of hardware, software, deployment, localization, and other failures encountered in the ExScal demonstration.

Finally, low human involvement is a key to the operation of a large scale sensor network. Imagine the effort and time needed if one thousand sensor nodes deployed over a 1 km long region have to be touched individually by a human for some reason (e.g. to turn them on and off). Therefore, a large scale sensor network such as the ExScal needed to provide easy operation, require minimal or no touching of

[2]The appropriate notion of coverage for intrusion detection application is *barrier coverage* [18], where sensors form a barrier for intruders. For a precise definition of the concept of barrier coverage and several interesting results, including the optimal topology to achieve k-barrier coverage, we refer the readers to [18].

individual sensor nodes, and allow monitoring of network health and reconfiguration of network parameters from a remote central location.

A new sensor platform, called an Extreme Scale Mote (XSM) [9], was developed for ExScal. It was a refinement of Mica2 [7]. The details of this platform with regard to its power management capabilities appear in Section 3.1. The operating system used on this platform was TinyOS. Several middleware services such as routing, time synchronization, and localization were custom developed for ExScal. The signal chains that were used by the XSMs to locally process the sensor data were also custom developed. Finally, the application software to detect and classify intruders were also developed in-house.

To demonstrate the concept, approximately 1000 XSMs were deployed in a 1,200 m $\times$ 288 m rectangular region [16] and intruders such as persons and Sport Utility Vehicles (SUVs) were shown to be detected and classified by the sensor network. At the end of year 2004, this was the largest wireless sensor network in the world deployed on the ground. Figure 1 shows the XSMs deployed for ExScal demonstration.

Each XSM ran on a pair of AA batteries. If the XSMs were left continuously active, the ExScal network would have lasted only 3 days (see Section 4.3 for details of the calculation). However, when a wireless sensor network is deployed on the ground for real-life application (rather than to demonstrate the concept), such a network would be required to last for months, if not years. This motivated us to investigate the various power management schemes that

Figure 1: XSMs (white dots forming a grid) deployed on the ground (a 1,200 m $\times$ 288 m rectangular region) for ExScal demonstration. Figure 2 zooms on a single XSM.

can be used to extend the lifetime of the ExScal network from 3 days to several months, without deploying redundant sensor nodes or redundant batteries per node (keeping in mind the cost minimization objective of ExScal).

3.1 The XSM Platform

The XSM (Extreme Scale Mote) [9] is a sensor platform developed for the ExScal project. It was a refinement of the Mica 2 platform [7]. Its design was optimized for use in intrusion detection applications. Figure 2 shows an XSM deployed on the ground in its usual enclosure. The XSM had three types of sensor — a 2-axis Magnetometer to detect ferrous materials, a Passive Infrared (PIR) to detect motion, and an Acoustic sensor to detect objects making

sounds (e.g. vehicles). Figure 3 shows the circuit board of an XSM. The sensing ranges for these sensors for various types of objects appear in Table 1.

Figure 2: An XSM (Extreme Scale Mote) deployed on the ground in its usual enclosure during the ExScal demonstration.

Sensor	**Sensed Object**	**Sensing Range**
Magnetometer	SUV	7 m
PIR	SUV	30 m
	Person	12 m
Acoustic	SUV	30 m

Table 1: Sensing ranges (in meters) of the three sensors used in the ExScal project

The current consumption of the major components of XSM appear in Table 2. We use mA (milliamperes) for the unit of current consumption. The amount of energy consumed by a component will depend on the amount of time that it is used.

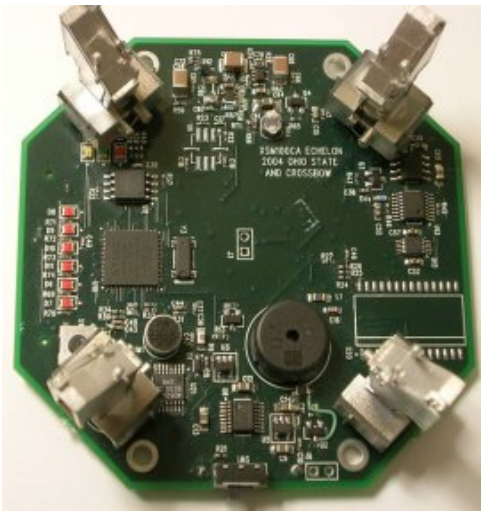

Figure 3: Inside an XSM.

3.2 Factors Affecting ExScal's Lifetime

The major factors affecting the network lifetime of ExScal are as follows:

1. **Continuous Monitoring:** The region should be continuously monitored so that intruders can be detected instantly. This may require keeping at least one sensor continuously active, consuming significant energy.

2. **Event Notification Requirement:** Intrusion detection events should be communicated to a base station quickly. In the ExScal application, the requirement was to receive event detection notification at the nearest base station within 2 seconds. In order to communicate event-notification messages quickly over a multihop wireless sensor network, several, if not all, sensors need to keep their radio in the receive mode either con-

Component	State	Current (in mA)
Processor	active	8
	sleep	0.01
Radio	active	8
	transmit	16
	sleep	0.001
PIR	active	0.292
	sleep	0.001
Acoustic	active	0.575
	sleep	0.001
Magnetometer	active	6.48
	sleep	0.001
One LED	active	2.2
	sleep	0.001
Flash	Read	6.2
	Write	18.4
	Sleep	0.002
Buzzer	active	15
	sleep	0.001

Table 2: Current consumption of major components in the XSM platform

tinuously or frequently enough so that they can route an urgent event-detection message towards the base station. This again consumes significant energy.

3. **Periodic Control Messages:** Two middleware services namely, *routing* and *time synchronization* required every XSM to transmit periodic messages. As we will see in Section 4.6, sending periodic control messages consumes significant energy.

