Design and Deployment of Sensor Network for Real-Time High-Fidelity Volcano Monitoring

Wen-Zhan Song, Member, IEEE, Renjie Huang, Student Member, IEEE, Mingsen Xu, Behrooz A. Shirazi, Member, IEEE, and Richard LaHusen

Abstract—This paper presents the design and deployment experience of an air-dropped wireless sensor network for volcano hazard monitoring. The deployment of five self-contained stations into the rugged crater of Mount St. Helens only took one hour with a helicopter. The stations communicate with each other through an amplified 802.15.4 radio and establish a self-forming and self-healing multihop wireless network. The transmit distance between stations was up to 8 km with favorable topography. Each sensor station collects and delivers real-time continuous seismic, infrasonic, lightning, GPS raw data to a gateway. The main contribution of this paper is the design of a robust sensor network optimized for rapid deployment during periods of volcanic unrest and provide real-time long-term volcano monitoring. The system supports UTC-time-synchronized data acquisition with 1 ms accuracy, and is remotely configurable. It has been tested in the lab environment, the outdoor campus, and the volcano crater. Despite the heavy rain, snow, and ice as well as gusts exceeding 160 km per hour, the sensor network has achieved a remarkable packet delivery ratio above 99 percent with an overall system uptime of about 93.8 percent over the 1.5 months evaluation period after deployment. Our initial deployment experiences with the system demonstrated to discipline scientists that a low-cost sensor network system can support real-time monitoring in extremely harsh environments.

Index Terms—SensorWeb, volcano monitoring, design and deployment.

1 INTRODUCTION

In the last 15 years, volcanic eruptions have killed more than 29,000 people and caused billions of dollars in damage [5]. Permanent conventional monitoring stations typically send their data from a single sensor to an observatory via analog or digital telemetry. The amount of data transmitted is limited by the bandwidth of the hardware and hardships of siting telemetry links. As a result, many threatening and active volcanoes maintain networks of fewer than 10 stations. Many scientists frequently lack sufficient real-time and high-fidelity data for volcano analysis and eruption prediction [26]. Wireless sensor networks have the potential to greatly enhance the understanding of volcano activities by allowing large distributed deployments of sensor nodes in difficult-to-reach or hazardous areas. Mesh wireless communication permits sensor nodes to communicate with each other and to a central base station via a smart self-healing multihop network, allowing intelligent real-time data reduction as well as retasking the sensor array after deployment. It takes U.S. Geological Survey (USGS) personnel several hours, or even days, to deploy a single monitoring station as it requires scientists to access the site, dig a hole, bury sensors, and install station infrastructure. In contrast, a self-contained smart sensor network makes rapid air-drop deployment possible.

An active volcano provides a challenging environment to examine and advance sensor network technology. The crater is a 3D environment with very rugged terrain (Fig. 1), with diverse seismic and geophysical signals from rock avalanches, landslides, earthquakes, and gas/steam emissions. Volcanic eruptions or explosions may even destroy stations. The occasional eruptions, as well as the heavy rain, snow, ice, and wind weather conditions pose significant challenges on the network’s robustness and self-organizing/self-healing ability [15], [16], [35]. We use one-year supply of non-rechargeable batteries on our low power system, because solar panels do not work with ash cover. Various geophysical and geochemical sensors generate continuous high-fidelity data, and there is a compelling need for real-time raw data for volcano eruption prediction research. For such a high data rate application, a key challenge is how to collect high-fidelity data subject to the limited bandwidth available to sensor nodes. In addition to the limited physical bit rate of the radios used on those low power platforms, radio links may experience packet loss due to congestion, interference, and multipath effects. These problems are exacerbated over multihop networking.

In this paper, we present our system design and deployment experiences of a sensor network for real-time high-fidelity volcano monitoring. The design goal is to support one-year continuous operation in the volcano environment. To achieve this goal, we designed a comprehensive system with a number of features, such as robust communication stacks, intelligent sensing components, hybrid time synchronization protocols, and lightweight network management tools. The system design has been
successfully tested in the lab environment, the outdoor campus environment, and the volcano crater as well. On 15 October 2008, we air-dropped five monitoring stations into the crater of Mount St. Helens, which took only 1 hour. These stations formed a multihop network and immediately began delivering real-time continuous seismic, infrasonic, lightning, and GPS raw data to the control center. Despite the harsh weather conditions (e.g., heavy rain, snow, icing, and gusts) in the 1.5 months after deployment, the sensor network achieves a remarkable packet delivery ratio above 99 percent and the overall system uptime is about 93.8 percent. Many development and management lessons have been learned and documented. The ground sensor network described here will be integrated with NASA EO-1 sensorWeb [3] and USGS science system to form an Optimized Autonomous Space In situ SensorWeb (OASIS) [1] for earth hazard monitoring. The OASIS system integrates space and in situ sensors into a semiclosed loop and feeds information into an earth science decision system. More information about the OASIS system can be found at [1].

The main contribution of this paper is the design of a robust sensor network optimized for rapid deployment during periods of volcanic unrest and provide real-time long-term volcano monitoring. The presented evaluation results demonstrate that the network achieves a remarkable reliability despite harsh weather conditions. The data quality meets the demand of USGS scientists. The presented system has broader impacts beyond volcano monitoring. Many seismic exploration applications, such as oilfield seismic explorations [7], have similar requirements: time-synchronized high-fidelity data acquisition with remote configurability. Partial results of this paper were published in [10].

The rest of this paper is organized as follows: Section 2 presents the system design of hardware and software components. Section 3 describes our campus test experiences and findings. In Section 4, we present our field deployment experience on Mount St. Helens and evaluate the performance of the system. In Section 5, we summarize design and deployment lessons. Section 6 discusses related work. Section 7 concludes the paper with our future plans.

2 SYSTEM DESIGN

The USGS scientists specified the following system requirements:

- **Synchronized Sampling**: To utilize the temporal and spatial correlation of volcano signals, the earth scientists require that all stations perform synchronized sampling and timestamp recorded signals with precise UTC time, with error no more than 1 millisecond.
- **Real-Time Continuous Raw Data**: There are no existing algorithms or models that are widely accepted to predict volcano eruptions, and hence the scientists need real-time continuous raw data to study the behavior of volcanoes.
- **One-Year Robust Operation**: The sensor network must be able to collect raw data continuously for one year, despite the extreme weather conditions and routine volcanic activities.
- **Remotely Configurable**: The OASIS system [1] aims to integrate complementary space and in situ elements and build an interactive, autonomous sensorWeb. The sensor network shall respond to external control from the NASA Earth Observation (EO-1) satellite and USGS science softwares, and adjust its behavior accordingly. The command and control needs to be delivered reliably in real time.
- **Fast Deployment**: Installing a conventional single-sensor monitoring station requires significant human involvement, which is not just costly, but also risky. In some situations involving volcanic unrest, it is strongly preferred that sensor stations are air-dropped and self-form a network to reduce the deployment cost and associated safety risks.

To meet these requirements, we comprehensively designed the system architectures, hardware, sensing and networking softwares, such as robust communication stacks, intelligent sensing algorithms, hybrid time synchronization protocols, and lightweight network management tools.

### 2.1 Hardware Design

With the direct involvement of experienced USGS engineers, the hardware design has considered various environment challenges in volcano hazard monitoring. We used a three-leg spider station (Fig. 2), which is about 4 feet tall (including the lifted antenna), and weighs about 80 kg. The
spider station was designed to support air-drop deployment and survive in the harsh volcano environment. We mounted a 6 dBi omnidirectional antenna on a steel pipe. AIR-ALKALINE batteries were used for one-year reliable energy supply. The battery’s weight also helps to stabilize the station in heavy gusts. A 900-MHz Freewave radio modem connected the sink station to the gateway over a 10 km radio link.

