Analyzing the Yield of *ExScal*, a Large-Scale Wireless Sensor Network Experiment

Sandip Bapat, Vinodkrishnan Kulathumani, Anish Arora
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210, USA

Abstract

Recent experiments have taken steps towards realizing the vision of extremely large wireless sensor networks, the largest of these being ExScal, in which we deployed about 1200 nodes over a $1.3km$ *by* $300m$ *open area. Such experiments remain especially challenging because of: (a) prior observations of failure of sensor network protocols to scale, due to network faults and their spatial and temporal variability, (b) complexity of protocol interaction, (c) lack of sufficient data about faults and variability, even at smaller scales, and (d) current inadequacy of simulation and analytical tools to predict sensor network protocol behavior.*

In this paper, we present detailed data about faults, both anticipated and unanticipated, in ExScal. We also evaluate the impact of these faults on ExScal as well as the design principles that enabled it to satisfy its application requirements despite these faults. We describe the important lessons learnt from the ExScal experiment and suggest services and tools as a further aid to future large scale network deployments.

1 Introduction

With the emergence of new hardware platforms, both research and commercial, as well as the sustained development of software architectures and development tools over the past few years, wireless sensor networks are envisioned to be deployed in scales of 10^5-10^6 nodes in industrial, military, agricultural, medical and several other applications. Based on our experience in building wireless sensor networks, we have seen sensor networks scale in several dimensions as shown in Table 1, the largest of these being *ExScal*, in which we deployed about 1200 nodes over a $1.3km$ by $300m$ open area.

Year	Nodes	Area	Program size
2002	10	10 sq.m.	5KB
2003	100	500 sq.m.	30-100KB
2004	1000	250,000 sq.m.	200KB-2MB

Table 1. Evolution of sensor networks.

This substantial increase in network, deployment and program complexity scales raises the following key concerns:

Failure of sensor network protocols to scale Past experiences in sensor network research [1] have shown that scaling an individual protocol for a network that is 10 times larger often involves redesigning the protocol itself. In addition to node failures themselves, the main causes for failure of wireless sensor network protocols to scale are network faults such as channel contention, interference and fading over the wireless medium. The extent to which these faults affect a network is determined by several factors such as internode separation, antenna polarization, presence of obstacles and the traffic pattern in the network. Network faults therefore have significant spatial and temporal variability making the design of scalable sensor network protocols a challenge.

Complexity of integration Sensor network applications are often built out of multiple protocols for low-level services such as medium access, reliable communication, sensing, time synchronization, etc., and integrating these protocols raises the following challenges. First of all, system correctness is hard to reason about in the presence of complex interactions between the various protocols. Secondly, optimizing the performance of such complex systems often involves simultaneous tuning of parameters across multiple protocols which may have conflicting requirements. For example, increasing the memory allocation for routing buffers may improve communication performance, but perhaps at the cost of memory available for filtering windows in sensory processing, leading to increased noise of false positives.

Lack of sufficient fault data Designing large scale wireless sensor networks is further complicated by the fact that there is little data on faults, their variability or their impact on applications at small and large scales.

Unpredictability of network behavior Due to the lack of sufficient data about faults occurring in a large scale network, there is also a dearth of simulation and analytical tools that realistically model scaling effects. This makes it hard to predict the behavior of sensor network protocols and often forces network designers to be conservative in allocating resources to the network.

Given these scaling concerns, we present in this paper, data and findings from *ExScal*, a complex, multi-tier and multi-phase, large scale sensor network field experiment that we conducted over a 15 day period in December 2004, for detection, classification and tracking of different types of intruders over an extended geographical area of $1.3km$ by $300m$. We analyze the impact of faults on *ExScal* based on the performance or yield of each of its operational phases. We define yield as the ratio of measured performance to the ideal performance.

Goals of the paper
(i) Use real experimental data to identify the types and characteristics of faults that can occur in a large scale sensor network,

(ii) validate our design of *ExScal* by analyzing the variability of these faults and their impact on the overall yield of the network,

(iii) summarize key lessons learnt and identify areas where our design was either too conservative or inadequate, and

(iv) identify additional services and tools to help network managers choose configuration parameters that best satisfy the overall system requirements.

Key findings of the paper We find the overall yield of *ExScal* to be 72.83%. About 6% nodes become completely inoperational over the 15 day period due to environmental effects and these failures have a uniform spatial distribution. The end-to-end network routing yield is 85.61% and by appropriate design of our network protocols we achieve a uniform spatial distribution of network reliability. Since we instrument *ExScal* with adequate redundancy, it is able to tolerate these uniformly distributed faults.

Certain faults such as node localization faults are spatially correlated while others like the failure of a backbone node are likely to induce faults that are spatially correlated. We design our protocols to tolerate such faults. We also encounter some unanticipated faults such as nodes which get stuck during reprogramming and byzantine sensor nodes which may report up to 15 false detections per minute. We handle such unanticipated faults by enforcing high level management policies or by using dynamic reconfiguration to instrument tolerance components such as detectors and correctors.

By handling anticipated faults through design and unanticipated faults with human-assisted management tools, *ExScal* is able to meet its specifications.

However, our design of *ExScal* is conservative due to lack of prior data about faults to predict the network yield at this scale. We find that the excess redundancy in *ExScal* can be used to increase system lifetime.

We also observe that reliable measurement of network state remains a challenge due to the unreliability of multi-hop wireless communications.

Organization of the paper We present previous results related to ours in Sec. 2. In Sec. 3, we describe *ExScal* and its operation. We then discuss the experimental methodologies in Sec. 4. In Sec. 5, we first present yield data for individual subsystems; deployment in Sec. 5.1, reprogramming in Sec. 5.2, localization in Sec. 5.3 and *ExScal Op-Ap* in Sec. 5.4. We then present the net yield of *ExScal* in Sec. 5.5. We discuss key takeaways and lessons learnt in Sec. **??** and make concluding remarks and discuss future work in Sec. 7.

2 Related Work

As described in Sec. 1, the network size and application complexity of sensor network deployments is growing substantially. Existing sensor networks can be categorized into two types: low duty-cycle, periodic sensing applications versus low latency, always-on sensing application. Habitat monitoring [2–4] and structural monitoring [5] are typical examples of the former whereas intrusion detection [1] and shooter localization [6] are some examples of the latter; *ExScal* belongs to the latter class.