4. **One Time Control Operations:** There were several one time activities performed in the ExScal application. The major ones among them were wireless reprogramming and localization. These operation re-

quired the sensor nodes to be active for a long duration (on the order of tens of minutes), send a large number of messages (in reprogramming), and perform actuation activities (e.g. sounding buzzers). All of these consume significant energy.

5. **Frequency of Events:** Every event requires the sensors near the event to not only stay awake for a few seconds to detect the event but also to transmit messages in a multi-hop sensor network, and potentially route other XSM's messages. Staying awake with the processor and all the sensors active consumes significant energy and the total energy consumed this way depends on the frequency of the events.

Each of the above factors dictate which fine-grained power management schemes can be used in ExScal and which ones are not usable. For example, LPL can be used in ExScal but the periodic sleeping time of the radio should be low enough (less than 400 ms) to satisfy the 2 second event notification latency.

4 Lifetime Analysis of ExScal

In this section, we analyze the lifetime of ExScal. We first discuss some key assumptions needed in the lifetime analysis, then we derive the parameters needed in the lifetime analysis. Next, we use these parameters to derive the network lifetime in the fully active mode, when using the LPL feature, and when using the hierarchical sensing feature. Finally, we analyze the effects of other fine-grained power

management schemes such as varying the frequency of periodic control messages, performing in-network data aggregation, tuning the number of wireless reprogrammings, and controlling the amount of actuations performed in the network, on the lifetime of ExScal.

We define the lifetime of a WSN to be the time period during which the network continuously satisfies the application requirement. The application requirement can be stated in various forms. One simple way to express the requirement of an always-on application is in terms of the degree of coverage and the notification latency. For example, in ExScal, all intruders were required to be detected by the network at least 5 times in their trajectory through the network (in order to perform detection with a high probability) and the event notification was required to reach the closest base station in at most 2 seconds. We derive a lower bound on the lifetime of ExScal. The purpose of doing so is to allow some buffer so that even if some factors are missed in the analysis (which almost always are), the network has a high likelihood of lasting at least as long as predicted by the analysis.

4.1 Key Assumptions Needed in Lifetime Analysis

In analyzing the lifetime of any WSN, some key assumptions need to be made based on the expected use of the network. We make the following assumptions for ExScal, with the goal of deriving a conservative estimate of its lifetime:

1. The lifetime of the network is determined by the heaviest-

loaded XSM. This is an XSM close to the base station. This is conservative because ExScal network will continue to meet its requirements even if all XSMs within one hop of a base station fail.

2. Every hour, 6 intrusion events occur in the vicinity of the heaviest-loaded XSM[3]. With equal probability, the event can be the intrusion of a person or that of an SUV. Further, the intruders are assumed to follow the least-covered path through the network. With this last assumption, the number of sensors detecting an intruder in the ExScal network is given by the values in Table 3.

3. Every time an event occurs, the heaviest-loaded XSM keeps its processor as well as all three of its sensors active for an average of 10 seconds. This is because, on average, a slow moving target (a person on foot) will spend an average of 10 seconds in the sensing range of a sensor.

4. One-eighth of the event detection messages generated in the vicinity of the heaviest-loaded XSM are routed by the heaviest-loaded XSM[4].

5. The average number of times a data packet is transmitted for reliable delivery across a single hop is 1/0.7 = 1.43. This is because the per-hop reliability of data packets was 0.7 in ExScal[5]. If we assume the probabil-

[3]This event rate is higher than what ExScal was required to handle.

[4]This is because grid routing [26] is used in ExScal, which balances the routing load on multiple routes and because of the topology used in ExScal [16].

[5]The per-hop reliability was close to 1 in the absence of interference but was 0.7 in the presence of interference (as is the case in an operating sensor network).

ity of losing a packet is independent and identically distributed, the expected number of times a packet needs to be transmitted for successful delivery is $1/p$ (expectation of geometric distribution [23]), where p is the probability of success in each transmission[6].

6. Periodic control messages such as routing and time synchronization updates are never retransmitted.

7. We assume a fixed packet length of 36 bytes, which is the maximum length of a packet in B-MAC [21] (the MAC protocol used in ExScal), and that it takes 20 ms to transmit a packet. In practice, event detection messages are very short, and therefore the packet length will be smaller and so will the transmission time.

The assumptions for any other always-on application will mostly be along the lines of the above assumptions with some changes specific to that application.

Intruder	**# of PIRs**	**# of Acoustic**	**# of Magnetometers**
SUV	30	30	5
Person	10	None	None

Table 3: Number of sensors detecting an intruder type in the ExScal application

4.2 Parameters for Lifetime Analysis

In this section, we define the parameters used in analyzing the lifetime of an always-on WSN and derive their values

[6]The value of 1.43 is an approximation. The actual value will be lower than 1.43 because the maximum times a packet was transmitted was 3, whereas in geometric distribution the maximum number of trials is assumed to be infinite (trials are made until there is a success).

in the ExScal application.

Identifier	Meaning	Value
$e_{battery}$	Usable capacity of the batteries	1800 mA-hr
i_{act}^{proc}	Current drawn by the processor in active mode	8 mA
i_{act}^{rad}	Current drawn by the radio in active mode	8 mA
i_{tx}^{rad}	Current drawn by the radio in transmit mode	16 mA
t_{tx}^{pkt}	Time to transmit a packet	0.02 s
t_{slp}^{lpl}	Sleep period of the radio in the LPL mode	variable
i_{lpl}	Amortized current drawn by an XSM in the LPL mode	variable
m_{pr}	# control messages transmitted every hour	240
m_e	# event-detection messages generated by the heaviest-loaded XSM every hour	variable
m_{pp}	# event-detection messages routed by the heaviest-loaded XSM every hour	variable
i_{act}^{msg}	Current drawn in transmitting 1 packet when the radio is in active mode (no LPL)	variable
i_{lpl}^{msg}	Current drawn in transmitting 1 packet when the radio is in LPL mode	variable
i_{awk}^{event}	Amortized current drawn in staying awake due to events	variable
i_{sensor}	Current drawn by continuously active sensors	variable
f_{repr}	# reprogramming is performed wirelessly	6
e_{repr}	Energy spent in 1 wireless reprogramming	18.22 mA-hr
$e_{localization}$	Energy spent in 1 localization	4 mA-hr

Table 4: Notations for parameters used in the lifetime analysis and their values in ExScal. The values of the parameters that are variable are derived in Section 4.2.

