EARTHQUAKES are devastating natural disasters. They are of particular concern to the people of the Los Angeles area as the San Andreas Fault and other faults lie nearby. ShakeOut, a scenario constructed by the United States Geological Survey, conservatively estimates that a 7.8 or higher magnitude earthquake in this area would result in a death toll of 2,000, with an additional 50,000 injured and over $200 billion in damage. In such an event, early warning would be a lifesaver. With as little as ten seconds of early warning, automated systems could take action to reduce damage and loss of life. Elevators could be stopped and the doors opened so that the passengers could step out. Data servers could suspend read/write operations to prevent corrupting millions of dollars worth of data. The electrical grid could be placed in a more stable configuration to reduce rolling blackouts as a result of downed wires.

Early warning systems are already in place in Japan, Mexico, Romania, Taiwan, and Turkey. In this paper, we describe a community-based sense and response system using a distributed network of sensors. A typical seismic network relies on a few high-quality sensors. Ensuring that the sensors are high-quality makes each sensor very expensive. In addition, the sensors must be installed by trained technicians at great expense. Our proposed network is a departure from earlier work through its emphasis on community participation: it relies on large numbers of volunteers throughout the community to build a distributed network by installing inexpensive sensors in their desktop computers or by using built-in sensors in their mobile phones or laptops. It also relies on community response to early warnings provided by the network.

Scientists at the University of California, Riverside are also working to distribute seismic sensors to volunteers. They have been able to detect some recent earthquakes with their Quake Catcher project. This is very promising news for distributed seismic networks.
**HOW THE SYSTEM WORKS**

Under our system, a volunteer will be able to use one of several different kinds of accelerometers to make his or her computer a client in our network. Each client will then log seismic data using the accelerometer. If significant shaking occurs, the client computer will "pick" that data and send a message to the server, alerting it to possible earthquake activity.

The server will receive a stream of picks from clients scattered throughout the Los Angeles area. It will then evaluate the incoming picks to determine if it is likely that an earthquake is currently occurring and, if so, where. The server will then immediately generate a ShakeMap, which will be very useful for evaluating damage and organizing relief efforts. An early version of this can be seen in (Figure 1). Ideally, an estimate of the size and location of the earthquake will be generated before the bulk of the shaking actually occurs, allowing the system to distribute that information in the form of an early warning.

The data-picks stream will inevitably contain a lot of noise. Since these sensors will not be underground or connected to bedrock, they will be subject to vibrations in their environment. It is reasonable to assume that users will often accidentally set off their sensors by bumping them or kicking the table on which the phone or laptop rests. It is therefore important to have a dense network so that noise will be damped by the information from the surrounding network.

Since this network needs to be very trustworthy, we have included an additional playback feature. If requested, the server can distribute prerecorded acceleration data to the clients. At a specified start time, the clients would play back the data to simulate an event across the entire network. This feature will allow us to effectively test the system, checking the ability of clients to provide accurate reporting of earthquake events as well as simulating historical events (such as the 1994 Northridge quake) to determine how our system would have reacted to an actual natural disaster. Successful testing will greatly increase public confidence in our system and encourage widespread participation.

**SYSTEM DESIGN**

**Conceptual**

This network of sensors will rely on Internet communications and volunteer support from the public. Each participating individual will use an existing accelerometer device inside his or her computer or mobile phone or purchase an accelerometer and store it in a buffer on the client’s computer. The data is processed with an exponential filter to reduce noise. The frequency of data is limited by the hardware used; for our example client shown in Figure 1 it is approximately 60 data points per second. The buffer will be read by a picking algorithm, which will trigger and send a message to the server if seismic activity is suspected. The picking algorithm is described later in the paper. In order to minimize inconvenience to participants, the whole application will run silently in the background and require minimal system resources.

To encourage use, participants will be able to watch the data stream gathered by his or her machine in real-time via a graphical user interface (Figure 2). In addition to these features, the client will be able to "play back" a simulated dataset. Once a day, the client will "call home" to alert the server to its presence. To assure participants of their privacy and security, the server will never be allowed to initiate contact with the client. After receiving the initial message, the server will alert the client to any playback requests. If the client receives a playback request, it will create a new thread which will activate at a specified start time. The new thread will then start reading data from the file given by the server, rather than the accelerometer. This will allow us to test the network from end to end as previously described.

**Client**

Once the client computer has downloaded and installed the necessary software, it will be part of our seismic network. Whenever the client computer is free, it will read data as often as possible from the accelerometer and store it in a buffer on the client’s computer. The data is processed with an exponential filter to reduce noise. The frequency of data is limited by the hardware used; for our example client shown in Figure 1 it is approximately 60 data points per second. The buffer will be read by a picking algorithm, which will trigger and send a message to the server if seismic activity is suspected. The picking algorithm is described later in the paper. In order to minimize inconvenience to participants, the whole application will run silently in the background and require minimal system resources.