Szewczyk et al [4,7] present detailed analysis of experimental results over a 4 month deployment of 150 sensor nodes for habitat monitoring. The key results from this experiment include lifetime analysis and measurements for the deployed system and analysis of routing performance over time. This study also addresses some of the node design and deployment issues and unanticipated faults observed during the experiment. Similar results about other periodic monitoring applications can also be found in [8, 9].

There are significant differences, however, between the requirements for an application like habitat monitoring and one like *ExScal*. *ExScal* is a real-time event detection system as opposed to a periodic sensing applications. As such, it is more sensitive to faults such as false positives, network unreliability and latency. Van Dyck and Miller [10] have proposed a simulation framework and performance criteria for event detection systems. Our study hopes to provide an experimental data point to address some of the issues they identify.

Cerpa et al [3] and others have proposed the use of multi-tier network design to achieve scalability in sensor networks. *ExScal* uses a multi-tier architecture designed to handle thousands of low-level devices and hundreds of higher tier nodes forming their own multi-hop wireless ad hoc network, which is again the largest such experiment till date. ExScal thus provides validation of the scalability of using a multi-tier, hierarchical network.

3 *ExScal* System Overview

In this section, we present an overview of *ExScal* with respect to its requirements, architecture and operation.

3.1 Requirements

The three main requirements of *ExScal* are:

• **Cost effective coverage:** To cover a large region at low cost, *ExScal* needs to scale sensing and communication ranges of nodes, use a topology designed with minimal redundancy to get efficient coverage and maintain low power consumption to maximize system lifetime.

• **Quality of service:** *ExScal* needs to provide accurate and timely performance, i.e. low false positives, false negatives, misclassifications and tracking error in near-real-time execution, and be robust by tolerating deployment errors and component faults.

• **Low human involvement:** To be easily usable even at large scale, *ExScal* needs to have minimal physical human involvement, implying a need for services enabling easy operation, monitoring and reconfiguration of the network.

3.2 Multi-tier design

To guarantee network reliability and keep end-to-end latency bounded, *ExScal* uses a three-tier network design. The lowest tier, Tier-1, consists of XSMs (for eXtreme Scale Motes) [11] which are derivatives of the Mica2 mote [12]. XSMs perform the tasks of sensing and detection using onboard magnetometer, acoustic and PIR (for passive infrared) motion sensors and communicate detected events to a local base mote. Each local base mote aggregates detections from an average of 50 XSMs and is connected to a Tier-2 node called the XSS (for eXtreme Scaling Stargate) [13] through a 51-pin connector interface. XSS nodes have a 400 MHz processor, 64MB RAM and 32MB flash memory and

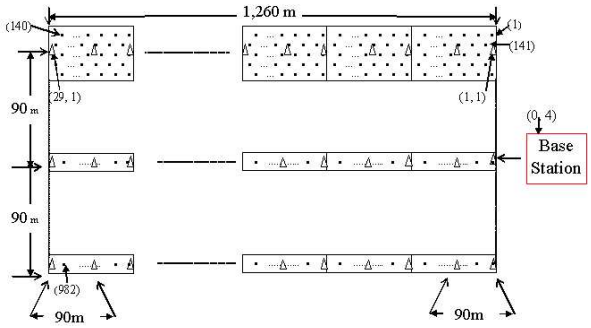


Figure 1. *ExScal* network topology.

are thus more resource-rich than XSMs. Each XSS has a GPS device and can communicate reliably over several hundred meters using a 2.4GHz radio, connected to a 5 ft tall, 9dBi omnidirectional antenna. XSS nodes form their own peer-to-peer ad hoc communication network using the IEEE 802.11b MAC protocol. This network is rooted at a special XSS node that is connected via wired ethernet to a Tier-3 node. The Tier-3 node is a laptop or a PC running the classification, tracking and visualization applications and also serves as the command and control station for network management.

Network Topology Fig. 1 shows the topology of the *ExScal* network which consists of 983 XSMs, represented by dots, arranged in two regions. The dense region at the top consists of 5 rows with 140 XSMs each at a separation of 9m arranged in a regular hexagonal grid. The sparse consists of two XSM lines starting at 90m from the dense region and 90m from each other. These XSM lines enable us to track intruder motion after it has left the dense region. Fig. 1 also shows 45 XSS nodes, represented by triangles, arranged in a 15 x 3 grid with 90m separation. The dense region thus has a total of 686 XSMs and 15 XSSs resulting in about 50 XSM nodes per XSS.

To demonstrate scalability of multi-tier design, *ExScal* also deployed 203 XSS nodes in a 29 x 7 grid over the same area for experiments involving the Tier-2 ad hoc communication protocols [14].

Sensing and communication coverage *ExScal* uses influence field estimation [15] to classify and track detected intruders. Given the sensing ranges for each intruder as shown in Table 2 and an empirically measured rate of false positives, we calculate the minimum density required to distinguish between the intruder types with high probability. However, this calculation does not take into account the effect of any node or network faults. In the absence of prior estimates for these faults, we conservatively select network density to be twice the minimum required. This translates to a 9m internode separation between XSMs in the *ExScal* topology. *ExScal* thus has a 100% planned redundancy in sensing. The communication range of XSMs is 30m, so redundancy in communication is even higher than in sensing.

Intruder type	PIR	Acoustic	Magnetometer
Person	12m	-	-
SUV	25m	30m	7m
ATV	15m	50m	3m

Table 2. Sensing range for intruder types in *ExScal*

Further details about design considerations, coverage calculations and possible optimizations for the *ExScal* topology can be found in [16].

3.3 Multi-phase operation

To manage application complexity, the operation of *ExScal* is broken down into the following phases.

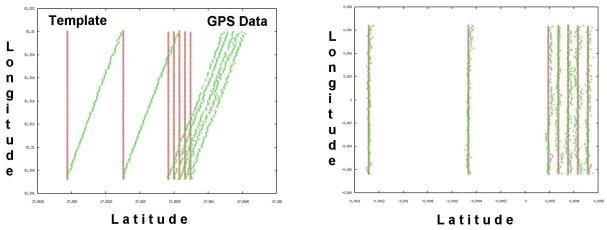
• **Pre-Deployment** The first phase of *ExScal* consists of a default application for all XSMs. This application, which we call *Trusted Base* has the ability to download new programs using the Deluge [17] reprogramming protocol and the ability to perform certain management functions like sleep/wakeup and network health querying using the Sensor Network Management System (SNMS) [18] protocol. The *Trusted Base* also includes a *deployer response* application. This application is enabled when the XSM is turned on during deployment, and sends out *Hello* messages containing the unique identifier of the XSM and emits an audible beep as confirmation of its liveness.

