Senslide: A Distributed Landslide Prediction System

Anmol Sheth*
University of Colorado, Boulder

Chandramohan A. Thekkath
Microsoft Research, India.

Prakshep Mehta, Kalyan Tejaswi, Chandresh Parekh, Trilok N. Singh, Uday B. Desai
I.I.T. Bombay

Abstract

We describe the design, implementation, and current status of SENSLIDE, a distributed sensor system aimed at predicting landslides in the hilly regions of western India. Landslides in this region occur during the monsoon rains and cause significant damage to property and lives. Unlike existing solutions that *detect* landslides in this region, our goal is to *predict* them before they occur. Also, unlike previous efforts that use a few but expensive sensors to measure slope stability, our solution uses a large number of inexpensive sensor nodes inter-connected by a wireless network. Our system software is designed to tolerate the increased failures such inexpensive components may entail.

We have implemented our design in the small on a laboratory testbed of 65 sensor nodes, and present results from that testbed as well as simulation results for larger systems up to 400 sensor nodes. Our results are sufficiently encouraging that we intend to do a field test of the system during the monsoon season in India.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms

Algorithms, Design, Reliability

Keywords

Fault tolerant, landslide prediction, sensor network application

* Work done in part when the author was a summer intern at MSR India.

1. INTRODUCTION

This paper describes SENSLIDE, a distributed sensor system for predicting landslides. The idea of predicting landslides using wireless sensor networks was originally conceived at the SPANN Lab in the EE Department IIT Bombay. It arose out of a need to to mitigate the damage caused by landslides to human lives and to the railway network in the hilly regions of Western India. With support from Microsoft Research India and collaboration from the Earth Science Department IIT Bombay, SENSLIDE became a joint research project among these three research groups. Having an inter-disciplinary team with expertise in each area has been invaluable in coming up with a solution. The system uses a combination of techniques from Earth Sciences, signal processing, and distributed systems and fault-tolerance. We believe that our design principles are not specific to landslide prediction, and may be widely applicable in other contexts as well.

Landslides are serious geological hazards caused when masses of rock, earth, and debris flow down a steep slope during periods of intense rainfall and rapid snow melt. In our particular case, the western (Konkan) coast of India is subject to many such landslides every year. Landslides in this rocky region are mainly caused by the increase in strain due to percolating rain water in rocks fissures, causing rocks to fracture and slide down the slope. According to government reports, from 1998 to 2001 alone, landslides have killed more than 500 people, disrupted communication and transportation for weeks and destroyed thousands of hectares of crop area.

Existing solutions are restricted to landslide *detection*. A trip wire is installed along the railway track in the landslide prone areas, and a break in the trip wire due to the falling rocks and debris triggers an alarm. Although this is an inexpensive solution for landslide *detection*, it is ineffectual in providing warning of the impending landslide.

An obvious, and superior alternative to using sensors that detect landslides is to use sensors that *predict* landslides. Providing sufficient warning time before the impending landslide allows taking precautionary measures, minimizing the damage caused by the landslide. The precautionary measures could range from geological strengthening of the rocky surface, covering the hillside with a mesh to prevent the rocks from falling, evacuating people, stopping trains, etc.

Various such prediction sensors have been proposed in the literature [6]. Typical sensors used for monitoring slope stability are multi-point bore hole extensometers, tilt sensors, displacement sensors, and volumetric soil water content sensors. These require drilling 20-30 meter holes into the surface, making the installation very expensive (about $50 per meter) and requiring skilled labor. Furthermore, these are expensive sensors, making wide scale deployment infeasible. Installing a single sensor for monitoring an entire hill side is not sufficient as the properties of the rocks change every 100-200 meters. Wiring each sensor to a central data logger is also not feasible in the rocky terrain because it requires high maintenance, and is subject to a single point of failure.

In contrast to existing fixed single-point approaches, a fundamentally new approach to measure slope stability is by combining observations from a large number of distributed inexpensive sensors. SENSLIDE uses an array of inexpensive single-axis strain gauges connected to cheap nodes (specifically, TelosB motes [3]), each with a CPU, battery, and a wireless transmitter. Our sensors make point measurements at various parts of a rock, but make no attempt at measuring the relative motion between rocks. Our strategy is based on the simple observation that rock slides occur because of increased strain in the rocks. Thus, by measuring the cause of the landslide, we can predict landslides as easily as if we were measuring the incipient relative movement of rocks.

Geologists familiar with our terrain estimate that sensors can be separated by 30–40 meters. This would imply that we need about 600–900 sensors for each square kilometer of hillside surrounding the railway track. We call this collection of sensors a *sensor patch*, which is our basic unit of scale. The sensor patch contains only a modest number of sensors that we can reasonably expect our solutions to work at this scale. Yet, the patch is large enough that we can duplicate it conveniently to cover the stretches of the railway track that are most prone to landslides. In our current design, each patch is independent of other patches; for all practical purposes this appears to be a reasonable assumption.

The point measurements made by individual sensors are propagated to a set of "base stations" that have GPRS and/or 802.11 connectivity to each other. In addition to their improved network connectivity, base stations are located in places with access to ground power, and have more computation and storage resources than the sensor nodes. Each sensor patch has 3–6 base stations, both for increased fault-tolerance as well for limiting hop count between sensor nodes and base stations.

There are some advantages to using simple strain sensors. The strain sensors can operate at low depths (25–30 cms); the orders of magnitude lower depth of operation make strain gauge deployment much cheaper and more convenient without specialized equipment or a sophisticated labor force. The small size, low cost, and wireless connectivity allow denser coverage over a larger area without significantly increasing the difficulty or the cost of deployment.

1.1 Key Contributions

We believe our design is novel compared to previous sensor network systems because of the unique challenges we face in our environment. A potential issue with using small, cheap sensors is that failures are more likely to occur and system software has to compensate to keep the system functional. Similarly, the larger number of sensors implies that the system software must be scalable.

We also believe that as sensor networks become more widely deployed in harsh environments where failure is common, and power and network connectivity is unreliable, our design techniques will become increasingly more applicable. To our knowledge, though there have been sensor network systems deployed in hostile environments (e.g., volcanic monitoring [25]), typically these systems have not focused on fault-tolerance or scalability. As a result, they can suffer from network outages and low data yield due to the failure of system components.

In trying to design a system to solve the needs of our application, we encountered several challenges, which we briefly enumerate below. We use distributed systems algorithms as well as techniques from machine learning to provide solutions to these challenges. The combination of these solutions makes our design unique, which ensures fault tolerance at every level of the system and maximizes the lifetime of the system.

1. **Noisy sensor data:** Triggering an alarm based on locally sampled raw sensor data would lead to a large number of false positives and negatives. The sensor signal must therefore be smoothed before further processing.
2. **Fault Tolerance:** The system should not contain single points of failure, and should automatically adapt to communication links, nodes and base station failures.
3. **Spatial summary of sensor data:** Even though data from individual sensors is smoothed, single sensor observations are insufficient to predict a landslide; data from several spatially distributed sensors must be incorporated to detect anomalies indicative of a landslide. Without due care, such incorporation of data could lead to excessive communication.
4. **Unequal energy depletion in the network:** The protocol used for routing packets from the individual sensor nodes to the base station should avoid the formation of hotspots in the network. Hot spots lead to unequal energy drain at some nodes due to the increased energy requirements for transmitting and receiving packets. Nodes that are energy depleted can cause a network partition.
5. **Balance between rare event detection and periodic data collection:** Landslides are relatively rare events. Thus, we could potentially conserve power and never transmit data until a landslide is incipient. This conserves battery life in the network, but it could lead to a large number of false alarms and also defeats the purpose of timely gathering of seismic data [8].