To encourage use, participants will be able to watch the data stream gathered by his or her machine in real-time via a graphical user interface (Figure 2). In addition to these features, the client will be able to “play back” a simulated dataset. Once a day, the client will “call home” to alert the server to its presence. To assure participants of their privacy and security, the server will never be allowed to initiate contact with the client. After receiving the initial message, the server will alert the client to any playback requests. If the client receives a playback request, it will create a new thread which will activate at a specified start time. The new thread will then start reading data from the file given by the server, rather than the accelerometer. This will allow us to test the network from end to end as previously described.

**Server Architecture**

The server will collect all the pick data sent by the clients. It will then use existing Associator and Locator code to pinpoint the earthquake event. The Associator will associate a group of picks with a seismic event, which will then allow the Locator to locate the epicenter of that event. The server will use this data to output a ShakeMap, a heat map that uses different colors to provide an easily interpretable graphical display of shaking intensity (see Figure 1). This will aid in early warning and relief efforts.

**Picker**

Ideally, the algorithm will “pick” acceleration data only if seismic activity is present. In practice, such accuracy is not always possible; instead, the algorithm will “pick” whenever there is a sufficiently large change in the acceleration detected by the client. By properly calibrating the “picking” threshold, we attempt to maximize the probability of picking up true seismic events while minimizing the probability of false positives due to, for example, jostling of the sensor.

The picking algorithm depends on a comparison of short-term and long-term averages of recently recorded accelerations. The number of data points aggregated is variable, but currently the short-term average aggregates the most recent ten data points, while the long-term average aggregates the most recent 250 data points (for our example client, this is a little more than 4 seconds). If the ratio of the short-term to long-term averages exceeds a threshold value (for example, 10 percent above the long-term average), the client assumes that a seismic event is occurring and “picks” that data. The threshold value is variable to account for different sensor conditions for different clients — for example, some clients, such as mobile phones, may be more prone to random jostling than others, like laptops.

Once the picking algorithm has triggered, the software immediately saves the contents of the buffer to the hard drive on the assumption that the computer might soon lose power. This feature assures that if the accelerometer is attached to a desktop computer with no backup power source, recorded data will not be lost if an earthquake knocks out power, and will be available for later analysis once power is restored.

---

Fig. 1. Screenshot of the web interface showing simulated shaking detected by sensors around Caltech.

Fig. 2. The user is able to use the data gathered by his or her computer.
The software then continues recording data for a short time \( t_{\text{pause}} \) to determine as accurately as possible the magnitude of the shaking.

Finally, the software sends the “picked” data to the server for association. Some time later, the software assesses the buffer to the hard drive again to ensure that the entire earthquake was captured. Finally, the client waits some time \( t_{\text{pause}} \) before picking earthquakes and falsely reporting multiple earthquakes when in reality only one has occurred.

Both \( t_{\text{delay}} \) and \( t_{\text{pause}} \) are parameters that can be tuned both throughout the network and at the level of the individual client. If \( t_{\text{delay}} \) is too long, the shaking information will not reach the server in time and the server will not be able to give an effective early warning. On the other hand, if \( t_{\text{pause}} \) is too short, the client will underestimate the magnitude of the shaking, resulting in incorrect estimates of the earthquake’s magnitude. If \( t_{\text{pause}} \) is too short, the client will send several messages to the server for the same set of shaking, which will bog down the server and produce false reports of multiple back-to-back quakes. If \( t_{\text{delay}} \) is too long, the client might miss a second earthquake occurring soon after the first one, failing to report ACTUAL back-to-back quakes. The server will be able to calibrate these values on the client’s computer during the daily synchronization period to maximize the network’s accuracy and effectiveness.

**Associator**

The server receives a constant stream of data from the clients and the clients “pick” data. An Associator program on the server associates the incoming data with existing data to determine if it is likely that an earthquake has just occurred.

If such an earthquake is likely, the server estimates the likely epicenter and intensity of the earthquake and passes on this information to the client’s computer during the daily synchronization period to maximize the network’s accuracy and effectiveness.

For example, we expect \( t_{\text{delay}} \) to be about the duration of the first shock of an earthquake (approximately 1 second).