Inside the spider, the core hardware components are encapsulated in a small weatherproof box, with a dimension of $30 \times 20 \text{ cm}^2$ (Fig. 3). The box contains a wireless mote (iMote2), an acquisition board (MDA320CA), a GPS receiver (LEA-4T), and expansion connectors. We chose iMote2 instead of other sensor network platforms such as Mica2/ MicaZ/TelosB, because it has a good balance between low power consumption and rich resources: its PXA271 processor can be configured to work from 13 to 416 MHz; it has 256 KB SRAM, 32 MB SDRAM, and 32 MB flash. The adjustable computation capability and moderate memory spaces provide more flexibility for software development and allow adaptive and advanced in-network processing techniques in the future. In addition, it has rich I/O interfaces to support flexible extension, including two SPI, three UART, and multiple GPIO interfaces.

We configured iMote2’s PXA271 processor to operate in a low-voltage (0.85 V) and low-frequency (13 MHz) mode in normal operations. A low-power U-Blox LEA-4T L1 GPS receiver is connected to the iMote2 through UART interface for raw data capturing, and through GPIO 93 for pulse-per-second (PPS) signal capturing. The GPS is used to measure the ground deformations and timing for TDMA and data timestamp. The accuracy of its time pulse is up to 50 ns, and an accuracy of 15 ns is achievable by using the quantization error information to compensate the granularity of the time pulse. The GPS is also configured to provide UBX-RXM-RAW data for precise ground deformation measurement.

The seismic, infrasonic, and lightning sensors are connected to the electronics box through weatherproof connectors and cables. The seismic sensor, Silicon Designs Model 1221J-002, is a low-cost MEMS accelerometer for use in zero to medium frequency instrumentation applications that require extremely low noise. The infrasonic sensor, All Sensor Model 1 INCH-D-MV, is a low-range differential pressure sensor to record infrasound, low-frequency (<20 Hz) acoustic waves generated during explosive events. One of differential ports was open to ambient atmospheric pressure, and the other port had a low-pass pressure filter consisting of a rigid chamber and a 100 micrometer entry port. The lightning sensor is an RF pulse detector capable of detecting lightning strikes 10 km away. The lightning sensor was designed for this project by USGS. It consists of an antenna, MOSFET amplifier, and pulse conditioner. Heavy ash emissions are typically accompanied by lightning, and hence, it is a useful parameter for monitoring volcanic activity.

### 2.2 Software Stack Overview

Our sensor node runs TinyOS [6], the de facto operating system for wireless sensor networks. We choose TinyOS mainly because the TinyOS community has developed abundant client software tools and reusable node software components, such as SerialForwarder Java tool, routing, and MAC and time synchronization protocols. This reduces our development overhead and enables us to compare various design choices. When we started the project, TinyOS-2.x was not mature enough and does not support iMote2, and hence, we chose TinyOS-1.x as the software platform. Fig. 4 illustrates our node software architecture. In the application layer, the sensing module collects various sensing and utility data with synchronized timestamping. The network management module reports network health status and processes command and control from remote users. The situation awareness module detects events, prioritizes data, and implements priority-aware data delivery protocols. The transport layer provides reliable delivery for critical data such as seismic event data, and best-effort delivery for other data. The network layer
error compensation including clock skew estimation. The
utilizes MAC layer timestamping and comprehensive
[22] is a multihop time synchronization protocol which
may be damaged in the harsh volcano environment. FTSP
processing. While GPS provides high timing accuracy, it
timing accuracy ground deformation measurement through post-
sampling: all sensors in the network may perform periodical
2.3 Smart Sensing and Situation Awareness
USGS scientists require our network to support high-
fidelity synchronized sampling and be remotely configurable.
If the network has to drop packets, they prefer to drop those packets without event information. Naturally, the
resource usages shall be driven by the environment and network situations. Fig. 5 illustrates the framework of our
smart sensing component [24].

2.3.1 UTC Timestamping and Synchronized Sampling
One requirement from our earth science collaborators is that all stations perform synchronized sampling and timestamp recorded signals with precise UTC time, with error no more than 1 millisecond. Precise timing is important for utilizing the temporal and spatial correlation of volcanic signals. The high-resolution seismic analysis depends on the accuracy of seismic P-wave arrival time differences in the network. Note that synchronized sampling is implemented across the entire network. It means that all sensors in the network sample the environment parameters at the same time point, not just at the same interval.

The GPS is primarily used to measure ground deformation. Its raw data will be delivered to gateway for centimeter accuracy ground deformation measurement through post-processing of USGS software. We leveraged its existence to enhance time synchronization accuracy. Time synchronization could be realized either through network time synchronization protocols [13], [22], or through GPS signal processing. While GPS provides high timing accuracy, it may be damaged in the harsh volcano environment. FTSP [22] is a multihop time synchronization protocol which utilizes MAC layer timestamping and comprehensive error compensation including clock skew estimation. The
timing errors of less than 67 microseconds were reported for an 11-hop network of Mica2 nodes in [22]. However, previous research [33] has found that FTSP was not stable in its field deployment. Hence, we developed a hybrid time synchronization approach, Z-SYNC, that combines the merits of both GPS and FTSP: each node is synchronized to the GPS receiver by default; if the GPS signal disappears, then the node will switch to FTSP mode. Three situations can drive a node to switch from GPS mode to FTSP mode: 1) losing PPS signal for more than 10 seconds; 2) losing GPS data for more than 20 seconds; and 3) receiving two consecutive invalid GPS data. Later, if valid PPS signal and GPS data are received again, the node switches the synchronization back to the GPS mode immediately. In the original design of FTSP [22], the node with the smallest ID in a multihop network is selected as the FTSP root. All other nodes synchronize to the FTSP root. However, the FTSP root might lose GPS signal while some other nodes do not. To solve the problem, we extended the FTSP to support dynamic FTSP root election, as illustrated in Fig. 6. If there are GPS nodes in the network, then a non-GPS node synchronizes to the closest one with FTSP protocol. If the whole network loses GPS signals, the node with the smallest ID will be elected as the FTSP root.

To support synchronized sampling, we designed a Real-
Time Clock (RTC) module, which maintains a millisecond resolution timer with the support of iMote2’s microsecond resolution clock. RTC synchronizes its clock value $t$ to the UTC time and fires based on the clock value (instead of interval). In other words, the timer fires when the clock value $t$ satisfies $\% T = 0$, where $T$ is the sampling interval. For example, if $T = 10$ ms and the timer starts at 20 : 00 : 00 : 422, then the next fire point is 20 : 00 : 00 : 430, not 20 : 00 : 00 : 432. This enables strict synchronized sampling: all sensors in the network may perform periodical sampling at the same time point. Synchronized sampling is important for volcano signal spatial-temporal analysis and locating earthquake locations. Notice that, it is different from time synchronization. Assume the sampling rate is 10 Hz, without synchronized sampling timer design, as indicated above. Then, the time difference of samples from two synchronized nodes can be up to 100 ms, because each node may start a system timer at a different time point.