We address the first issue through straightforward signal smoothing techniques.

We deal with failures of the base station by synchronously replicating data on multiple base stations. We deal with the node and link failures by using skeptics [20] and a modified version of the Beacon Vector Routing algorithm [9] to build a resilient data forwarding protocol in the sensor network.

We deal with the last three issues using the same mechanism: hierarchical decomposition. A subset of the sensor nodes are designated as *aggregators* that collect smoothed local data, and create *spatial summaries*. These aggregator nodes communicate with the base station providing summary data at adaptively adjusted frequencies. The location of the aggregators are chosen to spread the energy depletion uniformly in the network. We tolerate failures of the aggregators by periodic re-election of these nodes. The presence of aggregators in the network reduces the average hop-count and thus the number of radio transmissions required. It also increases the data yield of the network by avoiding congestion caused by transmitting all the sensor data to a centralized base station. The improved yield and uniform energy depletion of the network significantly improves the detection accuracy and lifetime of the network.

Although SENSLIDE uses inexpensive sensor nodes connected by a wireless network, our focus is quite different from previous sensor network projects we are familiar with. A majority of these projects [18, 24] focus on innovative data collection techniques, where recovery from failures is not the principal concern. In contrast, our primary objective is to provide a distributed sensor system that is robust in the face of failures.

The data collection strategy of SENSLIDE is also different from prior work. In fact, our work falls between two extremes: rare event detection schemes [11, 22] and periodic sampling networks where data is collected at a central base station for offline analysis. In our system, data must be sampled periodically to help earth scientists gather much needed historical trend information [8], while ensuring that the lifetime of the network is not adversely affected by frequent sampling. Existing data collection based applications use a tree-based routing algorithm to route all the data to a centralized base station. This naive mechanism does not scale with network size and leads to higher energy depletion of the nodes near the base station. Section 3 describes SENSLIDE's hierarchical decomposition technique. Although the technique is similar in principle to LEACH [13], it overcomes some of the limitations of LEACH which make it unsuitable for our application.

We are aware of one other landslide detection research being done at John Hopkins [23], although we are not aware of any implementation of that design. Our design differs from theirs in two aspects: (a) they process data only at the central base station, and (b) they do not explicitly deal with the failures of components.

We have tested the efficacy of our techniques by deploying the above described system in an indoor 65-node laboratory testbed. We have also tested the scaling behavior of our system on larger configurations (up to 400 nodes) using the ns-2 network simulator [2]. Encouraged by our results from the prototype, we plan to do our field tests during the monsoons in India.

Section 2 describes the characteristics of our rocks, the laboratory set up we use to measure the behavior of the rocks under strain, and the details of our detection algorithm. Section 3 describes our design in greater detail and Section 4 illustrates the performance of our system on our testbed and in simulation. Section 5 provides a complete treatment of related research, and Section 6 concludes our current project status and future plans.

2. DETECTING LANDSLIDES

In this section, we describe the characteristics of the rocks that are prevalent on the Konkan hillsides and our method for detecting an impending landslide.

2.1 Gathering Rock Information

The primary rock characteristic we care about is the stress-strain behavior. The kind of rocks that are found in the Konkan region of India are classified as volcanic igneous, which are a type of "brittle" rock. The stress-strain curves of igneous rocks exhibit linear behavior until there is a fracture. This near ideal behavior makes the terrain particularly well-suited to our technique.

Unfortunately, not all igneous rocks of the same type have the same stress-strain behavior. Individual variations in the crystalline structure as well as macroscopic variation in chemical composition and physical characteristics lead to observable differences in their behavior to stress.

We measured the stress-strain characteristics of five types of igneous rocks typically found in our environment. This study was done using well established techniques from the earth sciences [10, 6]. Each rock sample was "dressed" to form a "core": a cylinder of length 10.2 cm and a diameter of 5.1 cm. We apply stress along the longitudinal axis, using a universal testing machine and measure the strain produced in the core using our strain sensor attached to the core. The strain sensor outputs voltage values in proportion to the strain in the core. We increase the stress applied until the core fractures. Since we know the stress applied at any point and we can measure the strain produced by that stress, we can characterize the behavior of the rock, as well determine its fracture point.

Figure 1 shows the stress versus strain characteristics of five rock samples. The end-point of each series is the fracture point of the rock. Notice the linear slope of each rock sample until the fracture point.

The characterization depicted in Figure 1 is incomplete because it applies only when the axis of the strain sensors is aligned with the stress; whereas in general, the strain measured by the strain sensor depends on the orientation of the sensor relative to the axis of stress. In the field, we cannot orient the sensor along the axis of stress because we cannot determine the axis of stress with any certainty. Figure 2 shows the results of an experiment to study the effect of sensor position on the stress vs. strain behavior of the rocks. In this experiment, we attached multiple strain sensors at various orientations to the first rock core (Rock 1) shown in

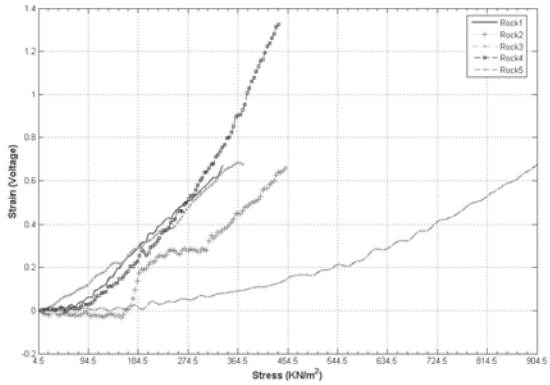

Figure 1: Measured stress vs. strain characteristics of igneous rocks. The X-axis is stress in Newton/m^2. The Y-axis is the A-to-D converted voltage in proportion to the strain on the rock. The variation in the stress-strain curves of the rocks is caused by both microscopic and macroscopic variations in each rock sample.

Figure 1.

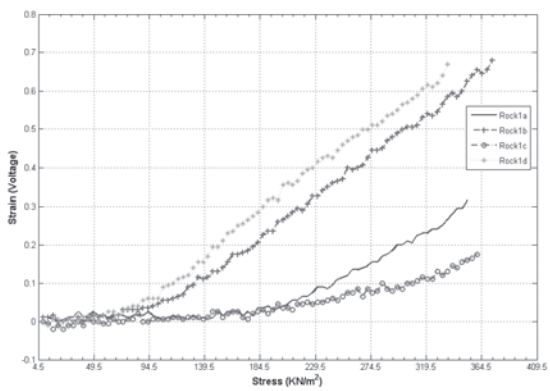

Figure 2: Effect of sensor orientation on stress-strain. The curve labeled Rock 1D is identical to the curve labeled Rock 1 in Figure 1. The remaining curves show slightly different behaviors because of the different sensor orientation.

Given the variations in the stress-strain characterization, we did a linear regression on the various rock samples and the multiple orientations of the sensors. Based on this analysis, we calculate the "average" stress-strain behavior, which can be characterized by a straight line, as well the average stress that causes a fracture in the rock. Clearly, such a model is simple, but our experiments indicate that it is adequate for our purpose.

2.2 Prediction Algorithms

In the simplest case, if the base station has strain readings from all the sensor nodes, it can quite easily predict if a landslide is likely to happen. We considered two alternative designs for landslide prediction.