The aggregator was tested using synthetic pick data, with performance shown in (Figure 3). As can be seen in the figures, in this simulation we determined the origin of the earthquake to within 25 km after approximately 10 seconds. The synthetic pick data included some erroneous picks to make the simulation more realistic, which caused the early spike in the error. We are currently implementing this portion of the system on the Google App Engine. This gives us the advantage of using Google’s distributed server network. It is unlikely that a single earthquake would simultaneously take down all of Google’s servers, giving us a robust platform with which to host the associator software.

**Machine Learning**

Each client has many parameters associated with it: \( t_{\text{delay}} \), the number of data points in the short term average, the number of data points in the long term average, and parameters associated with the signal filter. In order to improve the network, we can tune these parameters on a client by client basis. For example, if a client is located near to a construction site with a jackhammer, that client’s data is not very useful for low levels of shaking, since those levels of shaking are indistinguishable from the shaking caused by the jackhammer. We can improve the reliability of the network by raising that client’s threshold to decrease the number of false positives. Once fully implemented, the server will be able to detect successive false positives and alter the parameters dynamically to maximize network accuracy.

We expect the optimal parameters generated by the machine learning algorithm to reflect underlying physical events. For example, we expect \( t_{\text{delay}} \) to be about the duration of the first shock of an earthquake (approximately 1 second).

Since it is vitally important to establish the reliability of our sensors, we compared a sensor’s performance against an existing high-quality sensor in the Southern California Seismic Network (SCSN).

Our sensor consisted of a Phidget brand accelerometer (approximately $10) connected to an iOS FC (approximately $300). The low cost of this setup (particularly considering the high cost of existing computer ownership by the community) is one of the strengths of our system; for an insignificant cost compared to current systems, a robust network of distributed sensors can be rolled out community-wide to detect potential earthquakes.

We set up our sensor in the basement of Millikan Library on Caltech’s campus, next to a conventional sensor designated MKB for its location in Millikan Basement (Figure 6). Both devices were placed in a small basement maintenance room (Figure 7). After simulating an earthquake by hitting the concrete floor with a sledgehammer, we compared the waveforms received by both devices (Figure 8).

While the data gathered by our example sensor had much more noise than the conventional sensor, both detected acceleration spikes at the same times. As is readily apparent, the signal-to-noise ratio of our sensor is quite poor (less than 14 dB, or a peak-to-noise amplitude of about 5-to-1). We expect that by implementing improved signal filters, we will substantially improve the signal-to-noise ratio and produce clearly defined acceleration spikes that will be readily detected by the "picking" algorithm.

Some of the noise in our sensor’s data is random noise. This is a necessary trade-off of using low-cost sensors. However, the use of a distributed network of many sensors will wipe out this noise. Because random noise is uncorrelated across sensors, and because the data from many sensors will be averaged in the Associator, random noise will be damped out and won’t interfere with earthquake detection.
FUTURE DIRECTIONS

Our work demonstrates the possibility of inexpensive and effective early-warning systems to reduce the costs of future earthquakes. By using a system of cheap sensors, we reduce the necessity of installing multi-million dollar high-quality sensor arrays to monitor seismic activity. By basing detection on a distributed community-based network of sensors, we ensure robust, reliable data collection and estimation of earthquake locations and intensities.

The next steps in implementing the Community Seismic Network will be encouraging widespread participation and establishing distribution channels for the information gathered by the system. We are confident that the low cost of the system (in fact zero cost to owners of cell phones or laptops with built-in accelerometers) combined with the inherent “coolness” of knowing that one’s computer is collecting seismic data will ensure adequate participation in the system. Our simulated playback feature will also demonstrate the network’s power and effectiveness, which combined with sufficient publicity of results should encourage even more members of the community to buy into the idea.

Distributing the information to early-responders will be more challenging, as many systems (e.g., elevators, electrical grids) will require something more than just refreshment of the data provided by the Community Seismic Network. Nonetheless, the huge net cost-savings of implementing an effective early warning system instead of merely waiting passively for an earthquake to happen should motivate implementation of such changes.

Scientists still can’t reliably predict earthquakes, but we DO have the capability to detect them in their early stages and take preventative action to limit the destruction they cause. The Community Seismic Network seeks to make this idea an affordable reality.

ACKNOWLEDGMENTS

We would like to thank Dr. Carol Readhead, Erin Chin, and Erik Madsen for proofreading this manuscript.

This portion of this project started with Dr. Rob Clayton and Dr. Mani Chandy in early 2009. Work started on the software in the Spring term of 2009. Daniel Obenshain, Rishi Chandy, Daniel Rosenberg and Annie Tang all worked on the project starting then, continuing over the summer as SURF students. Daniel Obenshain, Rishi Chandy, and Daniel Rosenberg have since continued work on this project in the 09-10 academic year.

FURTHER READING