The notations for the parameters used in lifetime analysis appears in Table 4. In the following, we derive the values or expressions for those parameters whose values are not straightforward to calculate or whose values are variable (in the context of the ExScal application).

t_{slp}^{lpl}: Sleep period of the radio in the LPL mode. It can take a value of 0.01, 0.02, 0.05, 0.1, 0.2, 0.4, 0.8, or

1.6 seconds [21]. We do not use sleep periods of 0.8 seconds and 1.6 seconds in our analysis because the notification latency requirement of 2 seconds can not be satisfied with these sleep periods.

i_{lpl}: Amortized current drawn by an XSM in the LPL mode. Its value depends on the following four factors:

1. t_{slp}^{lpl} - The sleep period used for the radio (listed above).
2. i_{slp}^{lpl} - The average current consumed by an XSM when the processor and radio are sleeping (excluding the current consumed by any active sensor). By measurement on an XSM, we found that i_{slp}^{lpl} =0.1974 mA[7].
3. i_{sample}^{lpl} - The average current consumed by an XSM in sampling the channel every time the radio wakes up. By measurement on an XSM, we found that i_{sample}^{lpl} = 9.6571 mA[8].
4. t_{sample}^{lpl} - Time taken to sample the channel every time the radio wakes up to sample the channel. By measurement on an XSM, we found that t_{sample}^{lpl} = 0.014 seconds.

[7]This is a significantly higher value than expected in sleep mode. We expected it to be close to 0.035 mA. The higher value of 0.1974 mA is because in the LPL mode, the XSM wakes up every 27.6 ms. Every time it wakes up, it follows the following pattern: it consumes 0.035 mA during the 27.6 ms that the XSM is sleeping, it consumes 0.4 mA for the next 0.8 ms, 0.75 mA for the next 2.25 ms, and 6.35 mA for the next 0.5 ms, after which the 27.6 ms of sleep period follows. With careful configuration and some hardware improvements, the value of i_{slp}^{lpl} may be brought down to 0.035 mA, which will significantly increase an XSM's lifetime.

[8]Similar values for i_{sample}^{lpl} were reported in [9].

Using the above values, we obtain

$$i_{lpl} = \frac{i_{slp}^{lpl} * t_{slp}^{lpl} + i_{sample}^{lpl} * t_{sample}^{lpl}}{t_{slp}^{lpl} + t_{sample}^{lpl}} = \frac{0.1974 * t_{slp}^{lpl} + (9.6571 * 0.014)}{t_{slp}^{lpl} + 0.014}. \quad (1)$$

m_{pr}: Number of control messages transmitted every hour. With the configuration used in the ExScal application, every XSM sends 3 routing messages and 1 time synchronization messages every minute, for a total of 240 messages every hour. Therefore, in ExScal, $m_{pr} = 240$.

m_e: Number of event-detection messages generated by the heaviest-loaded XSM every hour. Two messages are generated for every sensor for every event. It means an XSM detecting an event sends 2 messages per sensor per event, for a total of 6 messages. With our assumption of 6 events per hour and a per-hop retransmission factor of 1.43 per event-detection message, $m_e = 52$.

m_{pp}: Number of event-detection messages routed by the heaviest-loaded XSM every hour.The heaviest-loaded XSM is the one close to the base station. Every event generates 2 messages per sensor from every XSM that detects this event. The number of XSMs that detect an intruder type appears in Table 3. These numbers depend on the topology of the sensor network deployed [16] and the assumption that intruders always cross through the network via least-covered paths. Assuming both the intruder types are equally likely to occur, every event results in the generation of $2 * ((30 +$

$30 + 5) + (10 + 0 + 0))/2 = 75$ messages (using the values from Table 3). The routing used in the ExScal application [26] balances the routing load on 8 XSMs that are within one hop of the base station. Therefore, the most energy constrained XSM routes an average of 10 messages for every event. Assuming a retransmission factor of 1.43, it will transmit an average of 15 messages for every event. Since 6 events are assumed to occur every hour, 90 event detection messages are routed by the most energy constrained XSM every hour. Therefore, in ExScal, $m_{pp} = 90$.

i_{act}^{msg}: Current drawn in transmitting 1 packet when the radio is in active mode (no LPL). Its value is given by the following expression:

$$\begin{aligned} i_{act}^{msg} &= \frac{(m_{pr} + m_e + m_{pp}) * t_{tx}^{pkt} * (i_{tx}^{rad} - i_{act}^{rad})}{3600} \\ &= \frac{(240 + 52 + 90) * 0.02 * (16 - 8)}{3600} \\ &= 0.02\text{mA}. \end{aligned} \tag{2}$$

i_{lpl}^{msg}: Current drawn in transmitting 1 packet when the radio is in LPL mode. Its value depends on the sleep period, t_{slp}^{lpl} used by the radio. More specifically,

$$\begin{aligned} i_{lpl}^{msg} &= \\ &\frac{(m_{pr} + m_e + m_{pp}) * (t_{tx}^{pkt} + t_{slp}^{lpl}) * (i_{act}^{proc} + i_{tx}^{rad})}{3600} \\ &= \frac{(240 + 52 + 90) * (0.02 + t_{slp}^{lpl}) * (8 + 16)}{3600}. \end{aligned} \tag{3}$$

i_{awk}^{event}: Amortized current drawn in staying awake due to events. Its value depends on two factors:

1. t_{awk} - Time (in seconds) that the heaviest-loaded XSM is awake every hour due to events. We assume that an XSM close to the base station stays awake for 10 s after the occurrence of an event. This may be for routing all event detection messages. With the assumption of 6 events every hour, $t_{awk} = 60$ seconds in ExScal.
2. $i_{sensors}^{slp}$ - The current consumed in the active mode by all the non-wakeup sensors. From Table 2, we obtain, $i_{senors}^{slp} = 0.595+6.48 = 7.075$ mA.