Threshold Based Prediction

In this scheme, the base station receives strain readings from all the nodes. It uses the average stress-strain curve to calculate the stress on each point in the rock. It then averages the stress values and compares it with the stress that is known to cause a rock fracture. If the calculated average is above a certain threshold, it predicts a landslide.

In practice, it is impractical to assume that the base station receives readings from all nodes. In our system, we have designated nodes (called aggregators) that collect and summarize strain values from multiple sensors. The base station uses these summaries sent to it by the aggregators to calculate the stress. Section 3 elaborates on the rationale for using these aggregators.

The threshold based algorithm has two limitations, including the fact that it has no special mechanism to cope with strain values that are lost by failures. It also cannot cope with sensor readings that are corrupted by noise with a non-zero mean.

We deal with the first limitation by using more sensors than what is strictly required by the geological constraints of the environment and by using a resilient routing protocol. The second problem of noise is not a major concern because most noise sources in nature have a zero mean, and simple averaging (as done by the base station) is adequate to filter this noise.

Since we have not deployed the system in the field, we cannot verify the properties of the noise source. We therefore developed a statistical prediction technique that is more sophisticated and more tolerant of noise.

Distributed Statistical Detection Algorithm

Our basic strategy is to predict a landslide by using the well-known statistical inference technique of Bayesian hypothesis testing [21]. The null hypothesis, H_0 is that there will be no landslide, and hypothesis H_1 predicts that there will be a landslide.

In order to do the standard Bayesian test, we must first calculate the likelihood ratio. We assign *a-priori* probabilities to the outputs when each hypothesis is true, (say P_{H0} and P_{H1}). Let C_{00} and C_{11} be the cost of correctly predicting that there will be no landslide and correctly predicting the occurrence of a landslide. We assign these costs to be 0. Let C_{10} be the cost of a false alarm; mispredicting that there will be a landslide, when indeed there is no landslide. We assign this a small value. C_{01}, on the contrary, is a false negative and is assigned a very high value. To do a Bayesian test, we need to calculate the quantity

$$\eta \triangleq \frac{P_{H0}C_{10}}{P_{H1}C_{01}}$$

called the "threshold of the test".

We can then calculate the likelihood ratio and decide in favor of hypothesis H_0 if it is less than η and in favor of hypothesis H_1 otherwise.

To calculate the likelihood ratio, we model the actual stress-

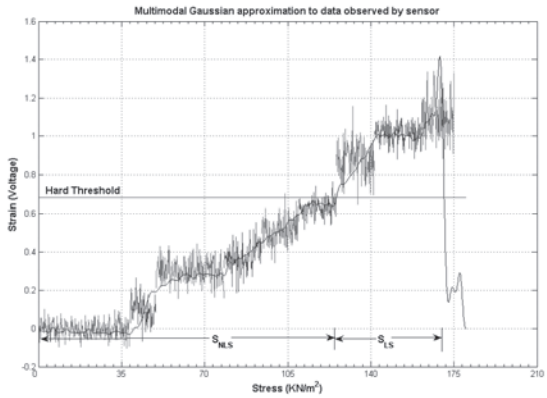

Figure 3: Typical stress-strain curve. A typical stress versus strain curve in the presence of noise.

strain strain curves seen in the *field* using multi-modal Gaussian random variables. Figure 3 shows a typical curve we see based on our experiments. We divide the curve into two regions— S_{NLS}, corresponding to no landslide, and S_{LS}, corresponding to landslides— based on a cut-off value of the stress. We then model the data in each region as a multi-modal Gaussian distribution. In our particular example, we have used 6 Gaussians in the S_{NLS} region, with mean and variance μ_{0i} and $\sigma_{0i}, i = 1..6$. Similarly we use 3 Gaussians in the S_{LS} region, with mean and variance μ_{1i} and $\sigma_{1i}, i = 1..3$.

Now, given k observations of the strain represented by the k dimensional vector α, the base station can calculate the likelihood ratio as:

$$L(\alpha) = \frac{\sum_{j=1}^{6} n_j \frac{1}{\sqrt{(2\pi)^k |R_{1j}|}} e^{\left(-\frac{1}{2}(\alpha-\mu_{1j})^T (R_{1j})^{-1}(\alpha-\mu_{1j})\right)}}{\sum_{i=1}^{3} d_i \frac{1}{\sqrt{(2\pi)^k |R_{0i}|}} e^{\left(-\frac{1}{2}(\alpha-\mu_{0i})^T (R_{0i})^{-1}(\alpha-\mu_{0i})\right)}}$$

μ_{ij} is a k dimensional mean vector, R_{ij} is a k x k covariance matrix for the j^{th} Gaussian distribution and hypothesis H_i, $i \in \{0, 1\}$. α is a k dimensional vector formed by data samples from k nodes. n_j and d_i are the weight vectors assigned to the Gaussians in the numerator and denominator.

Notice that we are taking a weighted-average of the Gaussians here, but for the moment, in the absence of field data, we use equal weights: 1/6 for each Gaussian in the numerator, and 1/3 for each Gaussian in the denominator. In practice, we expect that rock characteristics in different regions may require different weights.

Notice also that to compute the likelihood ratio, we must have efficient floating point calculations. Thus, this algorithm is computationally feasible at the base stations. Additionally, we also plan to implement a fixed-point version of the algorithm that could be executed on the sensor nodes.

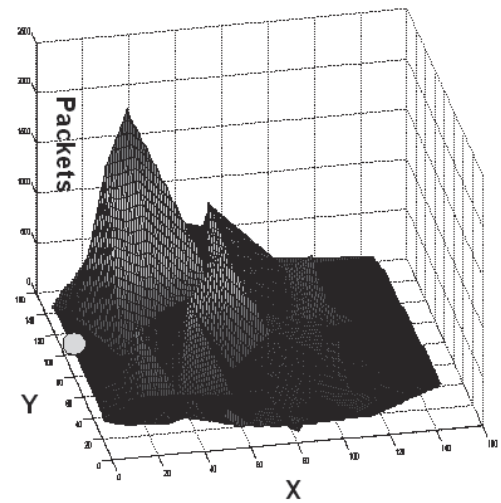

Figure 4: Non-uniform energy drain due to converge-casting traffic at base station The X-axis and Y-axis form a 160 × 160 meter grid. The Z-axis is the number of packets transmitted by the node. Nodes closer to the base station have to forward a higher volume of data, and hence exhaust their battery power at a higher rate. The circle at (0,100) represents the location of a single base station. Our experiments with multiple base stations confirm that a similar effect occurs in that case as well.

3. SYSTEM DESIGN

Our fundamental design goal is to arrange for the sensor data gathered at the sensor nodes to be delivered to a set of base stations so that we can run the prediction algorithm. Unfortunately this goal is complicated by two factors: the finite energy resources in the network, and the failures of sensors nodes and base stations.

Energy in the network is an important factor because our application needs periodic data. In most existing sensor network applications, sensor nodes periodically forward their data over multiple hops to a centralized base station. This data routing mechanism is ill-suited for long lived applications and leads to non-uniform energy drain in the network. As shown in Figure 4, nodes closer to the base station drain their battery power at a much faster rate and eventually lead to disconnection of the entire network. We solve the energy depletion problem by using a two-level hierarchy. A subset of the nodes are designated as *aggregators* that collect the locally smoothed sensor data and create summaries, which are then communicated to the base station that is closest to the aggregator node. Base stations are located along the railway tracks with access to ground power, in addition to GPRS and WiFi connectivity.

Although our use of aggregator nodes is similar to LEACH [13], our design does not have some of its limitations, as elaborated in Section 5. Our algorithms for energy conservation and fault-tolerance are facilitated by dividing time into fixed interval or "epochs", which are currently 15 minutes long in our implementation.