Evaluation on time synchronization accuracy. We evaluated time synchronization errors caused by clock drifts in sensor nodes. It is difficult to evaluate the timing accuracy in the volcano, and hence, we conducted outdoor evaluations on Z-SYNC before field deployment. We injected irregular sine wave with varying amplitudes and frequencies into sensor

![Fig. 5. Smart sensing component framework.](image)

![Fig. 6. FTSP root election with Z-SYNC: (Left) Node 3 is the TimeSync root; (Right) Node 1 is the TimeSync root.](image)
nodes through a D/A converter. The phase differences of the sine wave between a reference node and other node were measured as the time synchronization error. The evaluation was conducted on a nine-node network over a period of 8 hours in different modes (GPS, FTSP, and Hybrid). Under the hybrid mode, three nodes were connected with GPS while the other six were in FTSP mode. In the FTSP mode, only one node was equipped with GPS to provide UTC global time. The results are shown in Fig. 7. In the GPS mode, 1 millisecond time synchronization errors occur five times while 4 millisecond errors occur twice. In the hybrid mode, the number of 1 millisecond errors increases to 12 and the error of 9 milliseconds occurs once. The number of 1 millisecond errors increases in FTSP mode. However, the result is still acceptable since it only occurred with 24 out of thousands of data packets sent during the 8 hour period. Table 1 shows the time synchronization errors in accordance with the hop count over the network. The single 9 millisecond error under the hybrid mode as well as the 10 millisecond errors in FTSP mode only occurred at three-hop nodes. From the results, we can see that Z-SYNC can meet the science requirements of time synchronization errors ($t_{error} \leq 1$ millisecond) in most cases.

### 2.3.2 Configurable Sensing

Considering the longevity and remoteness of environment monitoring, remote reconfigurability is highly desired in a system deployment. With this feature, users can download the same program to different nodes, then adjust sampling rate, add or delete sensor channels, and configure data processing tasks on any subset of nodes. It naturally supports both self-adaptive configurations and external configurations.

<table>
<thead>
<tr>
<th>Hop Count</th>
<th>GPS</th>
<th>FTSP</th>
<th>Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
</tbody>
</table>

**Fig. 7. Time synchronization error.**

![Figure 7. Time synchronization error.](image)

**TABLE 1**

<table>
<thead>
<tr>
<th>Hop Count</th>
<th>GPS Error (mc)</th>
<th>Hybrid Error (mc)</th>
<th>FTSP Error (mc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
</tbody>
</table>

**Fig. 8. Configurable processing tasks.**

In a smart sensor network, the sampling rate and other sensing parameters shall be adjustable based on environmental conditions and mission needs. For example, the system may use, as a default, a 50 Hz sampling rate for normal data collection. If there is an event detected, it may then increase the sampling rate to get higher fidelity data in a short period. The sensing module in Fig. 5 performs synchronized sampling operation and maintains sensing parameters, such as sampling rate, channel (e.g., ADC port number), data resolution (e.g., 16-bit or 24-bit), sensor status, and reference voltage gain. Currently, our node has four sensor channels: the seismic and infrasonic sensors sample at 100 Hz with 16-bit resolution; the lightning sensor samples at 10 Hz rate with 16-bit resolution; the GPS receiver produces about 200 bytes raw data every 10 seconds. All of these parameters could be tuned according to environmental and resource situations to conserve energy or increase fidelity. This configuration meets the current requirement. In the future, if we want to introduce a new sensor into the system, we only need to attach the sensor to the station and configure a sensing channel for it without the need of reprogramming. If a sensor is broken, its channel can be closed to save energy and bandwidth.

Besides the configurable parameters, it is also useful to configure the data processing tasks for different data types. To support this, each processing algorithm is indexed in a task queue through function pointers. For instance, as illustrated in Fig. 8, assuming six different tasks have been programmed into the node and the default data processing task list is (1, 2, 3). The user may configure the task list (3, 4, 6) to process some type of data. With the future support of the incremental reprogramming over the air, a new task function’s binary code could be sent to the node and indexed in the task queue for configurable sensing. With this mechanism, we can choose to run different event detection and compression algorithms on different types of data on the fly, without reprogramming the nodes.

As different sensors generate different amounts of data at different times, and different hardware platforms have different amounts of storage space, we designed a sensing data management module to manage sensed data into different storage media based on platform limitations and
network situations. If there is enough free RAM space, sensing data will be saved into RAM space instead of flash. Otherwise, the unused data left in RAM will be moved to flash for later retrieval. In other words, the main function of our data management module is to manage sensed data into an appropriate place for processing and delivery, according to the data priorities and storage availability. The data management module hides the implementation details of how a free buffer is allocated and where the sensing data are saved. In our implementation, a list of sensor blocks are allocated initially. Each of the sensor blocks has the information of sampling start time, sampling rate, task execution code, status, and a raw buffer to save sensing data.

To support reliable remote configurations, we have designed a reliable data dissemination protocol and a remote procedure call mechanism, which will be described in Sections 2.5 and 2.7, respectively.

### 2.3.3 Situation Awareness

Real-time high-fidelity data collection is constrained by limited network bandwidth. If the network has to drop packets, then it should first drop the packets without critical event information. Hence, we designed a situation awareness middleware to detect the environmental conditions. The data during the event period will be assigned higher priority and have more chance to reach the sink through the QoS control in the underlying communication stack.

With the recommendation from earth scientists, we used the short-term average over long-term average (STA/LTA) algorithm [23] on seismic data to locate the earthquake events. To understand the STA/LTA algorithm, we need to first introduce the concept of Real-time Seismic Amplitude Measurement (RSAM), which is widely used by seismologists. It is calculated on raw seismic data by seismologists. It is calculated on raw seismic data.

$$\text{RSAM}_i = \frac{1}{C_0} \sum_{t=m+1}^{t=m+n} x_i$$

is the average seismic sample value in the $i$th and $(i-1)$th second, respectively, then

$$x_i = \frac{\sum_{k=m}^{t=m+n} s_k - s_{i-1}}{m}$$

where $x_i$ is $i$th-second RSAM and $n$ is the STA or LTA time window size. LTA gives the long-term background signal level while the STA responds to short-term signal variation. In our implementation, the STA window is 8 seconds; the LTA window is 30 seconds. The ratio of STA over LTA is constantly monitored. Once the ratio exceeds the trigger threshold, the start of an event is declared, and the LTA value is frozen so that the reference level is not affected by the event signal. The end of the event is declared when the STA/LTA ratio is below the detrigger threshold. Fig. 9 illustrates the relationship between the seismic waveform and the STA/LTA ratio. It shows that the STA/LTA algorithm performs seismic event detection with default STA/LTA ratio threshold 2. Although a ratio of 2 is recommended by seismologists and is a good threshold in our lab experiment, it is not always good in real volcano environments as discussed in Section 5.

For lightning data, which has a “strike value” when lightning takes place, a threshold-based trigger approach is applied to detect the event.

### 2.4 Agile Data Collection Routing

Our data collection routing protocol, MultihopOasis, is developed based on MultihopLQI in TinyOS-1.x [6]. MultihopLQI is a many-to-one distance vector routing protocol for data collection sensor networks. We improved MultihopLQI by considering various practical system issues. Table 2 shows some important parameters that we used in routing management. The improvements include:

