A Mobile Crowdsourcing Approach Towards Earthquake Detection and Early Warning System

Zeyun Zhu
University of Oulu
zeyunzhu@gmail.com

Haejong Dong
University of Oulu
S2haejong@gmail.com

Perttu Pitkänen
University of Oulu
Perttu.pitkanen@student.oulu.fi

ABSTRACT

This paper proposes a distributed smartphone sensors network which enables intelligent real-time monitoring and detection of building collapses with near-accurate location information using the AWARE framework. During an earthquake, a dense network of smartphones will detect any free-fall events by utilizing their in-built accelerometer sensors and send corresponding timestamp information to a centralized server. The server determine whether an actual building collapse by monitoring the aforementioned fall-event timestamps within a period of the time for which the server monitors and waits for all events to be sent, and alerting the users if necessary. Laboratory and in-the-wild experiments are designed to evaluate the proposed system's feasibility and effectiveness when deployed in real life situations where users tend to have their smartphones on them.

KEYWORDS

Accelerometer, Android, AWARE, Collapse of Building, Earthquake, Fall Detection, Smartphone

1. INTRODUCTION

So far this century has witnessed quite a few enormous earthquakes leading to uncountable casualties. Reducing the death toll is the highest priority of most of the earthquake protection strategies. A few studies carried out in 20th century has recorded over 1100 fatal earthquakes causing an unimaginable total loss of life exceeding 1.53 million people. It has been pointed out that from 1900 to 1992, the main cause of death in earthquakes were collapse of building, causing over 75% of the deaths [1]. In January, 1995 the Kobe Earthquake also known as "The great Hanshin earthquake" hit a highly populated area of about two million while most of the people were asleep. Over five thousand people were killed mostly due to collapsing houses. The disaster was warned by Coburn that "The primary cause of deaths in the world is still collapse of buildings"[1],[2]. Further, in Athena earthquakes in September 7, 1999, there were 127 of 143 fatalities in the earthquake, mostly due to building collapses [3].

As a consequence, earthquakes and specifically the building collapses that happen during the earthquake are a serious threat to many people living in high earthquake risk areas. Collapsing of buildings is one of the primary causes of fatality in any earthquake. Although high-fidelity seismic sensors stations are deployed for early detection of earthquake tremors, little effort has been put to detect collapse of buildings in real time. The post-earthquake period of detecting collapsed buildings mostly involve image processing of aerial photographs. This method introduces a significant delay between the time of occurrence of the actual 'collapsing' event and the time it is detected, thus not being suitable for real time collapse monitoring.

Smartphone have lately become ubiquitous in today's society. They are equipped with MEMS sensors, such as accelerometer, gyroscope, light, pressure, etc. [4], powerful on-board microcontrollers, and capacious memory. Thus it is increasingly being viewed as handheld computers. Through the quality of these sensors is not at par with that of the hi-fidelity stationary sensors, their precision is increasing on-demand. With the plethora of apps and an open source platform for developing them, the functionalities of these handheld devices can be enhanced or modified to suit any specific need.

The most common information available immediately following a catastrophic event such as an earthquake is its magnitude and epicentre location. However, an accurate and a full extent of the damage assessment require time and are currently mostly provided by estimates. For instance, the full extent of the damage from the 1995 Kobe earthquake, Japan, was not recognized by the central government in Tokyo until many hours later. This greatly affected the rescue and recovery operations. Products such as 'ShakeMaps' [5] come close to the rescue by providing near real-time estimates using the sensors that are available from traditional seismic networks such as the Southern California Seismic Network (SCSN). A trade-off associated with these high-fidelity seismic sensors is its sparse distribution (approximately 10 km in case of SCSN [6]) resulting in maps of low-resolution. Increasing the density of such high-fidelity networks beyond that is needed for its basic function of locating the earthquake epicentre is cost prohibitive. The cost factor has been partially addressed by the use of open-network of low-cost microelectromechanical systems (MEMS) sensors that are hosted by volunteers [6]. However, the count of actual collapse of buildings and fatality is not obvious from these methods.

