Abstract—Geologists have long sought an early warning system for earthquakes since it is the best way to save lives and prevent property damage. Currently, the seismic network in the Los Angeles area is too sparse to provide any early warning. We propose a distributed system using MEMS sensors attached to volunteers’ computers in homes and schools across the Los Angeles area. This system will have the advantage of a faster response time and a denser network than the existing system.

Index Terms—distributed computing, seismic network.

I. INTRODUCTION

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[4].

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 (from conversations between Dr. K. Mani Chandy and Southern California Edison).

Early warning systems are already in place in Japan, Mexico, Romania, Taiwan, and Turkey[1]. At the University of California, Riverside scientists are also working to distribute seismic sensors to volunteers[2]. In this paper, we describe a community-based sense and response system using a distributed network of sensors. The network proposed is different from most of the earlier work in its emphasis on getting large numbers of volunteers from the community to buy and install inexpensive sensors or to use sensors in their mobile phones and laptops, and also to participate in responding to warnings.

In 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 then send a message to the server, alerting the server to possible earthquake activity.

The server will receive a stream of picks from clients scattered all over the Los Angeles area. The server will then evaluate the incoming picks to determine if it is likely than an earthquake is occurring and, if so, where it is. The server will then immediately generate a ShakeMap[6], which will be very useful in evaluating damage and organizing relief efforts. Optimally, an estimate of the size and location of the earthquake will be generated before the shaking actually occurs, allowing the system to distribute that information in the form of an early warning.

The data-picks stream will contain a lot of noise. Since these sensors will not be underground or connected to bedrock, they will be subject to the vibrations in their environments. It is reasonable to assume that users will often accidentally set off their sensors by bumping them or kicking the table. Because of this, it is important to have a dense network so that errors will be dampened by the information from the surrounding network.

Since validation of the system is very important, the network will have an additional playback feature. If requested, the server can distribute acceleration data to the clients to have them play back. Then, at a specified start time, the clients would play back the data to simulate an event across the entire network. This should greatly increase public confidence in the system.

II. SYSTEM DESIGN

A. Conceptual

This network of sensors will rely on Internet communications and volunteer support from the public. Each participating individual will either purchase an accelerometer device (MEMS or similar) and connect that device to his or her computer or use an existing accelerometer device inside the computer. The participant will then download our software from our server and execute the software to begin sending data. Once the data is received by the server, the server will be able to use that information for early warning.

B. Client

Once the client has downloaded and installed the software, the client will be part of our seismic network. Whenever the client’s computer is free, it will read data periodically from the accelerometer and store it in a ring buffer (a buffer where adding a new value deletes the oldest existing value if the buffer is full) on the client's computer. The ring buffer will be read by a picking algorithm, which will trigger if seismic activity is suspected and send a message to the server. In order to minimize the inconvenience to the client, the whole application should take less than one percent of the system's resources.
To encourage use, the client will be able to watch the data gathered by his or her machine in real-time via a graphical user interface (Figure 1).

In addition to these features, the client will have the ability to “play back” old data. Every day, the client will “call home” via a heartbeat message to alert the server to its presence. During this exchange, the client will send a message to the server notifying that the client is functioning. For security purposes, the server will never contact any client; only the client may initiate contact. After receiving the initial message, the server will update the client’s parameters, alert the user to an improved version of the software if one exists, ask for any logs saved by the client, and alert the client to any playback requests. If the client receives a playback request, it will create a new thread which will then wait until the 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.

C. 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. It will output a ShakeMap from this data to aid in early warning and relief efforts.

III. ALGORITHMS

A. Signal Processing

The acceleration data is gathered by either a MEMS device attached to the computer, or an internal accelerometer device. With proprietary drivers, that data is gathered by the client and stored in a ring buffer (a buffer where the oldest value is deleted to make room for new values if the buffer is full).

To normalize the values, an average is calculated over each axis of the accelerometer and data is reported relative to this average. Gravity is discounted, since the z-axis normal value will account for this, so the client will only deal with changes from this normal value.

B. Picker

Optimally, the algorithm will “pick” acceleration data if and only if seismic activity is present. More practically, the algorithm will “pick” whenever there is a detectable change in the acceleration detected by the client.

The picking algorithm depends on a short term average of recent data points and a long term average of recent data points. The number of points aggregated by average 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. Each represents the average absolute magnitude acceleration seen by the client over all three spatial dimensions, minus the normalization values to discount the steady state. If the short term average exceeds the long term average by a threshold value, the client assumes that seismic data is occurring and “picks” that data. The threshold value is variable, to account for different sensor conditions for different clients, but a ten percent threshold value would be reasonable.