With the above values and the fact that the processor is also awake when the XSM is awake due to events, we obtain,

$$\begin{aligned} i_{awk}^{event} &= \frac{t^{awk} * (i_{act}^{proc} + i_{sensors}^{slp})}{3600} \\ &= \frac{60 * (8 + 7.075)}{3600} = 0.2618\text{mA}. \end{aligned} \quad (4)$$

i_{sensor}: Current drawn by continuously active sensors. Its value depends on which sensors are kept continuously active. The current consumption for the sensors used on the XSM appear in Table 2.

e_{repr}: Energy spent in 1 wireless reprogramming. One reprogramming of 55 kByte program (the size of ExScal program) requires every XSM to stay awake for approximately 45 minutes. Assuming the XSMs are not in the LPL mode during reprogramming, just staying

awake for 45 minutes consumes 18 mA-hr of energy[9]. The reprogramming of 55 kByte requires the most constrained XSM to send out 1,942 packets, where each packet has 29 bytes of data. Assuming a retransmission factor of 1.43, approximately 2,771 packets are sent by the most energy constrained XSM. One packet transmission takes 20 ms and consumes an extra current of 8 mA. Therefore, the additional energy spent in transmission of 2771 packets is 0.12 mA-hr (8*2771*0.02/3600.). Writing 55 kByte to flash takes less than 4 seconds. Each second, it consumes 18.4 mA of current. So, total energy consumed in flash writing is less than 0.02 mA-hr. Adding up the energy consumed in staying awake for 45 minutes (18 mA-hr), in transmissions (0.12 mA-hr), and in flash writing (0.02 mA-hr), we get a total of approximately 18.14 mA-hr of energy consumed in 1 reprogramming. Therefore, in ExScal, $e_{repr} = 18.14$ mA-hr.

$e_{localization}$: Energy spent in 1 localization. As seen in the calculation done above for reprogramming, the energy consumed in staying awake is the dominant part of energy consumption. In ExScal, the XSMs were awake for 10 minutes during localization, consuming 4 mA-hr. Therefore, $e_{localization} = 4$mA-hr.

[9]Keeping XSMs in the LPL mode during reprogramming will cause a higher energy consumption because the XSMs will need to send long preambles, consuming significant energy. Also, it will take much longer than 45 minutes to reprogram the network if the XSMs are in the LPL mode, because of reduced channel capacity in the LPL mode.

4.3 Lifetime in the Fully Active Mode

If an sensor node is always in the active mode, its lifetime, ℓ_{hr} (in hours) will be given by:

$$\ell_{hr} = \frac{e_{battery} - f_{repr} * e_{repr} - e_{localization}}{i_{act}^{proc} + i_{act}^{rad} + i_{act}^{msg} + i^{sensor}}. \tag{5}$$

Substituting the following values

$$\begin{aligned} i_{act}^{msg} &= 0.02\text{mA}, \quad \text{(from (2))} \\ i^{sensor} &= 0.292 + 0.575 + 6.48 = 7.347\text{mA}, \quad \text{(from Table 2)} \end{aligned}$$

and those from Table 4, we obtain $\ell_{hr} = 72.2$ hours (or 3 days) for the lifetime of ExScal.

4.4 Lifetime When Using Low Power Listening (LPL)

If we use the low power listening mode (see Section 2.1), then the network lifetime, ℓ_{hr} (in hours) is given by

$$\ell_{hr} = \frac{e_{battery} - f_{repr} * e_{repr} - e_{localization}}{i_{lpl} + i_{lpl}^{msg} + i_{sensor} + i_{awk}^{event}}. \tag{6}$$

Substituting

$$\begin{aligned} i_{lpl} &= \frac{(0.1974 * t_{slp}^{lpl}) + (9.6571 * 0.014)}{t_{slp}^{lpl} + 0.014}, \quad \text{(from (1))} \\ i_{lpl}^{msg} &= \frac{(382) * (0.02 + t_{slp}^{lpl}) * (8 + 16)}{3600}, \quad \text{(from (3))} \\ i^{sensor} &= 0.292 + 0.575 + 6.48 = 7.347, \quad \text{(from Table 2)} \\ i_{awk}^{event} &= 0.2618\text{mA}, \quad \text{(from (4))} \end{aligned}$$

and those from Table 4, we obtain a graph of the ℓ_{hr} as a function of t_{slp}^{lpl} shown in Figure 4 for the lifetime of ExScal.

The lifetime curve is concave because there is trade-off in choosing the wakeup period for the radio in the LPL mode. The higher the wakeup interval, the lower the energy consumption when an XSM is sleeping, but higher the energy consumed in sending a longer preamble. The optimal value of lifetime occurs at 187.99 hours or 7.83 days. This represents an increase by a factor of 2.6 over that in the fully active mode.