• **Deployment** The deployment process consists of two steps. In the first step, the grid topology is marked on the ground using techniques from civil engineering to an accuracy of a few centimeters. Marked grid points represent ideal node positions. In the second step, human deployers place the XSMs and the XSSs at the marked grid points and power them on. The *Hello* messages sent by the *deployer response* application on an XSM are received by a mote attached to a hand-held XSS node carried by the deployer. The deployer's XSS records the node-id in this message along with the GPS location of that point where the node has been deployed in a file on the XSS. Thus, at the end of the deployment process, the network deployers have a list of node-ids and their corresponding GPS co-ordinates.

• **Reprogramming** The reprogramming phase is used to download new application programs from the Tier-3 node to the entire network and is a recurring phase in *ExScal* operation. Tier-2 applications and protocols run on XSSs as Linux processes hence reprogramming Tier-2 nodes consists of replacing an existing executable with a new one. To download a new program on Tier-1 XSMs, we use the Deluge protocol in the *Trusted Base*. The new Tier-1 program is first downloaded to the XSS nodes which in turn execute the Deluge protocol to download it to the entire XSM network.

• **Localization** The next phase of *ExScal*, which we call *OASLOC* for Operator Assisted Localization, involves localization of deployed nodes. To perform localization, (node-id, GPS) pairs collected by each deployer's hand-held XSS are first downloaded on the central Tier-3 node and merged. This list is then fed to a geometric program which we call *Snap-to-Grid*, running on the Tier-3 node. *Snap-to-Grid* uses a template of ideal grid positions and performs a series of rotation, translation and heuristic-based matching operations to map each node-id to a grid position in the template. Fig. 2 illustrates the operation of *OASLOC*.

A similar strategy is used to localize Tier-2 XSS nodes, the key distinction being that XSS devices directly communicate their node-id and GPS locations to the Tier-3 node using an efficient flooding-based algorithm. The Tier-2 grid positions are communicated directly to the XSSs using the same flooding service, while the



(a) (b)

Figure 2. OASLOC Snap-to-Grid process: The perfectly spaced vertical lines in (a) form the ideal grid template to which the collected GPS positions are mapped in (b).

Tier-1 grid locations are communicated to the XSMs by first sending them to the nearest XSS node which then initiates a controlled flooding algorithm called *Epicast* to forward these to the respective XSMs.

• ***ExScal Op-Ap*** Upon localization, the nodes are ready to execute the main sensing and intrusion detection application, which we call *Op-Ap* for *Operator App*. *Op-Ap* uses a routing protocol called GridRouting [19] to communicate its detections to the local base node. GridRouting uses the output of *OASLOC* to conservatively select a set of potential parents with stable, reliable links for each XSM. The *ExScal Op-Ap* also uses an implicit acknowledgement based retransmission protocol called ReliableComm [20] to improve per-hop reliability. The routing reliability of *Op-Ap* at Tier-1 is thus the reliability provided by GridRouting using ReliableComm. Detections received at a Tier-2 node

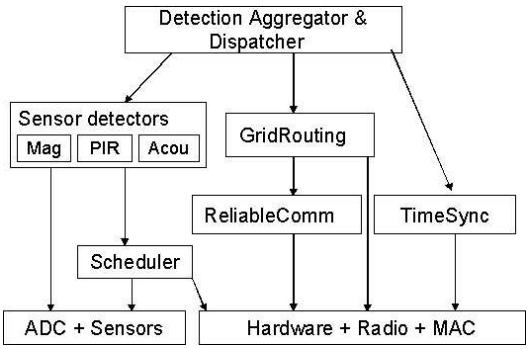


Figure 3. *ExScal Op-ap* Tier-1 component diagram.

are communicated to the central Tier-3 node where they are used to classify and track the detected intruders. This Tier-2 convergecast uses a beacon-free routing protocol called UniComm[1], which uses data traffic to perform link-estimation for selecting next-hop parents.

Fig. 3 shows the component diagram of the *ExScal Op-Ap* at Tier-1.

3.4 Central Command and Control

ExScal provides the network operator with a central command and control console at the Tier-3 node. From this console, the network operator issues commands for reprogramming, power management (sleep/wake-up) or issues queries to collect information about network state. These commands and queries are first communicated over the Tier-2 network to all XSSs which then relay them to the XSM network. Conversely, responses are aggregated from the XSMs at the nearest XSS and sent to the Tier-3 operator console using UniComm.

[1]This paper is currently under anonymous submission, so we are unable to provide a reference

4 Experimental Methodology

In this section, we discuss the setup for the *ExScal* experiments and data collection, and the method used to calculate yield results in this paper.

Data collection from dense region As shown in Fig. 1, the *ExScal* deployment consists of two regions, one dense and the other sparse. Since classification and high-accuracy tracking of intruders are performed in the dense region where network contention effects are worse, we present numerical analysis for data collected from the dense region only so that our results are not affected by the heterogeneity in deployment. The dense region consists of 686 XSMs deployed in 5 rows and 15 XSS-mote pairs. We have made the raw data collected from the entire network over the entire experimentation period publicly available [21].

Phase by phase yield measurement We identify faults occurring in each operational phase of *ExScal* and analyze their impact by measuring the yield of that phase. We also analyze the yield of each tier wherever applicable. Finally, we discuss the impact of the faults in one phase on the yield of subsequent phases.

To calculate yield, we need to know the state of the network – ground truth – at the beginning of each phase. We now describe our methodology to measure the ground truth.

Ground truth methodology ExScal uses the SNMS protocol [18] to collect information about the state of the network by querying certain pre-specified attributes. The SNMS multi-hop querying protocol firsts builds a spanning tree rooted at a base node over which query results are aggregated at the base. Ideally, this network based querying should enable us to gather state information from the whole network accurately and measure the performance of other protocols. However this method is inadequate, as shown in Table 3.