Another popular approach towards obtaining earthquake measurements data is the use of crowdsourcing. The "Did You Feel It" (DYFI) product of the US Geological Survey does this with a simple post-earthquake questionnaire [7]. The form of the questionnaire and the method for assignment of intensities are based on an algorithm developed by Dengler and Dewey for determining a "Community Decimal Intensity" [8]. Recent mild but widely-felt earthquakes in Los Angeles region have produced over 40,000 entries; supporting the likelihood of the popularity of such systems. However, this system has its caveats. One drawback of this form of sensing is that the human responders in the areas of heavy shaking usually do not make the data entry their first priority, and hence information from the most critical areas is usually late. Further, as this method of sensing requires users' attention during times of peril, they could be considered hazardous in nature.

In this paper, we focus on sensing the earthquakes and detecting collapsed buildings utilizing the smartphones as the distributed sensors. It introduces a system to cater to the needs of smartphone users with the help of AWARE framework [9].

2. SYSTEM DESCRIPTION

2.1 System Summary

We built a smartphone based open sensor network that is capable of detecting the collapse of buildings in near real time based on the algorithms deployed in the smartphone clients and server. It enable us to receive fall events at the server, detect building collapses, alert smartphone users of any collapsed buildings in their vicinity, and provide safety and first aid tips to users with the app; whilst respecting users' location privacy. Figure 1 shows the overview of our proposed system. There are two primary components of our system, Clients (Smartphone), and Server. This two tier architecture forms the entire Collapse Detection System (CDS).

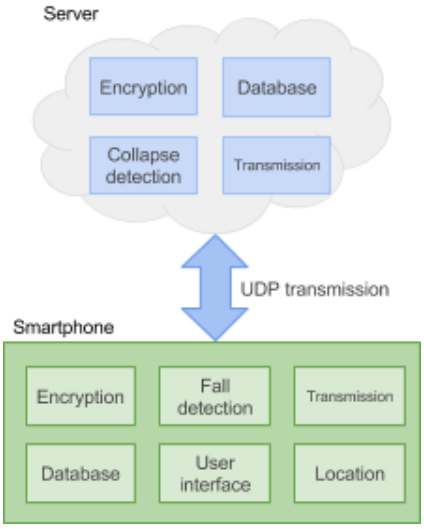

Figure 1. Overview of collapse detection system

2.2 Processing on the Client-side

The client side implementation was designed to meet the requirements that the system would have. Most importantly the client software's responsibility is the fall detection and sending the information to server.

The following algorithm is a simple way to find whether the phone is in a free fall. The client's fall detection algorithm works in following manner: The client software monitors the three axis of the accelerometer and calculates the Euclidian vector length of those three axis.

$$||a|| = \sqrt{x^2 + y^2 + z^2} \qquad (1)$$

If the vector sum falls below a predefined threshold the fall is detected. In theory this threshold should be zero in a free fall event. By default when the phone is stationary, it experiences acceleration of around 9.81m/s^2. By definition the Euclidian vector sum is positive in which every direction the phone is moving. For this reason the angle of the fall does not affect the fall detection.

The android API provides the options to adjust the update sensors' frequencies. This frequency has impact on the fall detection algorithm's accuracy as well as the battery consumption. The provided frequencies are: Fastest (0 microsecond delay), Game (20,000 microsecond delay), UI (60,000 microsecond delay) and Normal (200,000 microseconds delay) [4]. Both the threshold and the accelerometer frequency affect the algorithm's capability of detecting falls. This has been addressed during the evaluation process.