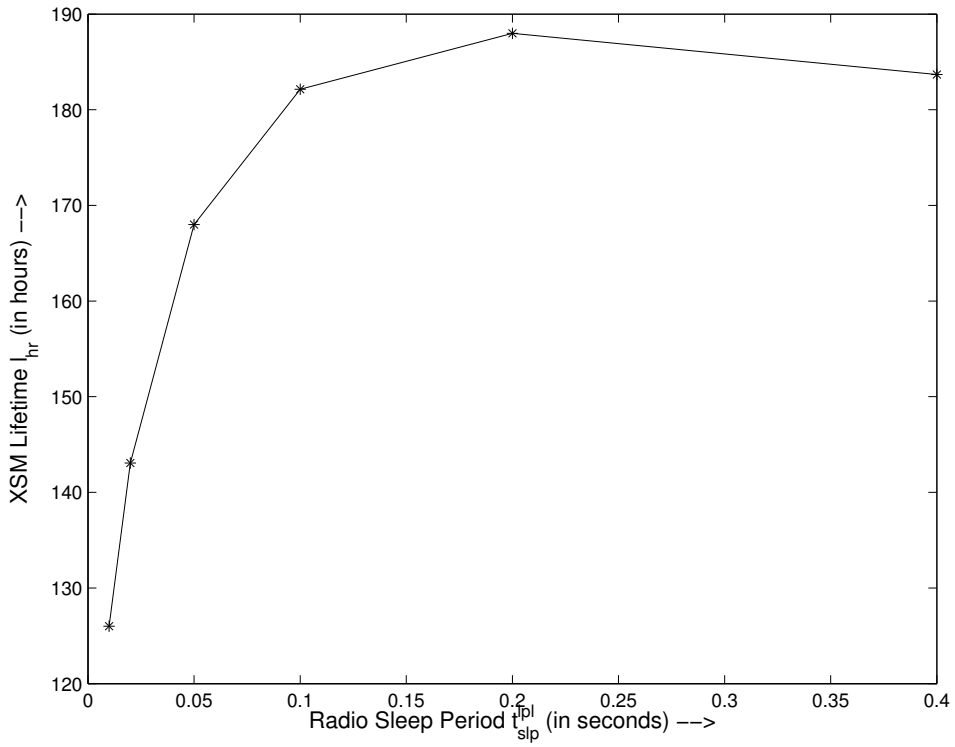

Figure 4: ExScal network lifetime in the low power listening mode as a function of radio sleep period, t^{lpl}_{slp}.

4.5 Lifetime When Using Hierarchical Sensing With LPL

When using a hierarchical sensor with the LPL mode, the network lifetime is still given by (6), except that now the value of i_{sensor} becomes lower because some other sensors are put to sleep.

In ExScal, the PIR sensor qualifies as a wakeup sensor. Fortunately, this is also the lowest energy consuming sensor, drawing 20 times less current than magnetometer. Figure 5 shows ℓ_{hr} for ExScal as a function of t_{slp}^{lpl}. Here ℓ_{hr} is given by (6) with $i_{sensor} = 0.292$ mA. We observe that the maximum lifetime achievable increases to 878.85 hours (or 36.62 days). This represents an increase by a factor of 4.67 over that achieved by just using the LPL mode and a factor of 12 over that achieved in the fully active mode.

4.6 Effect of Periodic Control Messages on the Lifetime

In this section, we analyze the effect of varying the frequency of periodic control messages on the network lifetime. To study the effect of periodic messages on the lifetime, we vary the value of m_{pr}. The effect of varying m_{pr} on the lifetime in the fully active mode is straightforward (derive a new value for i_{act}^{msg} in (5)). Analyzing the effect of varying m_{pr} when the network is using the LPL mode is, however, nontrivial, because the optimal lifetime now occurs at different values of t_{slp}^{lpl} depending on the range of values of m_{pr}.

The optimal value of ℓ_{hr} (given by (6)) occurs at $t_{slp}^{lpl} = 0.4$ seconds when $m_{pr} \leq 82$, at $t_{slp}^{lpl} = 0.2$ seconds when $83 \leq m_{pr} \leq 672$, and at $t_{slp}^{lpl} = 0.1$ seconds when $673 \leq$

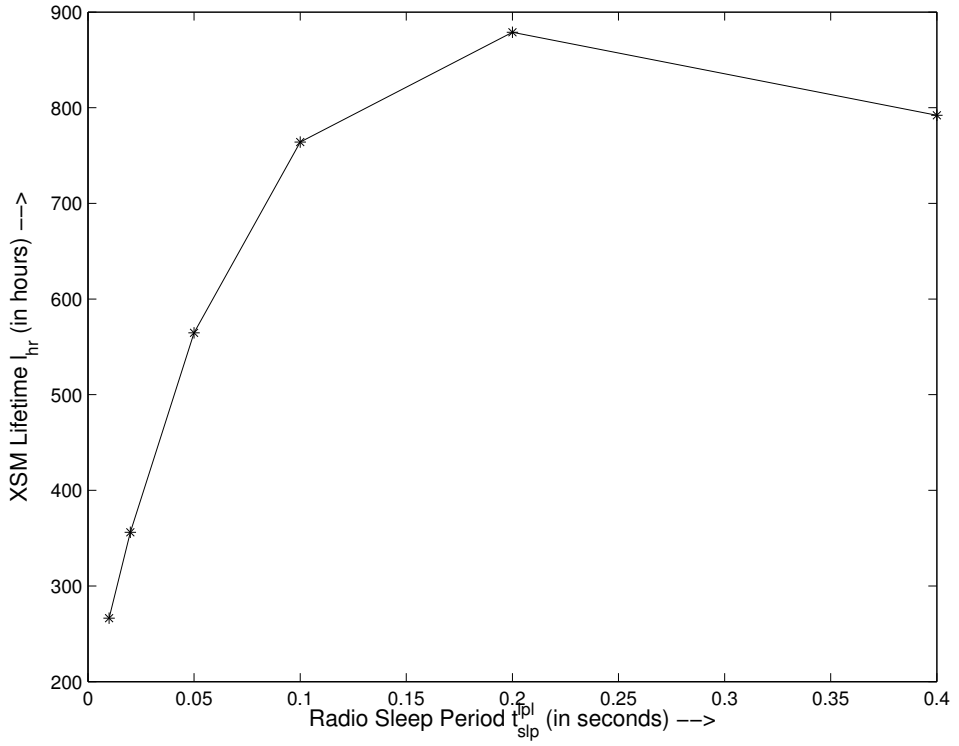

Figure 5: ExScal network lifetime in the low power listening mode and hierarchical sensing with PIR as the wakeup sensor, as a function of radio sleep period, t^{lpl}_{slp}.