# base stations(B)	# responding nodes(N)	Average N/B
4	117 (17%)	29.25
15	334 (48.7%)	22.27

Table 3. SNMS multi-hop query reliability

Table 3 presents data for two types of SNMS queries with varying number of base stations. The data for number of responding nodes has been averaged over multiple basic SNMS queries wherein each node replies with its node-id. This data indicates that the number of query responses increases with the number of base stations used. We thus obtain the most responses when each of the 15 XSS-mote pairs is used as a base station for an SNMS query tree. Even so, on average, an SNMS query only returns information from about half of the deployed network.

The data presented above illustrates that while network querying is useful to gain some confidence about the state of an arbitrary network, the reliability of multi-hop querying using SNMS is inadequate for

ground truth measurement. We also find that multiple executions of an SNMS query only increase its reliability marginally.

We therefore use application data from the *Op-Ap* phase along with results of multiple SNMS queries. We instrument additional information to be part of the application data that helps infer the yield of previous phases. By piggybacking this information on application traffic, we eliminate the need for additional messages for network state measurement. Moreover, since we choose the routing protocol that gives us the best reliability in each application phase, we are able to exploit this higher reliability for collecting ground truth data.

Inference methodology Upon detecting an intruder, an XSM sends out a message to its local tier-2 base station that contains its grid location, a sequence number and a timestamp. An XSM receives its grid location through the localization phase. If a node does not have this grid location due to localization faults, it uses its unique software id assigned in the *Trusted Base*. Nodes using their software id are thus inferred to be affected by localization faults. Similarly, missing sequence numbers in messages received from a node provide information about the network reliability.

At Tier-2, each base station appends its own id, sequence number and timestamp, and forwards the message to the Tier-3 node. This information is used to calculate the reliability at Tier-2 in a similar manner.

The application of this approach is illustrated in more detail in the following sections where we calculate the yield of each operational phase in *ExScal*.

5 *ExScal* Yield Analysis

In this section, we present data about faults occurring in each application phase and measure their impact. We then calculate the net yield of *ExScal* and show that it meets its requirements.

5.1 Deployment Phase

ExScal is designed for an outdoor setting with a large number of (previously untested) devices, hence deployed nodes are subject to environmental elements. Over the 15 day experimentation period, deployed nodes were left in an open area where they were exposed to passing vehicles or wildlife that crushed some, heavy rain that caused leakages and heavy wind that toppled some and reduced their communication range. A more serious fault was the failure of an XSS device since a single XSS was responsible for aggregating data from roughly 50 XSM nodes. We say a node is *up* if it is known to be working in at least one application phase and *failed* otherwise.

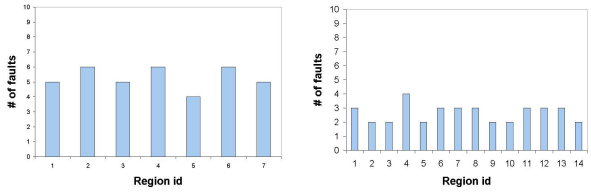
Experiment design We first measure ground truth for a section of 100 XSM nodes. This 100 node section is small enough that we can reliably collect fault data from it on an ongoing basis, yet large enough to capture most interesting faults that could occur in other similar sections. We then extrapolate the data from this section to estimate the fault rate for the whole network. To verify this estimate, we use the methodology described in Sec. 4. Specifically, we log all messages received from the network during the entire 15 day period. These messages are generated during different application phases and communicated using different routing protocols. We then count all unique nodes from which at least one message is received, i.e. the number of *up* nodes as defined above. This number is then compared to the estimated value to derive a lower bound on the yield of the deployment phase. The Tier-2 XSS nodes, being much smaller in number and having more resources, are wirelessly accessible for collecting ground truth data about faults.

# nodes (XSM + XSS)	701 (686 + 15)
# *failed* XSMs in one section	5
Estimated # failed XSMs	35
# *failed* XSSs	1
# *up* nodes	647

Table 4. Deployment fault data

Results Table 4 compares the estimated and observed values for deployment yield. Based on the fault data collected for one section, the number of XSM failures for the whole network is then estimated to be 35 implying that 651 XSMs should be *up*. It can be seen from Table 4 that the measured yield (647 nodes) closely matches the estimated value (651 nodes).

Result 5.1(a) The yield of the ExScal deployment phase is 94.31%; implying a fault rate of at most 5.69%.



(a) 100 node region (b) 50 node region

Figure 4. Spatial distribution of deployment faults

We also calculate histograms of the number of deployment faults in different regions to obtain their spatial distribution. Fig. 4 plots two such histograms for regions of sizes 100 nodes and 50 nodes respectively. The flat shape of both histograms in Fig. 4 offers evidence that the spatial distribution of deployment faults in *ExScal* is uniform.

Result 5.1(b) Deployment faults are spatially uniform.

Observation 5.1(c) We observe deployment failures to have a temporal distribution as well. At the time of deployment, each node is verified to be *up* using the audible beep in the *deployer response* application. In the 100 node section that we closely monitor, we discovered 1 *failed* node after 3 days, 3 *failed* nodes after a week and 5 *failed* nodes at the end of the 15 day period, indicating a temporal attrition rate for deployed nodes.

Impact of deployment faults A failed node in the *ExScal* network cannot participate in any of the remaining phases of the application. Deployment faults are thus additive and impose an upper bound on the net yield.

5.2 Reprogramming Phase

Recall that the Deluge protocol in the *Trusted Base* is used to reprogram XSMs with a new application image. Deluge divides an application image into smaller *pages* which are downloaded one at a time and stored in the external flash on an XSM. An XSM can be rebooted to this image only if it has downloaded all pages correctly. Deluge is epidemic, so a node with a partial application image continually tries to download missing pages from neighbors that may have them. We identify two types of reprogramming faults observed in *ExScal*.

• *Initialization faults*: We say a node has an *initialization fault* if it cannot execute the Deluge protocol due to flash initialization errors during startup.

Restarting nodes with *initialization faults* often results in successful flash re-initialization, however, it is not feasible to detect and restart individual failed nodes at large scale. Also, since this fault is *unanticipated* and occurs in the *Trusted Base* which cannot be reprogrammed, we are unable to instrument detector/correctors to recover from this state.