#### 2.4.1 Link Quality Estimation

MultihopLQI is a distance vector routing protocol, using link quality indication (LQI) as the routing metric. In CC2420 radio, the LQI measurement is a characterization of the strength and/or quality of a received packet. However, some previous works have reported that LQI fluctuates over time. Our experiments showed that, instead of using each beacon’s LQI, the average LQI is better to reflect packet delivery ratio. We apply the Exponentially Weighted Moving Average (EWMA) on LQI values in each time frame (window) and calculate, $lqi(t) = (1 - \alpha) \times lqi(t) + \alpha \times lqi(t - 1)$, where $\alpha$ is 0.25.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial TTL (Time To Live)</td>
<td>31</td>
</tr>
<tr>
<td>$t_{WD}$ (WatchDog check period)</td>
<td>10 min</td>
</tr>
<tr>
<td>$t_B$ (Beacon Interval)</td>
<td>10 sec</td>
</tr>
<tr>
<td>$t_E$ (The time for a valid Path expire)</td>
<td>6 $t_B$</td>
</tr>
<tr>
<td>$t_F$ (The time window to monitor broken link)</td>
<td>15 sec</td>
</tr>
<tr>
<td>$\alpha$ (EWMA factor for average LQI)</td>
<td>0.25</td>
</tr>
<tr>
<td>Queue size reserved for forwarding packets</td>
<td>66.7%</td>
</tr>
</tbody>
</table>
2.4.2 Count-to-Infinity Problem
MultihopLQI, as a distance vector routing protocol, intrinsically has the count-to-infinity problem. In our design, each packet has a TTL field (5 bits) which is initialized to 31 at the source node. TTL decreases 1 each time the packet traverses an intermediate node. When TTL reaches 0, the packet is discarded. Moreover, each sensor node maintains a history message queue. Each entry records a triple (sourceID, seqno, TTL) of a forwarded packet in history. When forwarding a packet, a node looks up its history queue. If there exists an entry that has the same source address and sequence number, but a different TTL, then a loop is detected; otherwise, a duplicated packet is received. TTL is necessary to differentiate route loops from packet duplications. If a loop is detected, the routing protocol will invalidate the path and rediscover a new path immediately.

2.4.3 Agile Network Self-Forming and Self-Healing
When a node detects a loop, or it does not receive route beacon packets from its parent for more than six beacon periods, or all packet transmissions in the last 15 seconds fail, it will invalidate its link to its current parent. In a standard distance vector routing protocol, each node periodically broadcasts beacon packets to update route information. The node with an invalid parent must wait for new beacons to re-discover a new path to the sink. During this waiting period, the node has to wait and may discard some packets due to buffer overflow. The time delay is proportional to the length of the beacon period. A short beacon period can mitigate the problem, but it has the disadvantage of introducing more beaconing overhead, which is undesirable. To make the network agile to link status changes, we make a few changes to enable fast self-healing: 1) When a node’s parent becomes invalid, it invalidates the parent immediately and issues a beacon packet indicating infinite path cost. Additionally, if a node receives a beacon from its current parent containing bad link news, it also invalidates its current parent and sends out a beacon message to notify other nodes immediately. 2) If a sensor node with an invalid parent receives a beacon and discovers a new parent, it will broadcast a beacon of the new route information immediately. In that way, the network will take a short time to self-form or self-heal. This is especially effective during the network startup.

As mentioned above, when a node detects a loop, beacon timeout, or maximum successive transmission failures, it will invalidate its current parent and wait until a new path is discovered. While a reactive approach for fast network healing is proposed above, we also utilize a proactive approach, multihop backup, to further reduce the recovery delay. If a node’s parent becomes invalid, it searches for an alternative in its route table first. In our design, a neighbor management module caches all valid paths toward the sink proactively. Once a received beacon indicates a valid path toward the sink, the neighbor management module deposits the route information to the route table. In a dense network, a node may have more neighbor nodes than the route table size. Our replacement strategy not only considers path cost, but also considers the liveliness. Each routing table entry has a liveliness field. If an entry is updated by a beacon, its liveliness is reset to a maximum value; if an update timer fires, the liveliness value will decrease by 1; if its liveliness is decreased to 0, then that entry becomes invalid. When the routing component looks up an alternative path to the sink, the path with lowest cost will be chosen without waiting beacons to rediscover a path.

2.4.4 Asymmetric Link Problems
Asymmetric links commonly exist in a wireless network, due to many factors, such as the irregularity of radio transmission range and antenna orientations. As illustrated in Fig. 10, node c can hear from node a, but node a cannot hear from node c. Asymmetric links can cause complete node failure: although a child node can receive beacon packets from its parent, its own packets may never reach its parent. With Automatic Retransmission Request (ARQ) mechanism in the link layer, the child node will retransmit the unacknowledged packet and degrade the network throughput. MultihopOasis addresses the asymmetric link problem by monitoring link status. If its link loss ratio during a time window exceeds the threshold, the child node switches to an alternative parent as described in previous paragraph, and puts the old parent into blacklist to avoid switch oscillation. As illustrated in Fig. 10, with MultihopLQI node c chooses node a as parent, but its packets never get an acknowledgment; with MultihopOasis, node c will switch parent to node b after maximum successive transmission failures.

2.5 Reliable Command and Control
As introduced earlier, the presented sensor network is part of an autonomous space in situ sensorWeb, hence interaction with external components is an important requirement. For example, a science software may generate control commands to adjust sensing parameters in real-time when some monitored area becomes more active. Those feedback commands require 100 percent reliable delivery in the correct order. Moreover, the dissemination process should terminate in an acceptable time window. To our best knowledge, the existing dissemination protocols either do not guarantee 100 percent reliability or do not terminate [17], [21] in short time (e.g., based on gossip). So, we designed the Cascades protocol [25] for reliable and fast data dissemination. The Cascades protocol ensures 100 percent reliability by utilizing the parent-children relationship in a data collection tree: each node monitors whether its children have received the broadcast messages through snooping. Each node rebroadcasts periodically until successful reception is confirmed by all children. The broadcast flow does not depend on the data collection tree structure. A snooped new message from neighbor nodes will be accepted and rebroadcasted. Therefore, it is possible that a node first hears new data from its children before it hears it.
in [25].

More details can be found in [25].

node’s sequence loses synchronization, it can resynchronize with its parent. In other words, it is a fast opportunistic data dissemination protocol. For example, in Fig. 11, assume node a’s DATA does not reach f, but node b’s ACK and node g’s DATA have reached f, then f sends an ACK to a without further rebroadcast. It is not difficult to imagine that: the opportunistic design makes the protocol not just faster, but also more robust as a node does not only rely on its parent node to get disseminated data. In addition, Cascades performs a reactive fetch mechanism [32] if there is gap in packet sequence, which denotes missing packets. If the link reliability between a parent and child is temporarily not good, it still has a good chance of getting the data from some other neighbor nodes.

In an integrated system, a key challenge of command and control is the consistency of broadcast message sequence number. There might be multiple control clients of a single-sensor network. In addition, there might be multiple sink nodes in the sensor network, and a sensor node could be rebooted. All those facts could cause the sequence number in a network to be inconsistent, resulting in confusion of nodes to differentiate between old and new commands. We solved this problem by maintaining the sequence number in SerialForwarder, a TinyOS tool customized to support packets forwarding between Internet and multiple sink nodes. In addition, Cascades resets its sequence number if it is idle for 10 minutes. With this mechanism in place, even if a node’s sequence loses synchronization, it can resynchronize with the SerialForwarder after 10 minutes. More details can be found in [25].

2.6 Link Layer Optimization

TinyOS is an event-driven system and does not support multithreading. In the recommended practice of TinyOS-1.x, an uplayer component cannot send another message to a low-layer component until the sendDone event of the previous message is received, which could cause a big gap between two consecutive sends. A similar situation happens during receiving. In addition, our network has two independent routing protocols for data collection and command dissemination, respectively, above the link layer. They could compete for the resources of the link layer. One may occupy the link layer, while the other cannot access the link layer. Recently, this issue has also been studied by Choi et al. [37]. We addressed these system challenges by designing sending/receiving queues at each component to support pipelined sending/receiving and enable reserved queue spaces for different upper layer components. Pipelined sending/receiving is also important to TDMA MAC protocol. In a TDMA protocol, once a node is allocated a time slot, the node should use the slot efficiently, especially when high throughput is demanded. However, in the simplex approach of default in the case of a TinyOS-1.x link layer (e.g., GenericCom), when a packet is sent by the MAC layer, the MAC module will signal up a sendDone message, usually via posting a task. And only after the original sender processes the sendDone event, will it make another send request to MAC component. In this way, the precious channel time is wasted. In TreeMAC [28] design, we overcome the problems by adding queues at each communication component. In this way, when the MAC finishes sending out one packet, it will immediately fetch the next packet in the queue, instead of waiting for the upper layers to send down a new packet.