2.3 Processing on the Server-side

Server side of the system is required to identify the probability of a building collapse according to locations from where the clients (smartphones) have sent a fall event. The server based on that information eventually broadcasts to all subscribers (clients) a location of the assumed building collapse .As seen from Table 1,the duration for earthquake shaking depends on the magnitude[10].The time window for the server is focus on magnitude which is more than 7.3.

Table 1. A comparison of Magnitude and Duration for Earthquakes in the United States

Magnitude	Date	Location	Duration (seconds)
9.2	March 27,1964	Alaska	420
7.9	November 3,2002	Denali,AK	90
7.8	January 9,1857	Fort Tejon,CA	130
7.7	April 18,1906	San Francisco, CA	110
7.3	June 28, 1992	Landers,CA	24
7.3	August 17,1959	Lake Hebgen,MT	12
7.0	October 17,1989	Loma Prieta,CA	7
6.9	October 28,1983	Borah Peak,ID	9

Once server receives the first fall event it opens a time window (duration varies depending on the setting, for the purpose of this project we have decided it to be 1 minute. This setting was based on the background study) and the server starts gathering data from the subsequent fall events. The data is then put into a group and sent to another method to calculate distance between each component in the group by given GPS location. The implemented algorithm for the process is described in Figure 2. Assume that the server has received five fall events(A, B, C, D, E) within time window then the server assigns them as a set of data group for calculating distance of each component that is a process shown in Figure 2. Once the computation has completed each component in

the data group gets different probabilities of building collapse in which location the component describes.

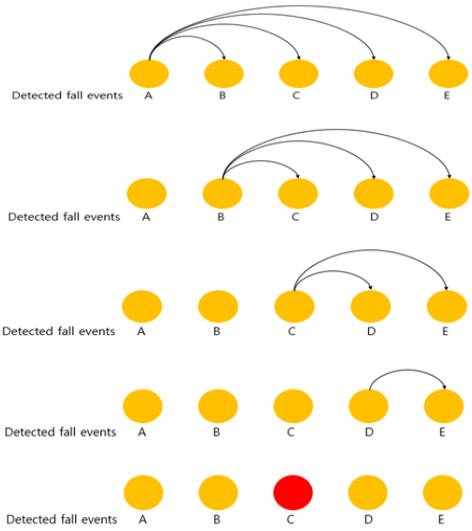

Figure 2. Fall building decision making process on the server

To get the distance between two geographical points with given GPS location data, we leverage "Ed William's aviation formulary" which calculates distance within a great circle track that is known to be a shortest.

$$D = \text{acos}(\sin \varphi_1 \cdot \sin \varphi_2 + \cos \varphi_1 \cdot \cos \varphi_2 \cdot \cos \Delta\lambda) \cdot R \quad (2)$$

When distance between two points reaches within 125 meters we assume that the location of two respective devices are likely placed in the same building e.g. Figure 3 (based on background study, we have agreed on the building base area is approximately 250(meter)x250(meter)). The fall events are sorted out by number of other events detected nearby. Highest count indicates a collapse of building in that location with highest probability within the data group.

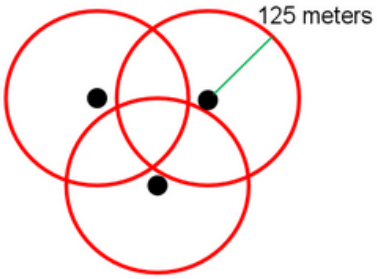

Figure 3. Algorithm that counts where more than two points are likely located closely

The results are broadcasted to all subscribers who have their devices online.