• *Lagger nodes*: We say a node is a *lagger* if it can participate in the Deluge protocol, but progresses at a much slower rate than its neighbors. In the worst case, a *lagger* is always stuck trying to download the first page of the new image.

Laggers have a potentially disastrous effect on the network due to the epidemic nature of Deluge; they reduce the lifetime of nodes in their neighborhood due to their repeated requests for message transmissions as well as flash operations. Our offline measurements show that the current drawn by an XSM is nearly doubled due to extra message transmission and flash read operations. Also, the number of neighbors for an XSM in the *ExScal* topology is between 10 and 20. Thus, even a small fraction of *laggers* can significantly reduce the lifetime of a large number of nodes. Another problem caused by *laggers* is persistent reprogramming traffic in the network that causes higher contention for application messages. This leads to reduced reliability, increased latency and degraded application performance.

Experiment design The network operator of *ExScal* downloads a new application image on XSMs running the *Trusted Base* application. The operator then queries the network repeatedly to monitor the progress of this download over the network. As described in Sec. 4, this query provides only statistical assurance that the network has completed the download process. The operator then issues a reboot command to switch from the *Trusted Base* into the desired application image. As before, the yield of reprogramming cannot be measured directly by querying the network; it is inferred from data collected in the subsequent application phase. Specifically, we measure the number of XSMs from which application messages for the newly downloaded program are received. This represents the lower bound for reprogramming yield since these XSMs must have finished their download to be running the new application.

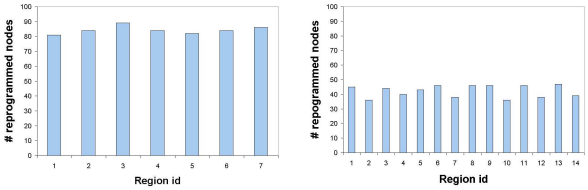
Results The data collected by repeated querying of XSMs during program downloads shows that 5% of the XSMs are affected by *initialization faults* and thereby cannot download new applications. We find the number of laggers to be 0.5%. Finally, the data collected from the subsequent application phase contains messages from 93% of the *up* nodes.

Result 5.2(a) The yield of the ExScal reprogramming phase is 93%; with 5% initialization faults and 0.5% lagger nodes.

Although *laggers* are an unanticipated fault, our multi-phase design of *ExScal* helps contain their impact. Fault containment is achieved through the management policy of including Deluge only in the *Trusted Base* and not in other applications. Thus, when the operator issues a reboot command to start the next phase, *laggers* no longer have neighbors that can satisfy their Deluge requests and become quiescent. Note that since network querying only gives us statistical estimates of network state, some non-faulty nodes could still be in the download process when the operator issues the reboot command. However, from Result 5.2(a) we see that the number of such nodes can be at most 1.5%. This penalty is easily outweighed by the energy and performance benefits for nodes that would otherwise be adversely affected by *laggers*.

Result 5.2(b) Reprogramming faults are contained in ExScal by policy based management; the performance penalty for this containment is at most 1.5%.

Reprogramming faults follow a random uniform spatial distribution as expected, since both *initialization faults* which occur due to timing errors at startup and *laggers* which occur due to hardware flash errors or low battery are random events. We again calculate the histograms for number of correctly reprogrammed nodes in regions of different sizes to study the spatial distribution of reprogramming faults. Fig. 5 shows the histograms for regions of 100 and 50 nodes respectively and once again offers evidence of the spatial uniformity of reprogramming faults.



(a) 100 node region (b) 50 node region

Figure 5. Spatial distribution of reprogramming

Result 5.2(c) Reprogramming faults are spatially uniform

Impact of reprogramming faults Reprogramming faults result in unavailability of affected XSMs for running the subsequent application phase.

5.3 Localization

As described in Sec. 3, *ExScal* uses *OASLOC* for localizing deployed nodes. We observe the following types of faults during *OASLOC*.

• *Hole fault*: We say a node has a *hole fault* if it is not assigned a grid position by the localization protocol.

Hole faults occur as a result of deployment errors such as failing to turn on a deployed node or not waiting long enough for the hand-held GPS device to obtain

an accurate reading. A burst of *hole faults* may occur in in regions with poor GPS reception.

• *Mapping fault*: We say a node has a *mapping fault* if it is assigned a grid position other than the one it is deployed at.

Mapping faults occur as a result of drifts in GPS values collected for nodes spaced only 9m apart and due to the heuristics in the *Snap-to-Grid* algorithm. A burst of *mapping faults* may occur when GPS drifts cause multiple successive nodes to appear shifted from their actual locations.

• *Network fault*: We say a node has a *network fault* if it does not receive its assigned grid position.

A *network fault* may occur due to one of two reasons, either the node fails to execute the *Epicast* protocol due to a reprogramming fault, or a loss is encountered during *Epicast*.

Experiment design Localization yield is calculated as follows. During the deployment phase, in addition to the (id,GPS) data collected on the hand-held XSS, network deployers manually record the order in which XSMs are deployed. The output of the *Snap-to-Grid* program is then compared with this ground truth to detect *holes* and *mapping faults*. The spatial distribution of these faults is used to calculate the burstiness of each type. As before, *network faults* cannot be directly measured and are inferred from messages received in the *Op-Ap* phase as follows. *Op-Ap* uses a node's grid position as the source address in its messages. If a node has not received its grid position, it uses its software id whose domain is different from the domain of grid positions. Thus, by observing the source field for messages received during the *Op-Ap* phase, we calculate the number of nodes affected by *network faults*.

Fault type	**# affected nodes**	**Burst size**		
		Min	Avg	Max
Hole	6.1%	1	1.4	6
Mapping fault	3.2%	1	1.7	8
Network fault	2.1%	1	1	1
Total	11.4%			

Table 5. OASLOC fault data.

Results Table 5 shows the number of nodes affected by each type of localization fault. We find that the maximum distance by which a node affected by a mapping fault is displaced is 10m or one diagonal grid position.

Result 5.3 The measured yield of the ExScal localization phase is 88.6%; implying that 11.4% nodes were affected by localization faults.

Impact of Localization Faults As described in Sec. 3, GridRouting in *Op-Ap* uses the output of *OASLOC* to compute a set of potential parents with stable, reliable links for each node. Nodes affected by localization faults can thus have no parents or incorrect parents assigned to them. Although GridRouting is designed to operate in the presence of localization faults, its performance is marginally affected due to sub-optimal choice of links. Also, as seen from Table 5, localization faults are occasionally bursty, although in general they too occur randomly. However, as we shall see in the next subsection, GridRouting preserves uniformity of routing despite the presence of occasional bursts of localization faults.