In Section 2.3.3, we assigned higher priority to event data in the hope that event data can reach the gateway with higher probability. This requires prioritized data delivery in the link layer as well. We need to ensure that higher priority packets have a higher probability to reach the next hop. For low-priority packets, one retry is enough; but for those higher priority packets, more retries are desired. In our algorithm design, we have implemented a mapping which contains a static table for the mapping from priority to the number of retransmissions. The number of retransmissions is predefined based on empirical results obtained through simulation. When transmission failure occurs, a node looks up the table. If current retry count exceeds the limit, it gives up on that packet; otherwise, it retransmits the packet.

2.7 Network Management and Robustness

Network management is more challenging in sensor networks than in traditional networks, due to the limited computation, storage and communication resources, and diverse application requirements. The system robustness is also particularly important, because those sensors typically work in hostile environments, and it has to rely on itself for recovery. In this section, we present a lightweighted sensor network management mechanism [36], and a comprehensive software watchdog to increase network robustness.

2.7.1 Network Control and Status Report

Visibility into network failures is an important network management requirement, as knowing the cause of network failures can help to correct bugs and improve the system. We developed a transparent and lightweighted Remote Procedure Call (RPC) mechanism based on [34], which has little overhead on sensor nodes. It allows a PC to access the
exported functions and any global variables of a statically compiled program on sensor nodes at runtime. At pre-compilation time, an RPC server stub is automatically generated and added to a nesC application. All information necessary for an RPC client to use RPC is parsed from the application source code and exported to an XML file by a Perl program. At runtime, the RPC client imports all information from the XML file. Fig. 12 illustrates the process.

The RPC mechanism gives users great flexibility to read/write system variables and run any exported functions. Operations such as set/get sampling rate, beacon interval, power level, radio channel, and event report threshold are provided to remote clients. For example, we found that occasionally some nodes forwarded packets for others, but they stopped sending their own packets. It could be because the sampling timer stopped triggering, or the data management in sensing component got corrupted. The problem only happened occasionally and it was hard to repeat. So, traditional debugging approaches fail to find out the cause. However, with the visibility and flexibility of RPC, we were able to narrow down the scope of the suspicious code and identify the problem.

We also enabled the sensor network to report important events or node status periodically for network health diagnosis, such as battery voltage, buffer status, and neighbor RSSI and LQI. The control client is then able to monitor network connectivity and health. This information is particularly useful during field deployment and evaluations. We realize this through an EventReport component, which provides users the flexibility to monitor different levels of system status. It uses a mechanism similar to dbg() in software debugging. Users can customize the emergency level of events. Events with an emergency level below the report threshold will be filtered out to reduce the communication cost. The report threshold is configurable during runtime using the RPC command. It allows users to control the amount of status report information based on network traffic situations. By default, no filters are set for reported events so that all of the events are sent to the client.

### 2.7.2 System Design for Robustness

In the reality of software engineering, it is almost impossible to find and fix all bugs. A distributed wireless sensor network is especially hard to debug, although the community has developed various debugging tools. However, volcanologists expect our system to be capable of running continuously on Mount St. Helens for at least 1 year. It is, therefore, crucial to have an exception handling mechanism that allows it to recover automatically from software and hardware failures. We exploit the benefits of watchdog. The iMote2’s hardware watchdog can restart the node if there is any dead loop, illegal memory access, stack overflow, or division by zero. We further developed a software watchdog to enable self-recovery from unexpected logic errors. Those errors do not necessarily cause sensor nodes to die, but make them unstable. For example, if the communication protocol or the message queue corrupts due to whatever reasons, a node may no longer accept new messages. In our system, if no messages are sent or received for 10 minutes, then the node will be rebooted. The control center can also send an RPC command to reboot a node if necessary.

In addition, we also made some improvements in TinyOS-1.x. For example, we found and solved some problems of the CC2420 radio driver in TinyOS-1.x: 1) Under intense network contention, the radio stack may receive a packet with valid CRC, but with the length that is smaller or bigger than the real packet length, due to some problems in radio software or hardware. Those corrupted packets not only waste network bandwidth, but also cause the control client to die. We suppressed those corrupted packets by performing a sanity check on the packet length field. 2) Sometime, when sendDone fails, the packet header (e.g., AM type) may also be corrupted. This causes the message pointer to return to a wrong upper layer component in TinyOS. We avoided this through backing up the packet header before sending. When a sendDone event is received, the packet header is restored. 3) Sometimes, the sendDone event for the same packet may be signaled twice, perhaps due to some race conditions. Our queue management module is resilient to such unexpected situations and ignores the duplicated events.

### 2.8 Data Management

The OASIS data products have been feed into existing USGS data storage and analysis tools. Fig. 13 illustrates the software architecture of network management and data manipulation tools. ExportOasis was developed to import
the data stream from seismic, infrasonic, lightning sensors, and GPS, as well as RSAM, battery voltage, and LQI data, into an MySQL database with UTC timestamps of millisecond resolution. Wave data (seismic and infrasonic) is reformatted into standard form for storage by ImportEW before being logged into the MySQL database. During the campus outdoor test, we also learned that we should have paid more attention to the connection between components, including software interface and hardware interface. Besides the example of the L-connector, there are several problems caused by interconnection in the system. For instance, the sensor board is connected to the iMote2 and the amplifier. Originally, we planned to use a 4-foot tall 12 dBi omnidirectional antenna to extend the transmission range of the CC2420 radio. However, the transmission range was only about 25 meters, which is even less than without it. The reason will be explained later. Then, we began to seek an omni-directional antenna to extend the transmission range of the CC2420 radio. However, the transmission range was almost the same as that of the field deployment. The reason will be explained later. Then, we began to seek an omnidirectional antenna to extend the transmission range of the CC2420 radio. However, the transmission range was almost the same as that of the field deployment.

During the campus outdoor test, we also learned that we should have paid more attention to the connection between components, including software interface and hardware interface. Besides the example of the L-connector, there are several problems caused by interconnection in the system. For instance, the sensor board is connected to the iMote2 through an interconnection board. But during a movement, we made various attempts, such as replacing a new iMote2 or replacing a cable. Finally, we realized that it might be because the signal strength was high, but signal quality (e.g., signal to noise ratio) was low, and hence we developed a packet-level LQI measurement program called TOSBaseLQI (based on TOSBase in TinyOS). Indeed, we found that the packet LQI was below 70 (typically, above 90 is good, 110 is the maximum), though RSSI was about −30 dBm (which is far above the CC2420 receiver sensitivity threshold). We learned a lesson from this: signal strength does not necessarily reflect signal quality. It is actually not surprising because link quality is correlated with signal to noise ratio, not the signal strength. After we knew the true reason, we began to reduce the potential noise sources between iMote2 and the amplifier. Originally, we found that an L-connector between the iMote2 and the amplifier was the problem. The L-connector did not attenuate the signal much, but it added noise which was also amplified. Small changes made big differences. After the L-connector was removed, the LQI of received packets increased to 90-106, even at a distance of more than 400 meters. Without the quantitative measurements of link quality using TOSBaseLQI, we would think it was normal, since low link reliability is the common assumption in the wireless networking community. In fact, it may also be caused by a bad RF design, like the example given above. This experience also taught us that LQI was a more reliable and accurate link metric, although some research [29] appreciated RSSI. After we learned this lesson, we began to record the LQI of each link to a database.