3. IMPLEMENTATION OF THE CLIENT

For the client the following features were implemented: Fall detection algorithm, data encryption, UDP-connection to the server and the user interface for displaying collapse data received from the server and some first aid and safety tips

3.1 Client Architecture

The client side software architecture can be divided to six major components: Fall detection, transmission, encryption user interface, location and database. The **fall detection** component has the actual implementation for the fall detection algorithm as discussed in the chapter 2.2 Processing of a Client. **Transmission** component has the all necessary functions for UDP data transfer with server including the implementation for heartbeat message that is sent once in a minute. The heartbeat sensor tells the server that the client device is still online and ready to exchange data with the server. **Encryption** component provides the 128 bit AES encryption and decryption so that all the sent data and the received collapse data are encrypted in the database. All the features the user will see are implemented in the **user interface** component. It includes Google Maps screen displaying the collapse locations received from the server, first aid and safety information screens, and a debug screen for adding sample collapse data in the phones local database for testing purposes. The **location** component is used to acquire the phone's GPS coordinates easily with AWARE location sensor so that it could be sent to the server. **Database** is used to store the received collapse data and the map uses this database to display the collapses. The received data is at encrypted form in the database.

Android programming was done with Java programming language and the chosen IDE was Android Studio. The most important libraries and interfaces that were used for implementing the client side software were Java's default libraries e.g. Cipher for AES encryption and decryption, and DatagramPacket for UDP transmission. The AWARE framework included easy to use interfaces for ESM questionnaires and GPS location data. The Android API provided all necessary interfaces to access the accelerometer data and to create user interfaces. The map was implemented using Google Maps Java API.

3.2 Assumptions and Constraints

The smartphone plugins functionality relies on some assumptions. First assumption is that the smartphone users will leave their phones stationary on a table or other surface most of the time. Second assumption that had to be made was to use constant threshold for the fall detection algorithm. During the evaluation process some findings were made to find out what kind of values would be best for the threshold. Thirdly a strong assumption was made that the will actually enter a free fall during the building collapse. The phone may encounter some obstacles and the trajectory can be quite unpredictable in real life situation because of the nature of the earthquake and the randomness of the building collapse.

The floating point calculations used in the fall detection algorithm are calculation-heavy operations. This affects strongly on the smartphone's battery consumption. Some adjustments can be made to the sensor frequency for saving battery life.

4. PERFORMANCE EVALUATION OF THE CLIENT

For gaining a better understanding whether the implemented system would work in real life situations, series of test were made. During the client side evaluation, following aspects were tested: Accelerometer range during a fall, fall detection accuracy, the

plugin's battery consumption and the suitable accelerometer delay value.

4.1 Accelerometer Range Test

For pages other than the first page, start at the top of the page, and continue in double-column format. The two columns on the last page should be as close to equal length as possible.

4.1.1 Setup

During the test one smartphone was used to make several drops. For each drop the accelerometer values were monitored in the Android Studio IDE. By analysing these values the appropriate range can be found.

4.1.2 Results

Accelerometer threshold scale was easily found using planned method. During the test majority of the drops repeated values in the range of 0.2 to 0.6.

4.2 Accelerometer Delay Test

To find the appropriate accelerometer delay value, each accelerometer mode was tested by group of participants for approximately one week per mode. By testing each mode several days we gathered lots of valuable data on how many times phone falls gets triggered during normal activities and which kind of activities would trigger the fall. During this test phase we gathered some feedback on the user's opinions about the battery consumption.

4.2.1 Setup

For this test, a group of 7 people were recruited and they were given the plugin software. The users were told to install and launch the plugin for it to monitor fall events. The test started with the fastest accelerometer delay option. The mode was changed approximately once in a week until all four modes were tested. All fall data were sent to the server and saved in a database for further analyzing.

4.2.2 Results

When 7 people carried their phones during their normal activities, the average of fall event number is around 3 per day. This data may also be affected by the activity of the test users, which cannot be predicted. Although this data gave some good idea of how many accidental fall events happen when there are no earthquakes. The users also provided some feedback about the reasons why the fall was triggered. Most of the false positives occurred while the phone was placed on a surface or whilst riding a bicycle. Majority of the test subjects disliked the fastest accelerometer mode strongly for its large battery consumption.