5.4 *Op-Ap* Phase

Recall that *Op-Ap* performs the tasks of detecting intruders and communicating its detections to the central Tier-3 node. We identify two types of faults observed during the *Op-Ap* phase.

• *Byzantine sensor*: We say a sensor node is byzantine if its rate of false detections exceeds a certain threshold.

A sensor may become *byzantine* if it is deployed in adverse environments like tall grass, where wind-induced motion may trigger false PIR detections. Extreme heat, rain and wind may also lead to persistent false positives. Finally, nodes may observe arbitrary values while sensing if their battery voltage falls below a critical value. Just like *laggers* in reprogramming, *byzantine sensors* impose high load on other nodes that have to route their false detections and create network traffic that interferes with real detection messages.

• *Routing fault*: We say a message is affected by a routing fault if it does not reach the central Tier-3 node.

Routing faults occur due to fading, contention and congestion in the network and may occur at Tier-1 or at Tier-2.

Experiment design To determine the number of *byzantine sensors*, we instrument each node with a detector that continually monitors the rate of detections in that node. Whenever this rate exceeds a threshold determined by the expected level of intruder activity in the network, the detector flags this node as a *byzantine sensor*, logs this detection to flash and also sends a diagnostic message to the Tier-3 node. Likewise, if the rate falls below the threshold, the detector unflags this node. We measure the number of such diagnostic messages in the network to determine the number of *byzantine sensors*.

To measure *routing faults*, we instrument each node to send messages according to some traffic pattern. Each message contains the id of the node, a sequence number, a time-stamp and a traffic pattern identifier. These messages are first received and logged at the local base-XSS node and then sent to the Tier-3 node where they are also logged. *Routing faults* at each tier and for end-to-end communication are then computed by taking the ratio of the number of received messages to the number of sent messages.

Traffic patterns The basic traffic pattern in *ExScal Op-Ap* is one in which a set of nodes detecting an intruder report their detections to the nearest Tier-2 node, which then forwards them to the Tier-3 classifier and tracker. While this traffic pattern is suitable for event detection systems, it may not necessarily apply to other kinds of applications. Also, in practice, periodic traffic like timesync and routing beacons, or bursty traffic like false positives, interferes with *Op-Ap* detection messages. We select three traffic patterns as defined below, which we feel are general enough to be used in other types of sensor network applications and are better indicators of routing reliability in *Op-Ap*.

• *Low-freq*: In the low-freq traffic model, each XSM sends a message to the Tier-3 node every 120 seconds.

• *High-freq*: In the high-freq traffic model, each XSM sends a message to the Tier-3 node every 20 seconds.

• *Bursty*: In the bursty traffic model, each XSM sends a message at the same time, creating a message burst. The bursty traffic pattern may be applicable for event detections that are geographically far-reaching, e.g., acoustic shooter localization [6], or caused by a network wide false alarm, e.g., thunder or an airplane triggering acoustic detections across a large region.

Note that in all 3 experiments, no aggregation at Tier-2 is used. Aggregation reduces Tier-2 traffic to very few, if not a single message, in which case UniComm has been shown to be almost 100% reliable.

Remark on message generation. For the low-freq and high-freq traffic patterns, each XSM uses a local timer to generate periodic messages. In the bursty case, a command is flooded from the Tier-3 node to all Tier-2 nodes which then flood the Tier-1 network with this message. Nodes wait a fixed amount of time for the flood to subside before sending their message. Our experimental measurements show that the effects of the reliability and latency of the command message propagation on traffic burstiness are relatively insignificant.

Results We measure the number of *byzantine sensors* in the network to be 1% and the rate of false detections in a *byzantine sensor* to be as high as 15 per minute. We also observe the distribution of *byzantine* nodes in the network to be randomly uniform. Note that correlated fault positives due to phenomena such as thunder or an airplane do not constitute *byzantine* behavior. We also instrument each node with a corrector that suppresses messages for false detections within a node which is flagged as *byzantine*. This detector/corrector pair thus minimizes the impact of a *byzantine sensor* on other nodes.

Result 5.4(a) The number of byzantine sensors measured in ExScal Op-Ap is 1%. Byzantine sensor faults are contained in ExScal by instrumenting detectors and correctors.

Table 6 lists the routing yield in terms of the net average routing reliability and the average routing reliability for nodes with and without localization faults, as measured at the Tier-1 base station mote. The data in Table 6 shows that the low-freq traffic has

	Low-freq	High-freq	Bursty
Net average reliability	86.72%	58.32%	60.17%
Average reliability			
with no localization fault	89.36%	70.95%	79.7%
with localization faults	74.74%	50.97%	60.35%

Table 6. Yield of *ExScal Op-Ap* Tier-1 routing

the highest average reliability and the least variance as expected. The traffic load generated by intruders in *ExScal* is in fact lower than in the low-freq case. The common case Tier-1 routing reliability of *Op-Ap* is thus even higher and more uniform than in Table 6. We therefore use the yield obtained for the low-freq case as the yield of the *ExScal Op-Ap*. Fig. 6 plots the average routing reliability at Tier-1 as a function of the distance from the local base node. The maximum distance in this graph represents the farthest distance between an XSM and its nearest base-XSS node. Although reliability gradually decreases with increasing distance, we again observe that the reliability distribution is uniform for low-freq traffic. Fig. 6 also offers evidence that despite localization faults which may be non-uniform, the *Op-Ap* phase has high and

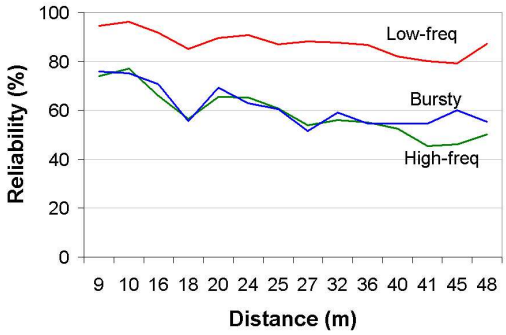


Figure 6. Routing reliability distribution at Tier-1

uniform routing reliability.