In the next section we describe the details of our strategy to have uniform energy depletion. The subsequent section describes the details of our fault-tolerance strategy.

3.1 Energy Depletion

Duty Cycling the Nodes

Each sensor node implements a low duty cycle sleep-wakeup schedule to conserve battery power. Having the micro-controller and radio idle when no data is being sensed or forwarded reduces the lifetime of the network to a few days. Each sensor node wakes up once per epoch and stays awake for a quarter of the epoch before going back to low power sleep. This interval (about 4 minutes) is sufficient for each node to sample the ADC, to transmit beacons for the routing protocol, and forward data to the closest aggregator node as described below.

Duty cycling the sensor nodes requires loose time synchronization of the sensor nodes with the base stations. Once every hour, the base station broadcasts a time synchronization packet containing a sequence number and a timestamp. On receiving this packet, sensor nodes synchronize their local clocks with the time stamp embedded therein and broadcast the packet downstream. Broadcasts are limited by embedding a Time-To-Live (TTL) counter. The local clock is not updated if a time update with a higher sequence number is already received. Also, time synchronization packets are given higher priority in the radio transmission queue to reduce queuing delays.

Clearly, our time synchronization protocol is not very accurate. When deployed on an indoor testbed of 50 nodes with a network diameter of 4 hops, it resulted in an error of 1-2 seconds. However, this magnitude of error is much smaller than the time scales required by a seismic application such as Senslide. If this proves to be a problem in the field deployments, we plan on implementing a low overhead accurate time synchronization protocol in the future.

Routing and Aggregation

In addition to duty cycling nodes, it is important to pay attention to routing in order to achieve uniform energy depletion in the network. To avoid the non-uniform energy depletion shown in Figure 4, Senslide uses aggregator nodes to compute summaries of the strain data received from the sensor nodes, and forwards only the summary data to the base stations. (We use multiple base stations to deal with failures.) This significantly reduces the volume of data forwarded by nodes close to the base stations, and also reduces the number of hops over which the data packets are forwarded.

Existing in-network aggregation schemes like TAG [17] use a tree-based aggregation scheme, where a parent node aggregates data from its child nodes before forwarding the data upstream to the central base station. The three main limitations of TAG that make it unsuitable for our deployment are: (a) Complex distributed statistical detection algorithms often require access to the raw strain data from the distributed sensors, (b) TAG supports only simple aggregation functions like MIN/MAX, AVERAGE etc., and (c) TAG aggregates data only along a tree based topology rooted at a central base station.

Implementing Senslide's hierarchical structure requires in-network *point-to-point* routing rather than a traditional tree based routing. We use Beacon Vector Routing (BVR), a greedy link-state point-to-point routing protocol [9], because it is robust under failures, and outperforms other point-to-point protocols with which we are familiar.

We modify the BVR algorithm such that the base stations in the network are permanently assigned as the beaconing nodes. Each base station sends out beacons once an epoch. These periodic beacons are used by every sensor node to calculate an n-dimensional coordinate vector, where the i^{th} element of the coordinate vector is its hop count to the i^{th} base station. Thus every sensor node is located within the "n-dimensional" virtual coordinate system based on radio connectivity. BVR routes data from the sensor nodes to the aggregator nodes based on their coordinates.

At the end of every three epochs, each sensor sends status information to its nearest base station. The status message contains the node's BVR coordinates, energy level, and the list of its neighboring nodes. The status messages received at the base stations are collected at a designated *leader base station*, where an aggregator selection algorithm is executed.

It is important to emphasize that the BVR coordinates do not necessarily reflect geographical coordinates. That is, two nodes that are close to each other physically may be out of radio range from each other and may have very different BVR coordinates based on their hop counts to the beacon sources. This difference has to be factored in when aggregation points are selected as described below.

Our aggregator selection algorithm satisfies the following constraints.

1. An aggregator node must be located at a suitable BVR coordinate so that several nodes require only a few hop counts to reach it, and it, in turn can reach the base stations in a few hops.

2. Aggregator nodes should not be geographically co-located. This ensures that a localized failure in the terrain does not cause all aggregator nodes to stop functioning.

3. An aggregator node should have sufficient residual battery power to process the data collected from the neighboring nodes

4. An aggregator node must be well-connected to the rest of the network so as to avoid formation of hot links.

The first constraint is satisfied by using k-means clustering— a well-known unsupervised learning algorithm [16]— to select k aggregator nodes in the network. The radio coordinate vectors of the nodes are used to select the k aggregator nodes, such that the distance in terms of radio hops is minimized to route data to the closest aggregator node. This effectively reduces the energy consumption of the network.

The second constraint can be readily satisfied because the base station has complete information of the physical location of the nodes. Thus, given multiple sets of aggregator nodes, the set of aggregator nodes whose pair-wise geographic distance is the largest is selected.

The third and fourth constraints are satisfied by the information present in the status messages. The number of neighbor nodes and energy level of the aggregator nodes are used to satisfy the two constraints.

It is not always possible to satisfy some of these constraints simultaneously. At each stage of the aggregator selection algorithm, sets of aggregator nodes that do not satisfy the constraint are eliminated. If at any point in the aggregator selection algorithm the set of aggregator nodes is empty, we relax the constraint and select the aggregator nodes from the non-empty set.

3.2 Fault Tolerance

The critical nature of SENSLIDE requires the system not have any single points of failure. SENSLIDE achieves fault tolerance by introducing redundancy at various levels of the system.

Intermittent radio link failures

Prior research [26, 27] has shown that dense wireless networks consisting of the low power radios [1] exhibit high temporal and spatial variation of link quality. Achieving packet delivery guarantees in the presence of varying and intermittent link failures requires a careful selection of links over which data is routed. To deal with intermittent link failures, SENSLIDE uses a skeptic [20] to reduce the failure rate of an intermittent link and weed out the low quality links. Link qualities are updated every epoch and only links above a preset threshold of 65% are used to route data. The three main properties of a skeptic [20] that are used by SENSLIDE are as follows:

- Links with good histories must be allowed to fail and recover without significant penalty. SENSLIDE achieves this by maintaining a weighted moving average of the link quality and marking the link as "live" only if it is above the threshold.
- A link's average failure rate must not be allowed to exceed a threshold. SENSLIDE achieves this by counting the number of time the quality of a link falls below the threshold. If the count exceeds a threshold over three epochs, the link is permanently marked "dead" and not used.
- A link that stops being bad should eventually be forgiven. SENSLIDE achieves this by raising the link quality threshold to 80% for links recovering from intermittent failures.

Existing link quality estimation techniques only maintain a weighted mean of the link quality [26]. Thus, even though the link quality estimate tracks the actual link quality closely, it would not be able to limit the failure rate of the link. A node having many such intermittently failing neighbor nodes could have a different coordinate vector each epoch. The second property of the skeptic listed above limits the failure rate of such intermittently failing nodes, and hence avoids such neighbors when estimating a nodes coordinate vector. Thus, the skeptic prevents unnecessary aggregator re-selections caused due to changing coordinate vectors of the nodes.

Sensor node failures

Our design tolerates the failure of sensor nodes by the use of periodic beacons and skeptics. Periodic beacons transmitted by the base stations automatically update the coordinate vector of a node in presence of node failures. Also, since BVR makes a locally greedy decision at each hop to route data, failed nodes are bypassed.