$m_{pr} \leq 2580$. We plot the optimal values of lifetime when m_{pr} is varied from 0 to 2580 to analyze the effect of periodic messages on the lifetime of ExScal in Figure 6 when hierarchical sensing and LPL both are used. We notice that if there were no periodic messages, ExScal's life increases to 1157.1 hours (or 48.2 days). This represents an improvement of 31.67%.

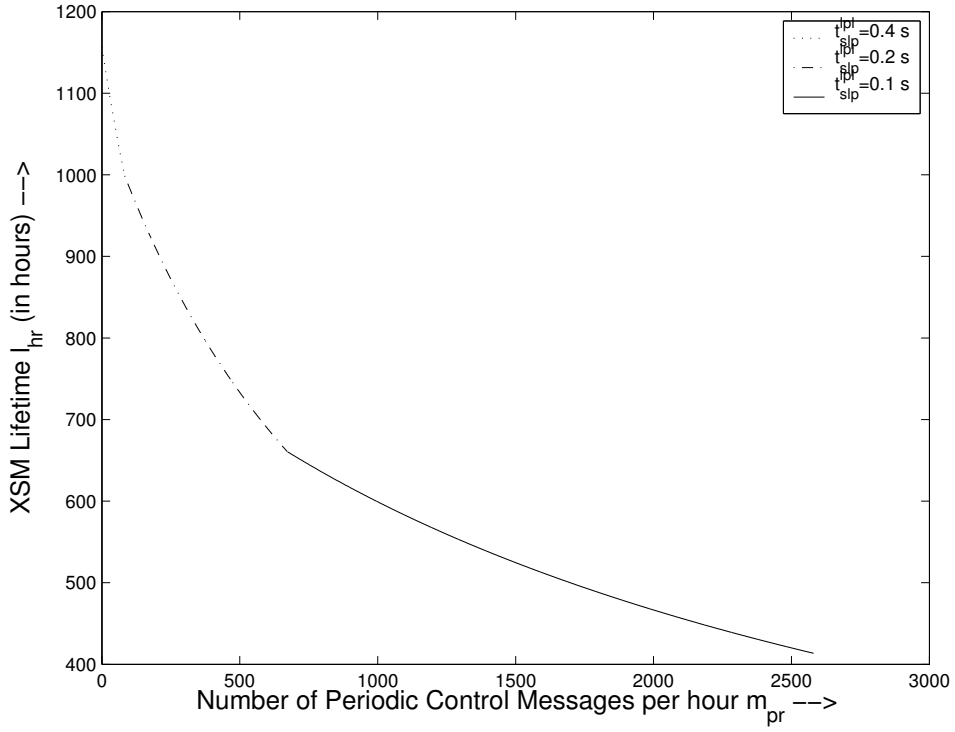

Figure 6: Optimal ExScal network lifetime in the low power listening mode and hierarchical sensing when the number of periodic control messages (m_{pr}) is varied from 0 to 2580 messages per hour.

4.7 Effect of In-Network Data Aggregation on the Lifetime

In this section, we analyze the effect of in-network data aggregation on the lifetime of a WSN. The lifetime of a WSN with data aggregation is still given by (5) and (6), but with new values for m_e and m_{pp}. The amount of data aggregation that can be performed in a WSN depends on the application traffic generated and on the topology as well as the routing protocol used. We can perform the following data aggregation in ExScal, assuming the most optimistic

scenario:

1. We can combine the detection message for all three sensors at an XSM into one message. This will result in reducing m_e by a factor of 1/3 to 17.

2. We can perform aggregation of detection messages flowing upward in a routing tree. Assuming that both intruder types are equally likely to occur, an average of (30+10)/2 = 20 XSMs detect an event (from Table 3). Since the routing load is distributed on 8 XSMs, each XSM forwards the detection messages from at most 3 other XSMs, all of whose data can be combined into one packet. Since each XSM generates two messages for every event, separated in time, an XSM close to the base station will need to forward 2 packets per event. Assuming a retransmission factor of 1.43, and 6 events per hour, each XSM close to the base station will forward 18 messages. Therefore, $m_{pp} = 18$.

Substituting $m_e = 17$, $m_{pp} = 18$, and $i_{sensor} = 0.292$ mA in (6), we obtain a graph of ℓ_{hr}, shown in Figure 7, for the lifetime of ExScal, if hierarchical sensing and LPL mode continue to be used. The maximum life achievable is now 954.6 hours (or 39.78 days). This represents an increase of 8.91% in the lifetime of ExScal over that achieved by using only the hierarchical sensing and LPL mode. In practice, it may not be feasible to achieve this extent of data aggregation. So, the increase in lifetime that we can achieve using data aggregation will be at most 8.91%.