Result 5.4(a) The routing yield at Tier-1 for ExScal Op-Ap is 86.72% and has a uniform spatial distribution. Localization faults have a marginal impact on the yield of Op-Ap routing.

End-to-end reliability Table 7 lists the end-to-end routing performance of *ExScal* and its breakdown at both tiers. It can be seen that Tier-2 has high routing

	Low-freq	High-freq	Bursty
Tier-1 reliability	86.72%	58.32%	60.17%
Tier-2 reliability	98.73%	94.55%	96.86%
End-to-end reliability	85.61%	55.14%	58.28%

Table 7. End-to-end routing yield

reliability for all three traffic loads. Also, as seen in Fig. 7, the routing reliability at Tier-2 is uniform across the entire 1.26 km distance. The end-to-end routing reliability thus has a distribution similar to the one at Tier-1 as shown in Fig. 6.

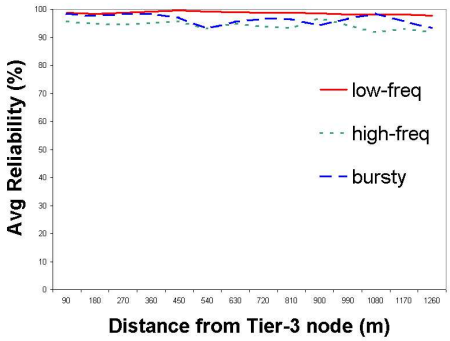


Figure 7. Routing reliability distribution at Tier-2

Result 5.4(c) The Tier-2 routing yield for ExScal Op-Ap is 98.32%, the end-to-end routing yield is 85.27% and its distribution is uniform across the entire ExScal network.

5.5 Net *ExScal* Yield

We now derive the net yield of *ExScal* and validate that *ExScal* meets its requirements. Recall that both deployment and reprogramming faults are uniformly distributed and that even though some localization faults may be non-uniform, the net routing reliability has a uniform distribution. The net impact of faults occurring in all of the *ExScal* phases on its performance

is thus uniform. The net *ExScal* yield can therefore be calculated by normalizing the measured yield of *Op-Ap*, i.e., the end-to-end routing yield at low-freq, over the entire 686 node network, which represents the ideal yield.

Result 5.5(a) The net yield of ExScal, measured in the presence of faults occurring in all operational phases, is 72.83%

Recall from Sec. 3 that ExScal is designed with 100% redundancy. This redundancy is more than adequate for ExScal to meet its requirements despite faults occurring in each phase that reduce its net yield to 72.83%. We discuss how the excess deployed redundancy of *ExScal* can be exploited in the next section.

6 Lessons learnt

In this section, we discuss some of the key lessons learnt from the *ExScal* experiment relating to its design principles and alternative design options. We also discuss important issues and challenges identified during *ExScal* that need to be addressed in large scale wireless sensor network design.

Successful design principles in *ExScal*

The reasons for *ExScal* satisfying its requirements can be traced to the use of the following design principles.

• *Planned architecture to reduce cost* A key challenge in designing large scale networks is to achieve a balance between hard performance guarantees required by the user and minimizing cost and effort required to deploy and manage the network. Researchers sometimes assume random or probabilistic models about deployment during protocol design. Our experience with *ExScal* shows that a planned, deterministic deployment is not unreasonable even for large scale networks. In fact, it is observed in [22] that we would need a significantly higher number of nodes to achieve the same coverage guarantees in *ExScal* if we use random deployment. In practice, random deployment may also not often be any easier than a planned deployment. We observe the planned system architecture of *ExScal* to be more efficient for deployment, management and operation and more predictable in terms of overall performance.

• *Multi-phase operation for performance optimization and fault containment* *ExScal* is a complex composition of multiple subsystems for intrusion detection, reprogramming, localization, management, etc. Put together, the resource requirements of these subsystems exceed the resources available on an XSM. Dividing *ExScal* operation into several phases thus allows us to satisfy the processing, communication and memory requirements of each phase. Multi-phase operation also allows each subsystem to choose protocols such as routing that are optimized for its own performance whereas resource constraints would perhaps force us to choose a single, common protocol for all subsystems otherwise. This design also has the advantage of fault isolation as observed in *ExScal* where potentially severe reprogramming faults are contained by phase transition.

• *Multi-tier design for reliability* Studies about the performance of multi-hop routing in wireless sensor networks show that while different routing schemes are well-suited for different topologies and traffic patterns, network reliability drops significantly as network size increases beyond 5-6 hops. *ExScal* uses a multi-tier network design to limit the number of hops travelled by a message at each tier. The maximum number of hops traversed by a message at Tier-1 in *ExScal* is 4 in the absence of faults. Projecting the distribution of reliability obtained in *Op-Ap* over distance, we estimate that Tier-1 reliability would drop to less than 10% over a 500m distance. The hierarchical topology in *ExScal* thus helps to bound the unreliability in the network. Even if schemes such as in-network processing are used by the *Op-Ap* to process information locally, end-to-end reliability is still an issue for collecting network state information at the operator console.

Implementation problems in *ExScal*

Our experiments revealed an implementation bug in *ExScal* that resulted in loss of *Op-Ap* message communication for 21 nodes or roughly 3% of the network due to failure of a Tier-2 node. One of the design goals in *ExScal* was to separate network functions from device names to be able to bind them flexibly. However, in the implementation of *OASLOC*, a Tier-2 XSS node was inadvertently hard-coded to propagate localization information only for nodes in its region. This implementation bug impacted nodes in the vicinity of the failed XSS in that they did not receive their grid positions. Consequently, not only did *Op-Ap* detection messages from these nodes have to traverse longer distances to reach the next XSS, but they were also forced to do so along sub-optimal paths.

Improving *ExScal* performance

• *Using available redundancy to increase lifetime* As described in Sec. 3, lack of prior data forced us to be conservative while choosing sensing and communication coverages for ExScal. However, the net yield of ExScal calculated in Sec. 5 exceeds this planned redundancy. The excess redundancy in *ExScal* can be used to extend its lifetime. Various power management schemes for extending system lifetime have been proposed for applications like *ExScal* [23, 24]. An analysis of how and by how much the lifetime of *ExScal* can be extended using such schemes can be found in [25].