4.3 Fall Detection Algorithm's Accuracy Test

Fall detection algorithms accuracy was tested with a series of phone drops on a soft surface. By this method we would able define how the amount of detected falls depends on the different thresholds and accelerometer delays. During these tests three different model phones would be dropped several times for each delay-threshold combination. We also included two different heights for the test: 15 cm and 70 cm.

4.3.1 Setup

Accuracy was tested with the three differing phone models. The height, accelerometer delay, and threshold were changed between tests. Used thresholds ranged from 0.1 to 0.6. This added up to 48 different test cases and for each case the phones were dropped 40 times. The phones were dropped on a soft surface so the phone wouldn't bounce on impact. The drops happened from approximately 15 cm and 70 cm heights. Phone models used were: OnePlus One, Galaxy Nexus and Galaxy S4 mini. During these tests the data was saved manually into a table for further analyzing.

4.3.2 Results

The data answered well to the initial questions whether phone model, fall height, accelerometer mode or threshold affected the results. Generally all the changed variables have their effect on the outcome. For smaller fall heights such as 15 cm the accelerometer delay has a big impact on the outcome. For this height the fastest and game modes seem to be the only ones that have reliable results with all tested phones. Whereas with lower height falls are highly dependent of the accelerometer frequency, the higher fall events were predictably easier to detect. With higher falls the phone has more time to update the sensor values and thus has higher probability of fall detection. Like in the case of 15 cm fall, the lower thresholds have worse fall detection capability. In Figure 4 and Figure 5, the differences in the amount of detected falls with the fastest (Fastest mode. 0 microsecond delay) and the slowest (Normal mode. 200,000 microseconds delay) modes can be seen. As can be seen the majority of drops are detected in both cases and with both accelerometer delays, with threshold 0.4 m/s^2. The higher drops are also more easily detected with slower frequencies.

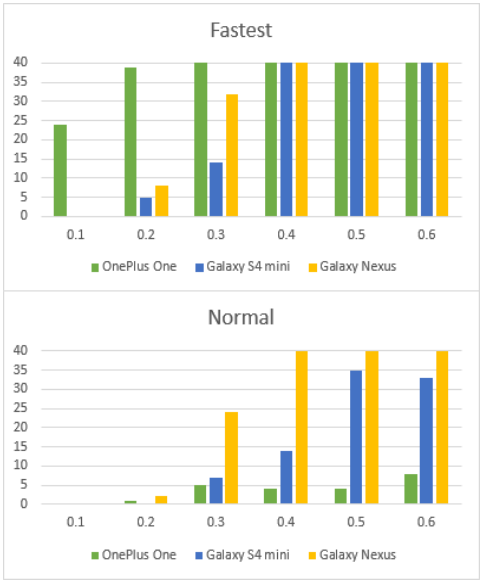

Figure 4. Comparison of the fastest and slowest modes fall detection capability with 15 cm height drops

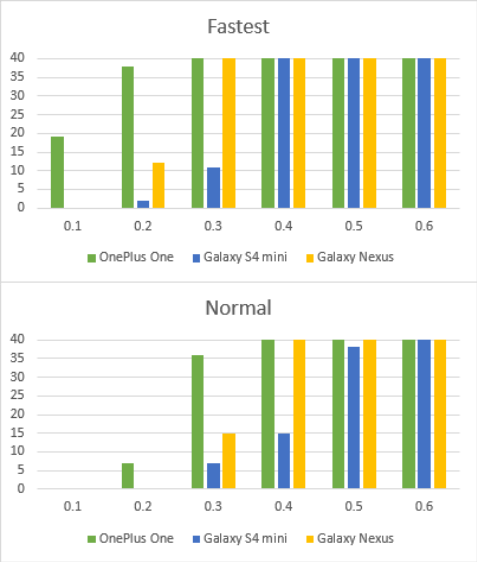

Figure 5. Comparison of the fastest and slowest modes fall detection capability with 70 cm height drops

4.4 Battery Consumption Test

For the system's user friendliness, the battery consumption needed to be tested. The hypothesis for this test was that the faster accelerometer modes will use more battery than the slower ones.