In the lab test, we set up a GPS antenna outside of the building and linked its signals into the room with a cable, and everything worked well. But sensor nodes became unstable on time synchronization after we moved to the outdoor campus. It took us several days to find the reason. In the lab test, the nodes were close to the building wall, and hence received at most eight satellite signals. When the nodes were placed in the open field, they could receive more than 13 satellite signals at the same time, which caused the GPS data buffer to overflow and triggered the watchdog to restart nodes frequently, thus becoming unable to synchronize to GPS. We fixed it by increasing the data buffer size.

During the campus outdoor test, we also learned that we should have paid more attention to the connection between components, including software interface and hardware interface. Besides the example of the L-connector, there are several problems caused by interconnection in the system. For instance, the sensor board is connected to the iMote2 through an interconnection board. But during a movement,
it will occasionally become loose. When the sensor board is loose, the data acquired is a straight line of 0xFFFF value.

4 FIELD DEPLOYMENT

4.1 Deployment Experience and System Configuration

On 15 October 2008, we successfully air-dropped five stations into the crater of Mount St. Helens and streamed data to the Internet. The deployment map is illustrated in Fig. 14. First, the sink node with ID 10 was lifted up and dropped into the crater and it immediately showed up on our monitor. Then, when the second node with ID 14 was lifted up and toward the crater, we were excited to see that node 14 had linked the remaining ground stations at the preparation site to node 10 inside the crater. The distance between the preparation site and the crater is about 10 km. The radio transmission range was remarkably long and beyond our expectations. The same situation repeated during the rest of the deployment. Our optimized data collection routing protocol worked remarkably well and formed the network immediately. In our routing protocol, once a node discovers its parent, it broadcasts a route beacon immediately. This helps to build the network immediately in that short transition period. We have made several other improvements on the routing protocols as described in previous sections. After all nodes are deployed, we found that node 16’s seismic data plotted a straight line. From our experiences in the campus test, we recognized that this meant that the seismic sensor was broken. We believe that the sandbox containing the seismic sensor was spinning too hard during the helicopter lift, causing the connection to break. We returned to the crater and fixed the problem. This was the only problem we encountered during the air-drop deployment.

Fig. 15 illustrates the system configurations. The rugged crater has a diameter of around 1 mile. During the deployment, the air-dropped nodes immediately formed a data collection tree, and delivered real-time sensor data to the sink node through the multihop network. The sink node then relayed the data to gateway (e.g., MOXA device server) at JRO through a Freewave radio modem. Then, the gateway relayed the data stream to our sensorWeb research lab through a microwave link of 90 km. In the lab, a Serial-Forwarder Java tool forwards data between the sensor network and the Internet. Multiple control clients may connect to it, access the sensor data stream, and control the network in real-time. In the OASIS project, a real-time network tool Monitor was also developed for system debugging. It can display the network topology and the status of each node, and visualize the real-time sensor data in an oscilloscope view. Moreover, it is also a network packet analyzer, with which we can capture and browse packet data for analysis and trouble shooting. It also allows users to retask the remote network and tune operational parameters such as sampling rate and data priority. The Monitor tool greatly benefits our system development and field deployment.

4.2 System Evaluations and Findings

In this section, we analyze the deployed system performance and compare it with existing USGS volcano monitoring stations in Mount St. Helens.

4.2.1 Network Evaluations and Findings

We computed the loss ratio from the data logged in the database, and found the end-to-end overall data delivery ratio as high as 91.7 percent. Notice that this loss ratio is not the sensor network packet delivery ratio, but the end-to-end overall system delivery ratio. If we do not count the failures of data servers, Java tools, Internet, and hardware, the sensor network itself achieves a remarkable 99 percent packet delivery ratio over the 1.5-month evaluation period. Fig. 16 illustrates the hourly loss ratio for all the nodes during the first month of deployment.

In Fig. 16, we can see that node 15 was down for 1 week. About 6 hours after the deployment, node 15 disappeared. A USGS engineer went to Mount St. Helens for another mission 5 days later, and found the station was tipped over.
He landed, placed the station in an upright position, and then weighted it with big rocks in the hope that it wouldn’t be tipped over again. However, only 2 hours after he left, node 15 disappeared again. We were wondering if the node was tipped over again, but no more helicopters were traveling to Mount St Helens. We had to find another solution to investigate what happened there. After having explored many possibilities, we decided that three persons would do a 6-hour hike to the crater to see what was going on. They discovered that node 15 was still on its legs but the voltage regulator had been damaged. After the problem was fixed, node 15 has stayed alive. The problems with node 15 made us realize the weather conditions inside the crater of Mount St Helens are very tough. Nodes experienced high temperature variations, heavy gusts, rains, and even snows during the second half month of the deployment. Very strong gusts were recorded by our infrasonic sensors. The infrasonic sensors are essentially pressure sensors, and hence, are able to capture those heavy gusts (Fig. 18).

In Fig. 16, when the loss ratio of all the nodes is at 100 percent at the same time, it means there was an error on the database server or Internet; when only one node has a high loss ratio, it means that this node had some troubles. Node 16 around 11 November 2008 had a relatively low packet delivery ratio. From our data log, it had been restarted frequently. We found that the battery voltage level was close to the lowest permissible limit, which perhaps explains why. Before deployment, USGS engineers intended to put the used batteries to all nodes to test our network self-healing ability and stability. Node 16 is one of them and probably ran close to the energy depletion status.

Fig. 19 shows the uptime of each sensor node and the database server. The uptime was calculated hour by hour using the assumption that if a node is sending data during that hour (even with some loss), then it is up for that hour. The overall network uptime is about 93.8 percent. The gaps in a solid line denote either the database server was down or there were no data from the station. From Fig. 19, we can see that all stations achieved a high percentage uptime except node 15 due to the troubles described earlier. The other four nodes’ uptime is about 100 percent.
Overall, we were excited to see that, during the whole evaluation period, no nodes died and the system successfully recovered from numerous challenges.

### 4.2.2 Data Evaluations and Findings

One important aspect to evaluate a data collection system is the data quality. To assess the quality of our data, we compare it with other data sources from the volcano. USGS owns four types of measurement stations on Mount St Helens:

1. **Dual-frequency GPS** with store-and-forward telemetry when polled.
2. **Short-period seismic** stations with geophones and analog telemetry.
3. **Broadband seismic** stations with digital telemetry (24-bit ADC, the gold standard in seismic monitoring).
4. **Microphones** for explosion detection added to the short-period seismic stations.

A quick comparison between the capabilities of existing USGS stations and our Oasis station can be found in Table 3. To differentiate our station from the USGS station, we called our sensor node Oasis station in the following, named after our project name OASIS [1].

**Scientific value of the data.** USGS volcanologists were satisfied overall with the data quality (e.g., signal to noise ratio). For example, a magnitude 1 earthquake on the volcano was detected on 11 April 2008. In Fig. 20, we can see the detected event data by our Oasis station and existing USGS stations. The dynamic range, resolution, and signal-to-noise ratio of the Oasis seismic signal compared favorably with short-period and broadband seismic station in the crater. For the purpose of earthquake event detection and warning, we can see that our system was able to accurately detect earthquake events with a precise timestamp. The biggest advantage of our sensor is its ability to be set on the ground without regard to orientation.

Later, when we analyzed the data from the deployed network, we found that some seismic data samples lost their six Least Significant Bits (LSBs) and have distortions. We worked closely with domain scientists to figure out the reason. We tested the data acquisition board MDA320 with another microcontroller using max clock rates up to 2.4 MHz, and found the sensor data has very good signal-to-noise ratio. Eventually, we figured out that the 6.5-MHz clock rate of iMote2 driving our ADC driver was the cause of the data distortion. The ADC chip ADS8344 [8] can only work normally at 2.4-MHz clock mode. In other words, the external clock cycle should be no less than 400 ns (2.4 MHz) to correctly accomplish the conversion. The iMote2 SPI clock rate is 13 MHz. Originally, we used SPI clock divisor 2, so the on-chip clock is set to be 6.5 MHz. Thus, we changed the SPI Serial clock rate to 2.6 MHz by configuring the clock divisor as 5. This SPI clock rate was still slightly higher than the specification, but ADS8344E worked normally and the data distortion was eliminated.