• *Additional services and tools for management* A key challenge faced during the design, implementation and operation of *ExScal* was the lack of ground truth information about the network. This is due to the unreliability of current protocols for querying network state. Since different routing protocols are well-suited for different topologies and traffic patterns, the performance of network querying can be improved by dynamically choosing a routing protocol that works best for the given network conditions. Another alternative, which we used in *ExScal* is to use application data itself to infer ground truth.

Even if ground truth information is available, a challenging task for a network manager is to maintain the network in a configuration that best satisfies application requirements under dynamic network conditions. For example, if the number of active sensors in a region falls below a critical threshold due to node faults, the manager has to activate other sensors in that region to maintain sensing coverage. We identify the need for automated, online filtering of network data to extract meaningful information such as perturbations in network state or patterns in network behavior that may be indicators of faults and identify alternate parameters to restore network state.

We also identify a need for greater local and autonomous management support. An example of such an autonomous management technique is the use of policy-based monitoring for dealing with false positives in *ExScal*, which requires minimal human support for specifying the policy and its associated detection and correction actions.

7 Conclusions and Future Work

In this paper, we presented data about faults occurring in *ExScal*, which is likely the largest wireless sensor network experiment performed till date in terms of network size and application complexity. We presented detailed experimental results to derive the yield of individual *ExScal* subsystems and the overall yield of the ExScal experiment in the presence of these faults. These results were obtained using an experimental data set containing more than 100,000 observations which forms only a part of the total experimental data collected for all of the individual subsystems which put together exceeds 1 million records, which we have made available publicly.

We also identified several key challenges not only in the design and implementation of such large scale networks, but also in evaluating the performance of these systems and proposed novel methods for dealing with both. We validated the design principles used in *ExScal* by demonstrating how ExScal satisfied its requirements.

As demonstrated by *ExScal*, a planned system architecture is not only easy to design, implement, deploy and manage but it is also more efficient in terms of cost and performance. There may be cases though where such planned deployments may not be feasible. In the future, we plan to study the impact of relaxing the assumptions made in a planned system on each of the *ExScal* protocols and their overall impact on the yield of the system. We also plan to continue working on both human-assisted and autonomous management services identified in this paper using self-stabilization techniques. Finally, we hope to provide tools for performing large scale sensing and communication experiments in real time using our 420 node, remotely accessible, indoor testbed which we call *Kansei* [26].

References

[1] A.Arora, P.Dutta, S.Bapat, and V.Kulathumani et al. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks, Special Issue on Military Communications Systems and Technologies*, 46(5):605–634, 2004.

[2] A.Mainwaring, J.Polastre, R.Szewczyk, D.Culler, and J.Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of WSNA'02*, Atlanta, GA, September 2002.

[3] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology, 2001.

[4] R.Szewczyk, J.Polastre, A.Mainwaring, and D.Culler. Lessons from a sensor network expedition. In *Proceedings of EWSN'04*, January 2004.

[5] N.Xu, S.Rangwala, K.Chintalapudi, D.Ganesan, Al.Broad, R.Govindan, and D.Estrin. A wireless sensor network for structural monitoring. In *Proceedings of SenSys'04*, pages 13–24. ACM Press, 2004.

[6] 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 SenSys'04*.

[7] R.Szewczyk, A.Mainwaring, J.Anderson, and D.Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of SenSys'04)*, November 2004.

[8] P.Zhang, C.M.Sadler, S.A.Lyon, and M.Martonosi. Hardware design experiences in zebranet. In *Proceedings of SenSys'04*, pages 227–238, New York, NY, USA, 2004. ACM Press.

[9] G.Werner-Allen, J.Johnson, M.Ruiz, J.Lees, and M.Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Proceedings of EWSN'04*, January 2004.

[10] R. E. Van Dyck and L. E. Miller. Distributed sensor processing over an ad hoc wireless network: simulation framework and performance criteria. In *MILCOM, Washington,DC, USA*, October 2001.

[11] 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 IPSN'05*.

[12] Crossbow Technology Inc. MPR400/410/420 MICA2 Mote. `http://www.xbow.com/Products/productsdetails.aspx?sid=72`.

[13] Crossbow Technology Inc. Stargate Gateway (SPB400). `http://www.xbow.com/Products/productsdetails.aspx?sid=85`.

[14] A. Arora, P. Sinha, E. Ertin V. Naik, H. Zhang, M. Sridharan, and S. Bapat. ExScal Backbone Network Architecture. To appear as poster in MobiSys'05.

[15] Sandip Bapat, Vinodkrishnan Kulathumani, and Anish Arora. Reliable estimation of influence fields in unreliable sensor networks. *OSU Technical Report OSU-CISRC-8/04-TR49*, 2004.

[16] S.Kumar and A.Arora. ExScal Topology for Node Deployment, ExScal Note Series, ExScal-OSU-EN00-2004-01-30. .

[17] J.Hui and D.Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of SenSys'04*. ACM Press.

[18] G.Tolle and D.Culler. Design of an application-cooperative management system for wireless sensor networks. In *Proceedings of the EWSN'04*.

[19] Y.Choi, M.Gouda, H.Zhang, and A.Arora. Routing on a logical grid in sensor networks. Technical Report TR04-49, The University of Texas at Austin, 2004.

[20] H.Zhang, A.Arora, Y.Choi, and M.Gouda. Reliable bursty convergecast in wireless sensor networks. In *Proceedings of MobiHoc'05*.

[21] OSU NEST ExScal Team. Experimental data from ExScal. `http://www.cse.ohio-state.edu/~bapat/exscaldata`.

[22] S.Kumar, T.H.Lai, and J.Balogh. On k-coverage in a mostly sleeping sensor network. In *Proceedings of MobiComm 2004*, pages 144–158.

[23] Q.Cao, T.Abdelzaher, T.He, and J.Stankovic. Towards optimal sleep scheduling in sensor networks for rare event detection. In *Proceedings of IPSN'05*.

[24] J.Polastre, J.Hill, and D.Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of SenSys'04*.

[25] S.Kumar, A.Arora, and T.H.Lai. On the Lifetime Analysis of Always-On Wireless Sensor Network Applications. Technical Report OSU-CISRC-5/05-TR28, The Ohio State University, 2005.

[26] OSU NEST ExScal Team. Kansei: Sensor Testbed For At-Scale Experiments. `http://www.cse.ohio-state.edu/kansei`.