To deal with errors due to malformed packets that bypass the CRC error checks and other memory corruption errors, we enable the hardware watchdog timer on the TelosB mote to reset the node once every 12 epochs. Before a node is reset, it writes a small amount of state information to non-volatile memory, and reads it back after reset. The state information stored in the flash includes the node's unique ID and its sequence number. After every reboot a node reconstructs its state information (route table and coordinate vector). This ensures that a sensor node is always left in a known state after a reboot and can recover from the transient errors.

Aggregator node failures

Aggregator failure is detected by monitoring the loss rate at base stations since aggregators transmit periodic summary data to the closest base station. Base stations keep track of aggregator liveness by monitoring sequence numbers in the data packets received from the aggregator. An aggregator failure triggers the aggregator node selection algorithm. Changes in aggregator nodes and other control messages are broadcast by the leader base station to the entire network.

Base station failure

Base stations are PCs that have wired power and network connectivity. We can therefore deal with failures of the stations using any well established technique such as replicated state machines [15]. We have not implemented anything so elaborate currently because of time constraints, but we have implemented a simple solution that provides reasonable fault-tolerance.

Base stations use ping messages to monitor the liveness of each other. The base station with the largest IP address elects itself the leader for the purpose of executing the aggregator selection and detection algorithms. We do not handle network partitions. Data received at the base station is synchronously replicated to the others. This includes summaries from aggregators and status messages from nodes. The replicated status messages from nodes allow us to redo the aggregator selection algorithm if a base station dies. The replicated summary data allows to reconstruct the strain over the entire patch in the presence of base station failure.

Since BVR is a distance vector based routing protocol, failure of a base station causes the sensor nodes to start counting to infinity. The count-to-infinity problem is solved by

judiciously selecting the maximum diameter of the network. Hence, if the coordinate of the failed base station increases above the preset maximum diameter, the node sets the coordinate to infinity. In our testbed sensor nodes detect base station failures within two epochs.

4. SYSTEM EVALUATION

We have not deployed our system in the field yet, but we have evaluated its behavior in our laboratory testbed as well as done simulation studies.

4.1 Laboratory Testbed Setup

We have set up 65 Mica TelosB motes interconnected by a wireless network and three base stations in our lab at the University of Colorado. These emulate a single patch, albeit at a scale that is an order of magnitude smaller than actual. Our testbed has a maximum diameter of 6 hops.

Each TelosB mote consists of a 8 MHz microcontroller with 10 KB RAM and 1 MB of external flash memory and runs the MANTIS operating system [4]. The radio is a 2.4 GHz IEEE 802.15.4 compliant with a maximum data rate of 250 Kbps. The mote also has a 8 channel 12-bit ADC converter. Each base station in the testbed is a regular Linux desktop. The base stations are networked together using a 100 Mbps wired network.

In addition to the wireless network, the TelosB motes are also connected to a wired USB backbone network, which is used to download software, induce faults, collect debug and status information, and for other controlled experimental steps.

We have collected strain data from rock specimens in our laboratory at IIT Bombay (as described in Section 2). We load a set of these strain measurements, after the addition of suitable noise, into the external flash memory of each TelosB mote in our testbed in Colorado. The additive noise comes from a zero-mean Gaussian noise source, and constitutes a significant fraction— up to about 40%— of the raw strain value. The raw strain values (before the noise is added to them) increase in magnitude at a rate that would be typical during an impending landslide in the field. Each time the ADC in the TelosB is sampled, we read a new strain sample from the flash memory.

4.2 Testbed Results

We did two sets of experiments to evaluate the energy drain characteristics of our patch, and the effect of using aggregators in the design.

In the first set of experiments, we used in-network aggregators, but varied their number so that data from 13, 26, or 52 TelosB motes are routed to each aggregator. Base stations were used in these experiments, but not as aggregators, only for generating BVR beacons. In this arrangement, each mote forwards data only to its aggregator (as chosen by the k-means clustering algorithm), so they make fewer network hops than if they went to a base station. Furthermore, the presence of aggregators in the interior of the network leads to less congestion near the base station.

In the second set of runs, we did not use aggregators, instead all aggregation was done at one or more of the base stations. Again, we varied the number of aggregation points, so that data from 13, 26, or 52 TelosB motes were sent to each aggregation point.

We are interested in two characteristics with respect to energy drain. First, energy drain should be uniform, so that hotspots don't develop in the network causing disconnection. Second, the average drain must be small, so that we can stretch our meager energy resources to the fullest extent. In-network aggregators can have a significant effect on both of these metrics, because their presence alters the number of packets each node has to transmit, which in turn has a measurable impact on energy drain.

Figure 5 plots the Jain's fairness index [14] for the two approaches. Jain's index is a commonly used metric for fairness and is similar to using standard deviation. It is bounded in the interval [0,1], with 1 indicating that each node transmits the same number of packets, and hence has the same energy drain.

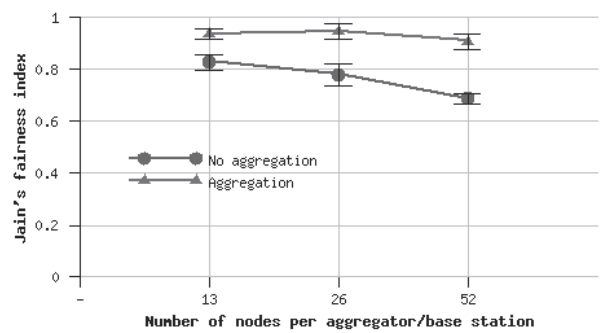

Figure 5: Uniform energy depletion in the presence of aggregators The X-axis is a measure of the number of aggregators in the system. The Y-axis is a measure of the variance in the energy depletion levels in the motes.

Our motes wake up every 15 minutes, and process data and then go back to sleep. During the sleep phase, they draw a current of a few micro amperes, which we ignore for this calculation. During normal processing, each packet transmission draws 18 mA, while an idle radio consumes 15 mA. In addition, we also account for the current drawn by the real strain sensor (12 mA). The battery is rated at 1800 mA-h. Figure 6 plots the lifetime of the motes. Again, with aggregation, it appears likely that our system can last a monsoon season in the Konkan region without running out of energy.

The results from Figures 5 and 6 are encouraging and support our decision to use aggregators for effective energy usage.

The next experiment evaluates the accuracy of our system. We measure accuracy as the number of landslides that are correctly predicted. Mispredictions can occur with our system due to three reasons:

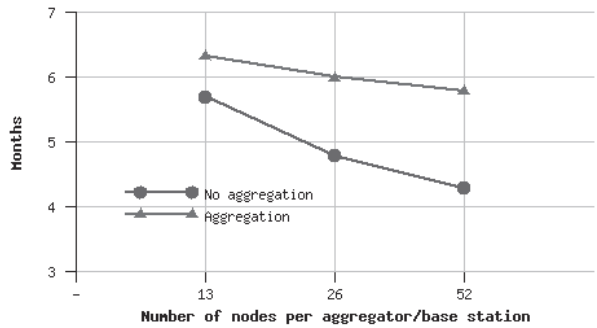

Figure 6: Battery lifetime in the presence of aggregators. The X-axis is a measure of the number of aggregators in the system. The Y-axis shows that our system can, quite comfortably, outlast a monsoon season, which typically lasts 3-4 months.

- Noise in the strain values.
- Dropped packets. When there is increased congestion in the network, especially near the base station, packets get lost or corrupted.
- Node failure.

We have not studied the effect of node failures in the our testbed, but we study its effect in simulation and report on it in the next section. Here we describe our experiments to study the effects of the other two factors on accuracy.