4.4.1 Setup

The test was completed with one smartphone phone using all the different accelerometer modes for fixed length of time. The used smartphone's model was Galaxy Nexus. Each accelerometer mode was tested for 3 hours and the percentage of the battery was monitored before and after the test.

4.4.2 Results

The battery consumption test result did not match the expected results. The gathered data seems quite the opposite of the expected results: the slower accelerometer delay seems to consume more battery than the faster ones. This may be a result of an unstable battery or some other processes using the battery. All in all more tests would be needed to be sure of the results. From the user tests some feedback was given about high battery consumption with faster accelerometer modes, which also contradicts the results we got from the tests. For better results more tests with several different phones would need to be completed, which couldn't be done during this phase because of the lack of time.

5. IMPLEMENTATION OF THE SERVER

Implementation on the server side has done over phases, first realizing requirements, second implementing functionality, third testing, and last refining.

The features that have been implemented on the server side are: UDP-connection to the client, time window generator, data security, message broadcast, database management, GPS distance calculator, identification of a location of collapsed building, network information management of the client.

5.1 Server Architecture

Main components of the server's functionality are database management, grid management, security, network information management, time window management and socket management.

Core computations that yield result data is fulfilled by grid management. It calculates GPS distance and identifies in which location has the highest probability of building collapsed among the data group. Employed algorithm needs to be refined for better performance. However this will be discussed more about in the server evaluation.

Database related methods handle data storage and retrieval of the database. Once data arrives from client server first stores it and then passes to other methods to continue process.

All the data streaming in/out to/from server are secured by data encryption/decryption. A client and server shares public and private keys so that private key is secured by encryption using public key when initial connection between a client and server has made.

The server also maintains network information of subscriber that is sent by every predefined time of duration (we have chosen every 1 minute in our system). These information are retrieved and enable broadcasting messages to the subscribers when the server has completed the computation of selecting an assumed location of collapsed building.

Communication with clients is handled by socket via UDP protocol. There are two different types of messages sent by client in our system that are fall event message and network up to date message.

Time window method manages opening and closing of time window. The opening time window takes place when initial fall event message arrives to the server.

5.2 Assumptions and Constraints

There are a number of assumptions that underlie the server system for which bringing the result data. First it assumes that every subscriber sends a fall event message as it falls without latency, second GPS location is correct regardless of indoor or outdoor, third a location with highest number of clients placed nearby is regarded to be the location where building collapsed.

The algorithm that server has chosen to use for getting the result data has limitation in capacity of a number of clients which whether the server can take all those clients into process whilst it can complete the entire computation within a min. If the system tries to open another time window whilst one exist that requires better flexibility of the server design in thread managing. Nevertheless the issue remains unsolved even by dynamic system design as with current algorithm it takes over minutes when the number of clients increases by thousands which is commonly happening when earthquake occurs. The computing time must be completed within a minute so that people around the area can actually benefit from alarm message sent by the system.

Another constraint will be the fact when building is monitored by a single client, there is no clear way to distinguish the building collapse from general isolate events at the moment.

6. PERFORMANCE EVALUATION OF THE SERVER

In evaluation phase, the server required to be put into a test in which environment consists of thousands of clients prepared to cooperate with the server. However that was rather unrealistic to set, thus we have come to write a module which can generate random values in the same format as the server receives from client. The result was as expected disappointing, computing time has increased dramatically as more clients attempt to be involved into the system.