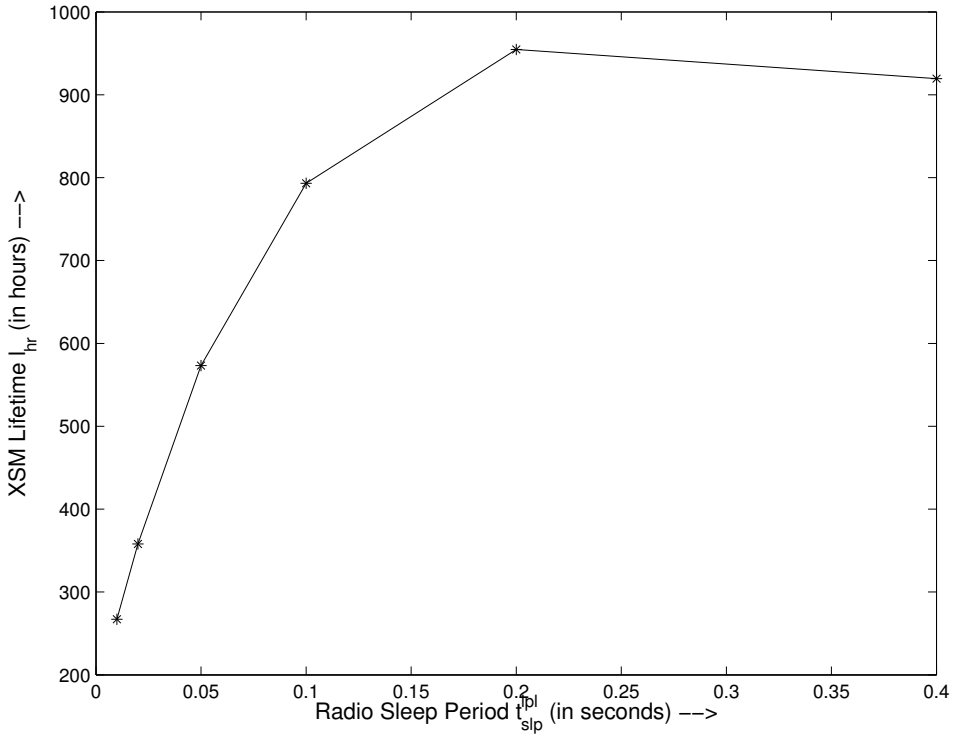

Figure 7: ExScal network lifetime in the low power listening mode with PIR as the wakeup sensor and in-network data aggregation, as a function of radio sleep period, t_{slp}^{lpl}.

4.8 Effect of Wireless Reprogramming and Actuation on the Lifetime

In this section, we analyze the effects of performing frequent wireless reprogramming, blinking LEDs, and sounding buzzers on the lifetime of ExScal.

One wireless reprogramming consumes 18.14 mA-hr of energy. From (6), we get that if f_{repr} is reduced from 6 to 5, the lifetime of ExScal will increase from 871.81 hours to 881.2 hours, an increase of approximately 10 hours.

Today's sensors have limited actuation abilities (e.g. blinking LEDs or sounding a buzzer). In the future, sensor nodes are expected to have more actuation abilities. Actuations are often a major source of energy drain. To analyze the effect of actuations on the lifetime, we can use (5) and (6) with a new term in the denominator to represent the average current draw per hour.

For example, an LED draws a current of 2.2 mA on an XSM(see Table 2). If one LED is kept active continuously, then the lifetime of ExScal will decrease from 871.81 hours to 412.4 hours (i.e. reduce it by more than half). (Substitute $i_{sensor} = 0.292$ mA in (6) and use a new term with a value of 2.2 in the denominator.) If an LED blinks instead of being continuously active, we can use the following approach. Let f denote the fraction of a second that the LED is on. Then, it can be assumed that the LED in consideration draws 2.2 mA $\times f$ amount of current continuously. The remaining analysis will be similar as in the case when an LED is active continuously.

Similarly, to analyze the effect of sounding a buzzer for 1 minute every hour on the lifetime of ExScal, substitute $i_{sensor} = 0.292$ mA in (6) and use a new term with a value of $15/60 = 0.3642$ mA in the denominator (because the buzzer draws 15 mA of current on an XSM). By doing so, we find that an XSM's life will decrease from 871.81 hours to 780.38 hours (a decrease of more than 90 hours of life) as a result of sounding the buzzer for 1 minute every hour by the actuator.

5 Conclusion

In this chapter, we discussed several fine-grained power management techniques that can be used to extend the lifetime of an event-based (always-on) application of wireless sensor network without requiring any additional sensor nodes than is absolutely necessary to meet the application's monitoring requirements. We showed using concrete numbers that the lifetime of ExScal — an application of wireless sensor network for intrusion detection and classification, can be extended by more than 16 times by using low power listening and hierarchical listening alone, both of which are existing features of the XSM platform. Contrary to popular belief, we showed that it is possible to achieve comparable network lifetime extensions for event-based (always-on) applications as for data gathering (always-off) applications, without deploying any more sensor nodes than are absolutely necessary to meet the application's monitoring requirements.

Acknowledgments

We would like to thank Sandip Bapat, Hui Cao, and Hongwei Zhang from the Ohio State University for their valuable feedback on an earlier draft of this paper. We would also like to thank Emre Ertin from the Ohio State University and Prabal K. Dutta from University of California, Berkeley for sharing with us their knowledge of hierarchical sensing and power management features of the XSM.

References

[1] Z. Abrams, A. Goel, and S. Plotkin, Set k-Cover Algorithms for Energy Efficient Monitoring in Wireless Sensor Networks, *In Proceedings of the Third International Conference on Information Processing in Sensor Networks (IPSN)*, Berkeley, CA, 2004.