Once the picking algorithm has triggered, the software will immediately save the contents of the ring buffer to the hard drive on the assumption that the computer might soon lose power. Then, the software waits some time $t_{\text{pause}}$ to determine the largest magnitude acceleration seen in that time. The software then packages that information in SAC format and sends it to the server via UDP, to avoid waiting for a response from the server. Then, some time later, the software saves the ring buffer to the hard drive again to ensure that the whole event was captured.

The client then waits some time $t_{\text{delay}}$ before picking again, to avoid picking multiple times during the coda of the earthquake.

Both $t_{\text{pause}}$ and $t_{\text{delay}}$ are parameters that can be tuned, both on the network level and at the level of the individual client. If $t_{\text{pause}}$ is too long, the shaking information will not reach the server in time and the server will not be able to give an early warning. On the other hand, if $t_{\text{pause}}$ is too short, the client will report an acceleration value that is too small, which will result in incorrect estimates of the earthquake’s magnitude. If $t_{\text{delay}}$ is too short, the client will send several messages to the server for the same set of shaking, which will bog down the server. If $t_{\text{delay}}$ is too long, the client might miss a second earthquake occurring soon after the first one. The server will be able to update these values on the client’s computer via the heartbeat process to make the network more effective.

C. Associator

The server will receive a constant stream of data from the clients and the clients “pick” data. An Associator program on the server will associate 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 will determine the likely source and intensity of the earthquake and pass on this information to interested parties. Ideally, this information could then be distributed before significant shaking had occurred.
IV. IMPLEMENTATION

The client program was written in Java to provide a platform independent program. An additional C library provided by Skyhook Wireless was included to locate the clients, using Skyhook Wireless software.

In the operation of the system, there are three types of messages exchanged between the client and the server. When the client is initialized, it will send a registration message to the server with its current location and be given a unique client ID in return. When the client undergoes significant shaking, it will send a pick message with a record of that shaking to the server. Periodically, the client will also “call home” to request any updates. The server will respond with a list of parameter changes, any code updates, any requests for saved data, and any playback requests.

If the server responds with playback requests, the client will save that data and wait until the specified start time. Then, it will start a new thread where it will read in the saved data as if from an accelerometer. This will allow us to test the system from end to end with either recorded or synthetic data.

V. EVALUATION

Since it is vitally important to establish the reliability of our sensors, we tested one of the sensors against an existing sensor in the Southern California Seismic Network (SCSN).

We set up our sensor (Figure 3) in the basement of Millikan Library on Caltech’s campus, next to a conventional sensor MIKB (Figure 4). Both devices were placed in a small basement maintenance room (Figure 5). After simulating an earthquake by hitting the concrete floor with a sledgehammer, we compared the waveforms received by both devices.

While the data gathered by the MEMS sensor had much more noise than the conventional sensor, both detected acceleration spikes at the same times. With more sensors, it will be possible to overcome the additional noise with a dense network of sensors.

VI. CONCLUSIONS

Here we have shown a plan to create a dense network of cheap sensors to detect earthquakes. We have shown that our sensors can detect seismic activity, though not as well as conventional sensors. We have outlined the pattern of network

Fig. 2. The accelerometer device, before it has been connected to a computer.

Fig. 3. An example client. The accelerometer device is seen here connected to the USB port on an example laptop.

Fig. 4. Sensor MIKB. This is one of the sensors in the Southern California Seismic Network, and is designated MIKB since its location is in the basement of Millikan Library on the Caltech campus.

Fig. 5. The arrangement of the two sensors. Both the example client and the MIKB sensor are visible. Also seen here is the sledgehammer used to simulate seismic activity for testing purposes.

Fig. 6. The impact of the sledgehammer, as detected by the two sensors. The signal detected by our sensor is much less clean.
traffic between the server and client and shown how we plan to implement a dense network of sensors.

VII. FURTHER WORK

In the future, we look forward to expanding this project to include mobile devices, such as cell phones and laptops with built-in accelerometers.

We also look forward to implementing this in the field, which will give us some real-life data.

ACKNOWLEDGMENT

Daniel Obenshain would like to thank Dr. Carol Readhead and others for proofreading this manuscript.

REFERENCES

Daniel Rosenberg is a junior at the California Institute of Technology.

Annie Tang is a sophomore at the California Institute of Technology.