Our prediction scheme is based on using a threshold, i.e., a fraction of the critical stress value that will cause a landslide. A landslide is predicted if we measure stress beyond this threshold. The value of the threshold and the rate at which stress builds up will determine how long before the actual event we will be able to make a prediction, assuming ideal noise-free and fault-free conditions. We refer to this time interval as the "lead time", denoted by Δ.

In real-life, we would pick a value for Δ based on how long we would need to take precautionary actions against a landslide. Fixing Δ and knowing the historical behavior of the stress build in rocks during landslides, we would select a suitable threshold. Based on the best estimates of geologists familiar with the terrain we use a value of 24 hours for Δ and pick a threshold of 65%.

Due to noise and other inaccuracies, a system may not be able to predict a landslide within this lead time. This results in false negatives, a condition we would like to avoid in practice.

The series of raw (noise-free) strain values gathered from the lab in IIT Bombay that is loaded in each sensor node always results in a landslide after some time. Call this time T. We say that a landslide prediction during the interval T-Δ and T, is a *true positive*. A prediction after time T, is a *false negative*, which can have dire consequences in real-life. With our current set of data samples, it is impossible in principle to have a *false positive*. However, for our purposes, we treat a landslide predicted before T-Δ as a *false positive*. This models our expectation that if noisy sensor data temporarily leads to a high value of stress in the rocks, our system should not predict a landslide.

Figure 7 shows the accuracy of our predictions in the presence of noise and dropped packets. We have also shown the beneficial effects of aggregators, which mitigate the congestion and leads to fewer corrupt packets. The absolute percentage values are a function of the specific values of the threshold and Δ we have chosen. However, the trend lines are more widely applicable and show that with aggregators we can reliably achieve a given level of prediction accuracy.

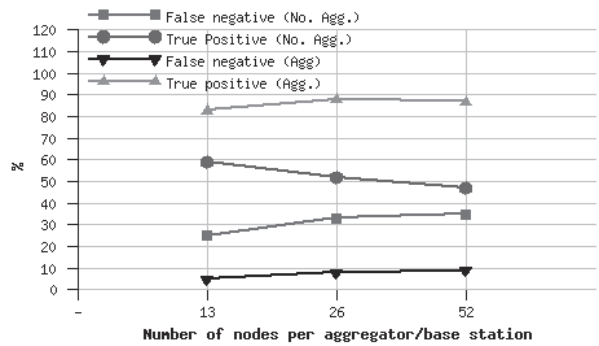

Figure 7: Accuracy in the presence of noise and network congestion. The X-axis is a measure of the number of aggregators. Aggregators lessen the ill effects of network congestion. False negatives are mispredicted landslides which can cause major loss to life and property.

In addition to the above, in practice we have a number of other effects that we would like understand. In particular, we would like to understand how lost packets, node, aggregator, and base station failures, and scale affect the energy profile of the system (and hence its longevity) and the overall accuracy of landslide predictions. We study these properties of our system using simulation.

4.3 Simulation Results

We implemented SENSLIDE in the ns-2 simulator [2], a discrete event network simulator. We modified the default parameters of the ns-2 radio model to match those of the CC2420 [1] radio in the TelosB motes. The 802.15.4 MAC protocol was used with link layer acknowledgments turned off. In addition, based on extensive measurements on the testbed, we build a simple radio link loss model for the simulator that resembles the link losses observed on the actual testbed. As part of the link loss model, we attach a probability of 0.87 to successfully forward a packet to the next hop and a probability of 0.13 to drop a packet.

In the simulations, we vary the size of the sensor patch from from 100m $\times$ 100m to 500m $\times$ 500m, but keep the density of motes in the patch fixed. This results in the smallest patch having about 20 motes to the largest patch having 400 motes. Also a patch of 200m $\times$ 200m contains 64 motes,

which is similar to the size of our laboratory testbed. We validated the simulator results from this patch with the real testbed.

Each simulation uses three base stations and a varying number of aggregators, proportional to the square-root of the number of sensor nodes in the patch. The sensor nodes were randomly selected within the patch, and for each patch we average results from 15 different random topologies.

The first metric we wished to quantify was "yield". Yield is measured as the percentage of motes from which data has been received at an aggregation point. A low value of yield leads to incomplete information about the spatial distribution of strain on the rock, and could lead to inaccurate detection.

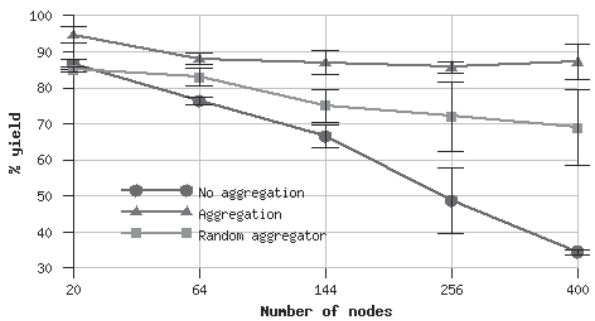

Figure 8: Variation of yield with scale. The yield is measured as an average over many epochs. Aggregator nodes selected by the k-means clustering algorithm provides higher yield and also scales with the patch size.

Yield is affected by scale. As the scale increases, more nodes send their data to the base station. As a result, more packets are dropped closer to the base station due to increased contention and packet collisions. However by selecting intermediate aggregator nodes that are optimally located, the sharp drop in yield is avoided. Figure 8 shows the effect of scale on yield. Yield is averaged over many epochs. (Recall from Section 3 that every epoch, sensor nodes wake up, sample the strain sensor and forward the data to the closest aggregation point.) The figure shows the beneficial effects of in-network aggregation. A random aggregator selection leads to a noticeable loss and variability in the yield.

Yield is also obviously affected by node failures, and we would like our system to be resilient in the presence of failures. We anticipate that the inexpensive hardware and harsh environmental conditions in which the system is deployed will lead to the failure of sensor nodes. To test the behavior of the system in presence of failures, we performed simulations for a fixed patch size of 400m $\times$ 400m and increased the number of sensor nodes that are failed. Sensor nodes were randomly selected to model random independent failures. We have not investigated the effects of correlated node failures.

Figure 9 shows the yield as the number of failed nodes are

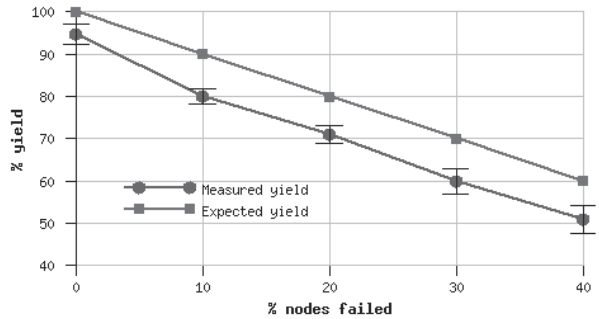

Figure 9: Variation of yield with node failures. Expected yield is what we would theoretically predict when a certain fraction of the nodes have failed. Measured yield is the yield that is observed in practice. The patch size is 400m $\times$ 400m.

increased. From the graph we observe that the difference between the measured yield and the expected yield remains almost constant as the percentage of failed nodes is increased. This fixed penalty can be attributed to packet loss due to lossy links and congestion, and roughly about 8% for this specific patch size. For smaller patches, the penalty would be slightly lower.

Having analyzed the variations in yield, we describe how the accuracy of the prediction is affected by this phenomenon due to scale and by node failures.