[2] A. Arora, R. Ramnath, E. Ertin, P. Sinha, S. Bapat, V. Naik, V. Kulathumani, H. Zhang, H. Cao, M. Sridhara, S. Kumar, N. Seddon, C. Anderson, T. Herman, N. Trivedi, C. Zhang, M. Gouda, Y.-R. Choi, M. Nesterenko, R. Shah, S. Kulkarni, M. Aramugam, L. Wang, D. Culler, P. Dutta, C. Sharp, G. Tolle, M. Grimmer, B. Ferriera, K. Parker, "ExScal: Elements of an Extreme Scale Wireless Sensor Network," *In Proceedings of the Eleventh IEEE International Conference on Real-Time Computing Systems and Applications (IEEE RTCSA)*, Hongkong, 2005.

[3] S. Bapat, V. Kulathumani and A. Arora, Analyzing the Yield of ExScal, A Large-Scale Wireless Sensor Network Experiment, *IEEE ICNP*, Boston, 2005.

[4] S.M. Brennan, A.M. Mielke, D.C. Torney, and A.B. McCabe, Radiation Detection with Distributed Sensor Networks, *IEEE Computer*, Vol.37, No.8, August 2004, pp. 57-59.

[5] M. Cardei, M. Thai and W. Wu, Energy-Efficient Target Coverage in Wireless Sensor Networks, *IEEE Infocom*, Miami, FL, 2005.

[6] H. Chen, H. Wu and N.F. Tzeng, Grid-Based Approach for Working Node Selection in Wireless Sensor Networks, *IEEE International Conference on Communications (ICC)*, Paris, FR, 2004.

[7] *Mica2 Series Motes*, http://www.xbow.com, 2004.

[8] M. Horton, D. Culler, K. Pister and J. Hill, R. Szewczyk, and A. Woo, The Commercialization of Microsensor Motes, *Sensors Magazine*, Vol.19 No.4, April 2002, pp. 40-48.

[9] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events, *In Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN)*, Los Angeles, CA, 2005.

[10] J. Elson and D. Estrin, Sensor Networks: A Bridge to the Physical World, in C.S. Raghavendra, K.M. Sivalingam and T. Znati (eds.) *Wireless Sensor Networks*, Boston, Kluwer Academic Publisher, 2004, pp. 3-20.

[11] L. Gu and J.A. Stankovic, Radio Triggered Wake-Up Capability for Sensor Networks, *In Proceedings of Real Time Applications Symposium,* May 2004.

[12] H. Gupta, S.R. Das, and Q. Gu, Connected Sensor Cover: Self-Organization of Sensor Networks for Efficient Query Execution, *In Proceedings of the Fourth International Symposium on Mobile Ad Hoc Networking and Computing (ACM MobiHoc)*, Annapolis, MD, 2003, pp. 189-200.

[13] T. He, S. Krishnamurthy, J.A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, and L. Hu, Energy-Efficient Surveillance System Using Wireless Sensor Networks, *In proceedings of the Second International Conference on Mobile Systems, Applications, and Services (ACM Mobisys)*, Boston, MA, 2004, pp. 270-283.

[14] J. Hill and D. Culler, Mica: A Wireless Platform for Deeply Embedded Networks, *IEEE Micro*, Vol.22 No.6, 2002, pp. 12-24.

[15] J. Hui, Z. Ren, and B. H Krogh, Sentry-Based Power Management in Wireless Sensor Networks, *In Proceedings of the Second International Conference on Information Processing in Sensor Networks (IPSN)*, Palo Alto, CA, 2003.

[16] S. Kumar and A. Arora, Topology and Naming for ExScal Clean Point Demonstration, *ExScal Note Series ExScal-OSU-EN03-2004-12-05*, December 2004.

[17] S. Kumar, T.H. Lai, and J. Balogh, On k-Coverage in a Mostly Sleeping Sensor Network, *In proceedings of the Tenth International Conference on Mobile Computing and Networking (ACM MobiCom)*, Philadelphia, PA, 2004, pp. 144-158.

[18] S. Kumar, T.H. Lai, and A. Arora, "Barrier Coverage with Wireless Sensors," *In Proceedings of the Eleventh Annual International Conference on Mobile Computing and Networking (ACM MobiCom)*, Cologne, Germany, 2005.

[19] K. Martinez, J.K. Hart, and R. Ong, Environmental Sensor Networks *IEEE Computer*, Vol.37, No.8, August 2004, pp. 50-56.

[20] *Telos*, http://www.moteiv.com, 2005.

[21] J. Polastre, J. Hill, and D. Culler, Versatile Low Power Media Access for Wireless Sensor Networks, *In proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Baltimore, MD, 2004.

[22] J. Polastre, R. Szewczyk, A. Mainwaring, D. Culler, and J. Anderson, Analysis of Wireless Sensor Networks for Habitat Monitoring, in C.S. Raghavendra, K.M. Sivalingam and T. Znati (eds.) *Wireless Sensor Networks*, Boston, Kluwer Academic Publisher, 2004, pp. 399-423.

[23] S.M. Ross, *Introduction to Probability Models*, Academic Press, 2001.

[24] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai and K. Frampton, Sensor Network-Based Countersniper System, *In proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Baltimore, MD, 2004.

[25] N. Xu, S. Rangwala, K.K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, A Wireless Sensor Network for Structural Monitoring, *In proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Baltimore, MD, 2004.

[26] Y. Choi, M.G. Gouda, H. Zhang and A. Arora, Routing on a Logical Grid in Sensor Networks, *UTCS Technical Report TR-04-09*, 2004.

[27] H. Zhang and J. Hou, Maintaining Sensing Coverage and Connectivity in Large Sensor Networks, *In Proceedings of the NSF International Workshop on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wirelsss, and Peer-to-Peer Networks*, 2004.

[28] Z. Zhou, S.R. Das, and H. Gupta, Connected k-Coverage problem in Sensor Networks, *In Proceedings of International Conference on Computer Communications and Networks (ICCCN)*, Chicago, IL, 2004.