**Cost of the data.** The cost includes the equipment and deployment cost. The gold-standard broadband seismic station costs more than $10 K, while the Oasis station costs about $2 K (which includes GPS, infrasonic and lightning sensors, besides a seismometer). Moreover, all old stations require human on-site intervention to become operational, while our five stations were dropped by helicopter. The turnaround time for each station was only 10-15 minutes; hence, deploying all five stations took only 1 hour. In the past, it took USGS several hours, even a day, to deploy a single monitoring station, as it requires scientists to enter the crater, dig a hole, bury geophones in the upright position, and orient the directional antenna to the gateway. The sensor networking technology allows the fast deployment of sensors into dangerous or nonhuman accessible spots. Also, the maintenance cost and the energy cost of our system are also significantly lower than all the old stations, which use telemetry for data acquisition and require higher energy consumption. Table 3 summarizes the major differences. The money cost counts the hardware and deployment labor cost.

The low deployment and maintenance cost is also contributed by the self-organizing and self-healing mesh networking technology. Smart mesh networking not only enables the flexible deployment in locations without line-of-sight, but also provides better fault tolerance and longer network lifetime. None of the old station technology has smart networking capability. The sink node is still a bottleneck of the system as it relays the data from the whole network to the gateway. In our future larger-scale...
deployment, we plan to use multiple sink nodes: if the active sink dies or its bandwidth is saturated, another node can also become a sink to forward data to the gateway.

Another advantage of a wireless sensor network, against the old monitoring technology, is the in-network processing capabilities. Instead of processing data in a central computer, the data processing algorithms, like seismic event detection, characterization, timing, and localization can be done in the network, and alarms can be triggered when too many seismic events are detected or when eruption is detected. Having such high-level information available in the network has numerous advantages. It allows the network to reduce its load and energy consumption by only sending event information and important data instead of continuous raw data. We are still in the preliminary explorative phase for such in-network detection and characterization algorithms. At this point, volcanologists are demanding continuous raw data that our system provides. Their current work involves designing event detection and classification algorithms using the data that our system provides.

5 Design and Deployment Lessons

We have learned enormous system design, deployment, and management lessons in the past several years. Due to space limit, we only enumerate several lessons as regards their various aspects:

Hardware Verification. When implementing this system, most of our effort was placed on software design and verification. We did not test the radio transmission range of our spider station until we began our campus test. As discussed in Section 3, the tough hardware problems exposed in the campus test were very time-consuming to investigate and solve, and forced us to postpone our deployment plan. This experience taught us that hardware shall also be verified as early as possible. Hardware connections shall also be carefully examined as it could cause unexpected errors and add significant overhead to software test.

Remote Configurability. The RPC [34] mechanism supports flexible system configurations after deployment and is very helpful. It allows a remote user to call exported RPC functions as well as read/change global variables (which are exported to an XML file at compilation time). In our system, to support remote management, we exported 23 RPC functions providing the flexibility to adjust sensing, routing, and radio parameters. However, after the deployment, we found they were still not enough for us. We found that the default LTA and STA time window parameter in our event detection algorithm is not good in real volcano environment, although it is good in lab environment.

When our STA/LTA algorithm identifies seismic events, it assigns the highest priority 7 to the event data, as shown in Fig. 21 (Up). The default STA/LTA ratio 2 works well in the lab test with seismic data injected through a D/A converter. However, postdeployment analysis shows that 2 is not always the optimal STA/LTA ratio. Due to the high sensitivity and low noise level of the seismometer, many small activities are also recognized as earthquake events. If we choose 3 as the STA/LTA threshold, those small volcanic activities will not be considered as events, as illustrated in Fig. 21 (Bottom). Using the appropriate parameter value can avoid false event triggering and result in better bandwidth allocation.

Fortunately, the RPC [34] supports reading or changing global variables as well. However, those parameters were hardcoded in the software and could not be changed. With that lesson, we changed all important parameters to global variables. This also exemplifies the importance of remote configurability. We can never plan perfectly. However, system design could always keep configurability and flexibility in mind. Hardcoding shall be absolutely avoided.

Tools Chain Challenges. To target 1-year continuous operation, we also faced enormous challenges in data management tool chain and network infrastructures. Issues such as the outage of routers, and exceptions of the data importer occasionally made our database server unable to log the data. For example, we met the Daylight saving time (DST) issue on a Windows XP PC. At 0:00 UTC time 26 October 2008, some mysterious event killed the ExportOasis tool. Fortunately, we have maintained a detailed data log and were able to identify the problem within several hours. First, we checked the SerialForwarder log file at 0:00 UTC time on 26 October. During that period, we saw that the ExportOasis disconnected from the SerialForwarder while the Monitor tool did not. Therefore, we can narrow down the problem in the machine running ExportOasis. Then, we found that the time is set back 1 hour at 1:00 UTC time 26 October. Notice that, from 1987 through 2006 [9], the start and end dates of DST were the first Sunday in April and the last Sunday in October (which would be 26 October in 2008). Starting in 2007, most of the US and Canada observe DST from the second Sunday in March to the first Sunday in November, almost two-thirds of the year. The 2007 US change was part of the Energy Policy Act of 2005. That Windows XP machine did not install the patches released by Microsoft and still used the DST before 2007. In our design, to reduce packet size, the timestamp of each sensor data packet only includes minute, second, and millisecond, not hour information. The ExportOasis must use the PC time to calculate the timestamp before inserting data into the database. In our design, we assumed that the hour will only round back at 23:xx:xx; the minute will round back at xx:59:xx. As a result, the problem occurs...
when the hour rounds back at 00:xx:xx. Obviously, such a system issue occurs only twice a year and is hard to repeat.

**Education and Project Management.** SVN, Wiki, and email list are useful tools to management source and information among the group. The learning curve of TinyOS is sharp. One main reason is that the TinyOS-1.x tutorial is not mature. Instead of explaining this or that rule in TinyOS, we found it was much more effective to tell students how nesC code maps to C code. We later told students that TinyOS code is preprocessed to generate a single C file called app.c, before being compiled to binary code. The rules or keywords in nesC are then very easy to understand, because students can look into app.c and see how it is converted from nesC code.

### 6 Related Works

The first-generation sensor network deployments typically have low duty cycle and low data rate application characteristics. In [11], a tiered sensing system was deployed on the UC James San Jacinto Mountains Reserve to collect dense environmental and ecological data about populations of rare species and their habitats within a mountain stream ecosystem and the surrounding conifer forests and meadows. In the tiered system architecture, data collected at numerous, inexpensive sensor nodes are aggregated by larger, more capable, and more expensive nodes. From 2002-2003, researchers used sensor network to monitor the habitat of the Leach’s Storm Petrel [30] at Great Duck Island. It is a representative application with low sampling and bandwidth demands. The results shed light on a number of design issues from selection of power sources to optimizations of routing decisions. On a smaller scale, a sensor network was deployed on a single redwood tree [31] using 33 nodes to cover roughly 50 meters. With this deployment, researchers were able to map the differences in the microclimate over a single tree. Those deployments have made valuable contributions in establishing sensor networks as a viable platform for habitat monitoring and developing essential components for the research community. Other habitat monitoring deployments include tracking Cane Toad populations [27], and the movements of zebras [18] and turtles [2].