Figure 10 demonstrates the accuracy of the prediction algorithm as the network is scaled. The specific percentage values of the curve are dependent on the specific threshold (65%) and lead time (24 hours) values we chose. However, the variation in the accuracy as scale changes is of greater concern. We observe from the graph that some form of in-network aggregation appears necessary, both for sufficient absolute values and for lowered variance.

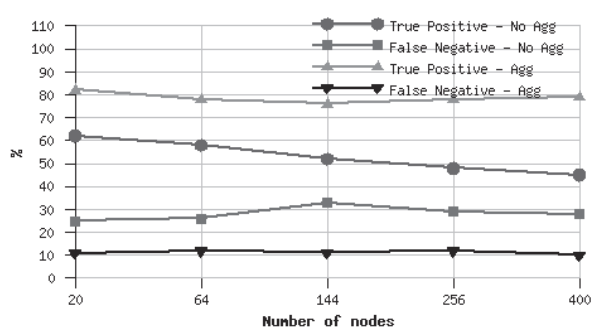

Figure 10: Accuracy of detection remains high as the network is scaled.

Figure 11 shows the detection accuracy when we introduce node failures into a patch of 400m $\times$ 400m. Normally, we would expect overall accuracy to be significantly affected by yield, and recall from Figure 9, that the yield falls from

about 92% to about 50% as the number of failed nodes increases. Nonetheless, the detection accuracy in Figure 11 remains largely unaffected. This is because we have used more motes than we strictly need to satisfy geological constraints, and a 50% yield is adequate for good prediction accuracy. For this over-provisioning strategy to work, we need a technique to keep additional losses in the network (due to congestion and packet loss) to an acceptable level. This is precisely the safeguards our in-network aggregator selection algorithm and routing protocols provide.

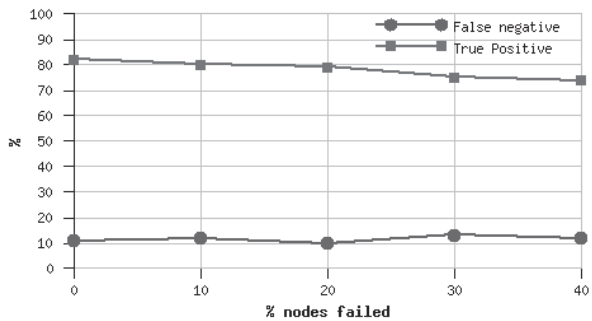

Figure 11: Accuracy of detection remains high as nodes are failed in a patch of $400\mathbf{m} \times 400\mathbf{m}$.

The last two experiments we describe study the energy profile of the system as the size of the system increases.

Figure 12 shows the energy drain in the system as it grows in size. Nodes closer to the base station consume much more energy due to the high volume of data that needs to be forwarded to the base station. Hence, hotspots are formed as the network size increases, which leads to unequal energy drain in the network. By selecting aggregator nodes and rotating them through the network, SENSLIDE ensures uniform energy drain across the network, i.e., the Jain's index in presence of aggregation is near optimal. Randomly selecting aggregation points in the network often leads to suboptimal aggregator selection leading to non-uniform energy drain and the formation of hotspots in the network.

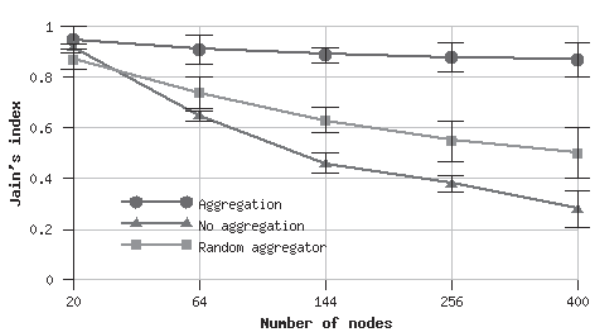

Figure 12: Energy depletion is uniform and scales with patch size.

The second important metric for energy consumption is the overall lifetime of the network. In our simulations, we consider a network to be disconnected when more than 30% of the nodes one-hop away from the base station deplete their battery power. Figure 13 shows the lifetime of the network. Energy consumption was measured using the parameters presented in Section 4.2. Without aggregation, nodes closer to the base station deplete their energy much faster due to the unequal energy drain. Hence, the lifetime of the network does not scale with the network size. However, with aggregation lifetime is not severely affected by the scale of the network.

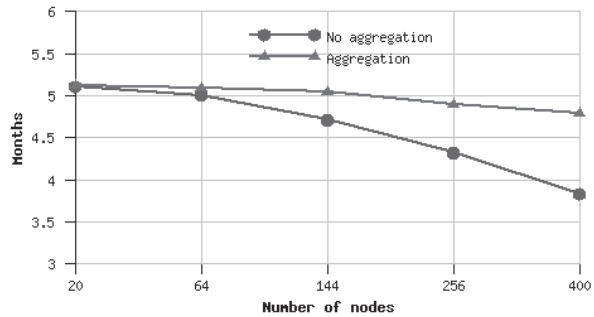

Figure 13: Lifetime of the network remains constant as the network is scaled.

5. RELATED WORK

Applications of sensor networks have become increasingly common. Existing sensor network applications can be broadly classified into data collection networks [7, 18, 24] and rare event detection networks [5, 11, 22]. The work described by Terzis et al. [23] is most closely related to SENSLIDE. Their paper describes a technique to use a Finite Element Model to detect the formation of a slip surface separating the sliding part of the hill from the static one. However, it provides little detail on the overall system design and implementation.

The two other earth science application deployments of sensor networks that we are aware of are GlacsWeb [19] to understand the behavior of ice caps and glaciers, and the work by Werner-Allen et al. to monitor volcanic eruptions [25]. The volcano monitoring project was a small scale (16 sensors) and short term (19 days) deployment. Unlike the design goals of SENSLIDE, the primary goal of the volcano monitoring deployment was to ensure high data yield rather than fault tolerance or scalability. In fact, the paper describes several single points of failure that lead to total absence of volcanic data during the deployment. We feel that the fault tolerance and energy conservation mechanisms of SENSLIDE complement such projects, and can be applied to eliminate single point of failures and increase the lifetime of the deployment.

Most environmental monitoring applications of sensor networks are "sense and forward" applications where data is aggregated at a central base station [12, 24]. The canonical system design for such applications involves periodic transmission of sensor data to a central base station. However,

these systems lack fault tolerance and have several single points of failure. Existing system designs for rare event detection like sniper localization [22], intrusion detection systems [5, 11] only communicate with the base station when the target is detected. However, the application goals of SENSLIDE require a fine balance to be maintained between rare event detection and periodic data collection.

In-network aggregation, as exemplified by TAG [17], is a commonly used data reduction mechanism in sensor networks. Our technique for aggregation differs significantly from TAG as already detailed in Section 3.

SENSLIDE's hierarchical decomposition of the sensor nodes into a two level hierarchy is similar to LEACH [13]. However, LEACH assumes that each sensor node is a single hop away from the aggregator node, requires each cluster head to be non-interfering with the others and requires 3 rounds of communication to setup a TDMA schedule. These features of LEACH make it ill-suited for a large class of applications, including SENSLIDE.

There have been a large number of studies that demonstrate the high variability of link qualities observed in dense deployments of sensor networks [26, 27]. In addition to the above, recent deployments of sensor networks have also reported low yields [12, 24]. SENSLIDE deals with low data yields due to the varying intermittent links by using skeptics [20]. A skeptic limits the failure rate of a link by delaying the recovery of a "bad" link, blacklisting links with a high failure rate and allowing "good" links to recover quickly from a failure.

To compensate for the low yield due to the varying link qualities of the sensor radios and noisy sensor data due to the low cost uncalibrated sensors, applications are required to incorporate signal processing algorithms in the system design. Due to the limited resources on the sensor nodes, most applications [22, 24] perform the complex signal processing at a centralized powerful server. In contrast, there are applications which perform the signal processing in-network [5, 11] and communicate with a central server only when an anomaly is detected. SENSLIDE adopts a hybrid approach, where the aggregator nodes perform the spatial summary of data and the base aggregates these spatial summaries from distributed aggregator nodes to trigger the warning.

6. CONCLUSION AND FUTURE WORK

In this paper we described the design, implementation, and evaluation of SENSLIDE, a distributed landslide prediction system. In contrast to existing sensor network applications, our primary objective is to build a distributed sensor system that is robust in the face of failures.

A unique feature of our design is that we combine several distributed systems techniques to deal with the complexities of a distributed sensor network environment where connectivity is poor and power budgets are very constrained, while satisfying real-world requirements of safety. We also believe that this combination of techniques will be more important in the future when sensor networks are widely deployed in environments where failures and intermittent connectivity have to be tolerated without compromising the correct functioning of the system.

Although SENSLIDE is not yet deployed in the field, we have built a 65-node indoor sensor network and also tested it in simulation. Based on laboratory experiments we have characterized rocks that are typically found in the landslide prone areas, and used this real data to drive the experiments in the testbed and simulations.

SENSLIDE hierarchically structures the network into sensor nodes and aggregator nodes. This hierarchical structuring of the network improves the data yield and ensures uniform energy depletion across the network. The improved yield and uniform energy depletion of the network significantly improves the detection accuracy and lifetime of the network. Our simulation results also show the robustness of the system design. By incorporating sufficient redundancy into the system and low overhead health monitoring, SENSLIDE can tolerate failures of sensor nodes, aggregator nodes, and base stations.

As part of future work, we plan to build improved models of the stress versus strain characteristics of the rocks. We also intend to field test the system during the monsoon season in India.

7. REFERENCES

[1] Chipcon cc2420 radio data sheet. http://www.chipcon.com.

[2] The network simulator - ns2. http://www.isi.edu/nsnam/ns.

[3] Telosb mote platform. http://www.moteiv.com/.

[4] ABRACH, H., BHATTI, S., CARLSON, J., DAI, H., ROSE, J., SHETH, A., SHUCKER, B., DENG, J., AND HAN, R. Mantis: system support for multimodal networks of in-situ sensors. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications* (New York, NY, USA, 2003), ACM Press, pp. 50–59.

[5] ARORA, A., DUTTA, P., BAPAT, S., KULATHUMANI, V., ZHANG, H., NAIK, V., MITTAL, V., CAO, H., DEMIRBAS, M., GOUDA, M., CHOI, Y.-R., HERMAN, T., KULKARNI, S. S., ARUMUGAM, U., NESTERENKO, M., VORA, A., AND MIYASHITA, M. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks* (2004), 605–634.

[6] BROWN, E. T. Rock characterisation, testing and monitoring. ISRM suggested methods. *International of Journal of Rock Mechanics and Mining Science and Goemechanics Abstracts 18* (December 1981), 109–110.

[7] CERPA, A., ELSON, J., ESTRIN, D., GIROD, L., HAMILTON, M., AND ZHAO, J. Habitat monitoring: application driver for wireless communications technology. *SIGCOMM Comput. Commun. Rev. 31*, 2 supplement (2001), 20–41.

[8] EBERHARDT, E., STEAD, D., STIMPSON, B., AND READ, R. S. Identifying crack initiation and propagation thesholds in brittle rock. *Canadian Geotechnical Journal 35*, 2 (1998), 222–233.

[9] Fonseca, R., Ratnasamy, S., Culler, D., Shenker, S., and Stoica, I. Beacon vector routing: Scalable point-to-point in wireless sensornets. In *NSDI '05: Proceedings of the 2nd Symposium on Networked Systems Design and Implementation* (Boston, MA, United States, 2005).

[10] Goodman, R. E. *Introduction to Rock Mechanics*. Wiley, New York, 1980.

[11] Gu, L., Jia, D., Vicaire, P., Yan, T., Luo, L., Tirumala, A., Cao, Q., He, T., Stankovic, J. A., Abdelzaher, T., and Krogh, B. H. Lightweight detection and classification for wireless sensor networks in realistic environments. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems* (New York, NY, USA, 2005), ACM Press, pp. 205–217.

[12] Hartung, C., Seielstad, C., Holbrook, S., and Han, R. Firewxnet a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments. In *MobiSys '06: Proceedings of the 4th International Conference on Mobile Systems, Applications, and Services* (Uppsala, Sweden, June 2006).

[13] Heinzelman, W., Chandrakasan, A., and Balakrishnan, H. An application-specific protocol architecture for. In *IEEE Transactions on Wireless Communications* (2002), vol. 1.

[14] Jain, R., Chiu, D., and Hawe, B. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *DEC Research Report TR-301* (1984).

[15] Lamport, L. The part-time parliament. In *ACM Transactions on Computer Systems* (May 1998), vol. 16, pp. 133–169.

[16] MacQueen, J. B. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability* (Berkeley, California, USA, 1967), University of California Press, pp. 281–297.

[17] Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. Tag: a tiny aggregation service for ad-hoc sensor networks. In *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation* (New York, NY, USA, 2002), ACM Press, pp. 131–146.

[18] Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications* (New York, NY, USA, 2002), ACM Press, pp. 88–97.

[19] Martinez, K., Hart, J. K., and Ong., R. Environmental sensor networks. In *IEEE Computer* (2004), vol. 37, IEEE, pp. 50–56.

[20] Rodeheffer, T. L., and Schroeder, M. D. Automatic reconfiguration in autonet. In *SOSP '91: Proceedings of the thirteenth ACM symposium on Operating systems principles* (New York, NY, USA, 1991), ACM Press, pp. 183–197.

[21] Savas, O., Alanyali, M., Venkatesh, S., and Aeron, S. Distributed Bayesian hypothesis testing in sensor networks. In *Proceeding of the 2004 American Control Conference* (June 2004), pp. 5369–5374.

[22] Simon, G., Maroti, M., Ledeczi, A., Balogh, G., Kusy, B., Nadas, A., Pap, G., Sallai, J., and Frampton, K. Sensor network-based countersniper system. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems* (New York, NY, USA, 2004), ACM Press, pp. 1–12.

[23] Terzis, A., Moore, K., Anandarajah, R., and Wang, I. Slip surface localization in wireless sensor networks for landslide prediction. In *IPSN '06: Proceedings of the 5th International Symposium on Information Processing in Sensor Networks* (Nashville, TN, USA, April 2006).

[24] Tolle, G., Polastre, J., Szewczyk, R., Culler, D., Turner, N., Tu, K., Burgess, S., Dawson, T., Buonadonna, P., Gay, D., and Hong, W. A macroscope in the redwoods. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems* (New York, NY, USA, 2005), ACM Press, pp. 51–63.

[25] Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., and Welsh, M. Fidelity and yield in a volcano monitoring sensor network. In *OSDI '06: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation* (Seattle, WA, USA, 2006).

[26] Woo, A., Tong, T., and Culler, D. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems* (New York, NY, USA, 2003), ACM Press, pp. 14–27.

[27] Zhao, J., and Govindan, R. Understanding packet delivery performance in dense wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems* (New York, NY, USA, 2003), ACM Press, pp. 1–13.