Some recent deployments involved high data rate signals with high-frequency sampling, like our application requirements. In [20], a 64-node sensor network was designed to monitor the structural health of the long-span Golden Bridge. Ambient vibration of the structure is monitored and is used to determine the health status of the structure. Signal processing is used to increase the quality of the sample. Similar to ours, the system includes a protocol for reliable command dissemination, and improvements to software components for data pipelining, jitter control, and high-frequency sampling. Because the nodes are almost on a line in this deployment, the bidirectional antenna is used to provide longer connectivity and reliable communication link. BriMon [12] designs a sensor network system for long-term health monitoring of railway bridges and reporting when and where maintenance operations are needed. It uses a simple time-synchronization scheme and multiple channel communication. It uses an event detection mechanism to trigger data collection in response to an oncoming train. When volcanology becomes mature enough and can accurately identify events, the data delivery upon event detected may also be designed to save bandwidth and energy. At this point, USGS scientists highly demand real-time continuous raw data.

Recently, researchers also tried sensor network deployment in rugged terrains and under harsh environmental conditions, similar to our deployment. Researchers developed the FireWxNet [14], a multiterrier portable wireless system for monitoring weather conditions in rugged wildland fire environments. FireWxNet provided the firefighting community with the ability to measure and view fire and weather conditions over a wide range of locations and elevations within forest fires. The task of deploying the in situ network was particularly severe, given the rugged mountainous and forested terrain over which FireWxNet was spread. The network covered a unique topography which had not been studied before, ranging from sharp elevational differences to a fairly wide coverage area spanning about 160 square km. In [4], a seismic network comprises 50 broadband seismic stations along a 500 km line across Mexico from Acapulco to Tampico. Stations were placed roughly every 5 km and linked by 802.11 radios with some repeater stations. In this wide-area deployment, the network is vulnerable and disruptive, and hence, a Disruption Tolerant Shell (DTS) mechanism is used to handle network breaks and avoid data loss. In our application, we used the low-power 802.15.4 radio with amplifier, which is sufficient to cover the volcano crater with 1-mile diameter. In the future large-scale volcano deployment, the network disruptions may be a severe issue as well. Then, we may consider applying a similar mechanism in the transport layer. Each base station is connected to the Internet, allowing data transfer from there to data repositories as well as check stations remotely in real-time.

Perhaps the most relevant prior research was that done by Welsh and his colleagues [33], who deployed a sensor network on an active volcano in South America to monitor seismic activity related to volcanic eruptions. They did a science-centric evaluation of a 19-day sensor network deployment at the flank of Reventador, an active volcano in Ecuador. They used an event detection algorithm to trigger an interesting volcanic activity and initiate reliable data transfer to the base station. During the deployment, the network recorded 229 earthquakes, eruptions, and other seismoacoustic events. However, they found their event detection accuracy was only 1 percent, which justifies the requirement of real-time continuous raw data delivery by USGS. Their network reliability and uptime was relatively low: the mean node uptime was only 69 percent. This work reveals many hard lessons in volcano monitoring, which greatly benefit our design of OASIS system. Our work differs from them [33] in many aspects. OASIS system is the first air-dropped sensor network for long-term volcano monitoring, driven by USGS demands. The goal is to replace existing data logger or telemetry systems and reduce the deployment risk, time, and cost. Second, our system was designed to survive, unattended, for 1 year and collect raw data continuously. Such longevity and data yield bring significant challenges
because we must ensure that the system is robust even under harsh environmental conditions and resilient to software and hardware failures. Overcoming these challenges pervades our design. Third, our system was co-designed with USGS scientists to exacting standards—perhaps the most important of which was the need to do time-synchronized data acquisition across the sensor network and the need to be remotely configurable due to the longevity and remoteness of volcano monitoring.

7 CONCLUSIONS AND FUTURE PLANS

Designed for rapid deployment during volcanic unrest, our system mainly focused on achieving real-time high-fidelity, remote reconfigurability, and a high-degree of robustness. The high data yield of our deployed system proves its robustness. The presented system design and deployment experience proves that the low-cost sensor network system can work in extremely harsh environments. Our next plan is to integrate a localized TDMA MAC protocol [28] and lightweight compression algorithm [19], and deploy a larger scale sensor network with multiple sinks and multiple channels for real-time volcano monitoring in Summer/Fall 2009.2

ACKNOWLEDGMENTS

The OASIS project has received enormous support from many USGS and JPL scientists and engineers. To name a few, USGS participants include John Pallister, Dan Dzurisin, Seth Moran, and Mike Lisowski; and JPL participants include Sharon Kedar, Steve Chien, Frank Webb, Joshua Doubleday, Danny Tran, Ashley Davis, and David Pieri. Matt Welsh of Harvard University also gave us many insightful suggestions. This work is supported by the NASA ESTO AIST program and the USGS Volcano Hazard program under the research grant NNX06AE42G. The short version has been published in Mobisys 2009 [10].

REFERENCES


2. When we work on the revision of this paper in November 2009, the deployment would have also been done successfully. We will report the results in a future work.


Wen-Zhan Song received the BS/MS degree from Nanjing University of Science and Technology and the PhD degree from Illinois Institute of Technology. He is currently an associate professor in Department of Computer Science at Georgia State University. His research interests span pervasive computing, sensor networks, peer-to-peer networks, distributed systems, and algorithms. He has received grant support totaling more than $2 million from federal agencies, including the US National Science Foundation (NSF), the National Aeronautics and Space Administration (NASA), and the US Geological Survey (USGS), and private sources, including Boeing. He has published extensively in these areas, including a number of journal articles, conference papers, and book chapters, and serves many conferences TPC and journal editors. He is a member of the AGU, the IEEE, and the ACM.

Renjie Huang received the BS degree from South China University of Science and Technology and the Master’s degree from Huazhong University of Science and Technology. He is currently working toward the PhD degree in computer science from Washington State University. His research interests mainly focus on communication scheduling and optimization in wireless sensor networks. He is a student member of the IEEE.

Behrooz A. Shirazi is currently the Huie-Rogers chair professor and director of the School of Electrical Engineering and Computer Science. His research interests include the areas of pervasive computing, software tools, distributed real-time systems, scheduling and load balancing, and parallel and distributed systems. He has received grant support totaling more than $8 million from federal agencies, including the US National Science Foundation (NSF), the US Geological Survey (USGS), and private sources, including Boeing. He has received numerous teaching and research awards. He is currently the Editor-in-Chief for the Special Issues for Pervasive and Mobile Computing Journal and has served on the editorial boards of the IEEE Transactions on Computers and the Journal of Parallel and Distributed Computing. He is a member of the IEEE.

Mingsen Xu received the BS and MS degrees from Beijing University of Posts and Telecommunications. He is currently working toward the PhD degree in computer science at Washington State University, Vancouver, and is a research assistant at the SensorWeb Research Laboratory. His research interests focus on data correlation and reduction in sensor networks.

Richard LaHusen received the BS degree from the University of California, Davis, and completed 4 years of graduate work at Humboldt State University. Thereafter, he has worked for the last 20 years as part of the USGS Volcano Hazards Team and is currently a senior instrumentation engineer at the USGS Cascades Volcano Observatory. Throughout the 20-year period, he has developed instrumentation, telemetry systems, and software for the research and monitoring of volcanic processes and hazards (Murray et al., 1996). He also developed a low-powered, high-resolution earth deformation monitoring system that optimizes allocation of limited resources for GPS data acquisition and communication needs (LaHusen and Reid, 2000). Most recently, he introduced and applied aerial deployment of self-contained volcano monitoring instrumentation stations during the 2004-2005 eruption of Mount St. Helens (LaHusen et al., 2006).

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.