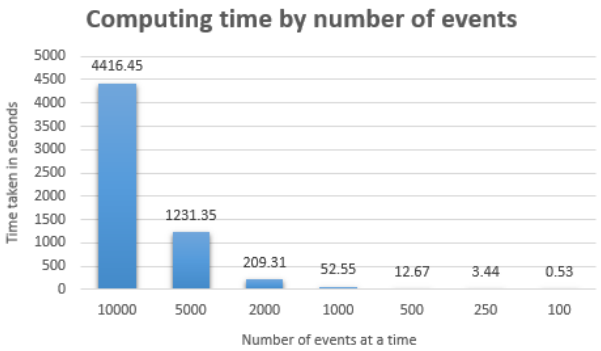

Figure 6. Overview of computing time within the server code

Given circumstance where the system is bounded by time critical issue, the current server has proven itself by the evaluation that it cannot hold more than a thousand fall events at a time which makes it first thing to improve in the future work. Unfortunately we couldn't come up with a novel mean of how to refine this issue however we have discussed about a few very high level conceptual algorithms that would possibly reduce the computing time effectively and significantly, which for instance if we could somehow remove relatively isolated events from the data set to be calculated, and take into account only those events with certain level of probability which has been yielded by calculation based on overall data or even based on big data for instance assigning different priorities depending on frequency of earthquake event history in each region instead of putting all data into the computing.

7. CONCLUSIONS AND FUTURE WORK

We studied the ways of using our smartphone plugin with different accelerometer delay modes coupled with different fall detection thresholds. Meanwhile, an active approach was used to test the accuracy of the plugin and the server's capabilities on handling multiple simultaneous clients. Further, we received user feedback via questionnaires.

The client test results gave useful insight whether the plugin would work in real life situation and which values should be chosen for fall detection threshold and which accelerometer delay modes would work best. In general the fastest accelerometer mode works best also with lower threshold values yet it is disliked by the users for it drains the battery quite fast. The deployed plugin could have pre-defined threshold for each accelerometer delay modes and these pairs could be defined as a result of a larger tests with different phone models. As part of the future work, larger and more statistically reliable tests should be made.

In the future, the system can be implemented further as followed.

- Provide navigation details to the scene.
- Make functionalities to report live events from the scene.
- A web based user interface for scientists and researchers to access the accelerometer data of the smartphones subscribed to the system.
- Match mode with corresponding thresholds for users.
- Find the times when users are more active to reduce the false events.
- Use fastest mode when phone is charging.
- Support more devices. Extension of the system on smart watch is promising.
- Multiple server centres over targeted area or a country to minimize data travel time, stable, high speed and reliable network infrastructure.
- Improve security level in the future.

8. ACKNOWLEDGMENTS

We would like to express our gratitude towards Professor Vassilis Kostakos for his expert guidance and Mr. Aku Visuri for his support throughout the lifecycle of the project. We also take this opportunity to extend our sincere thanks to Professor Timo Ojala for providing us with a concrete outline for the documents. We would also like to thank Mr.Pratyush Pandab for his support and guidance during the project lifecycle.

9. REFERENCES

[1] Cobum,A.W., Spence,R. J. S., & Pomonis, A.(1992). Factors determining human casualty levels in earthquakes: mortality prediction in building collapse. In *Proceedings of the tenth world* conference on earthquake engineering (VOL. 10, pp. 5989-5994).

[2] K. Shiono, F. Krimgold, and Y. Ohta. A method for the estimation of earthquake fatalities and its applicability to the global macro-zonation of human casualty risk. *Proc. Fourth Int. Conf. Seism.* ..., 1991.

[3] Robin Spence,Emily So.Estimating shaking-induced casualties and building damage for global earthquake events.

[4] "Sensors Overview | Android Developers," 2015. [Online]. Available: http://developer.android.com/guide/topics/sensors/sensors_overview.html.

[5] D. Wald and V. Quitoriano, TriNet 'ShakeMaps': Rapid generation of peak ground motion and intensity maps for earthquakes in southern California, *Earthq.* ..., 1999.

[6] R. Clayton and T. Heaton, Community seismic network, *Ann.* ..., 2012.

[7] D. Wald and V. Quitoriano, Utilization of the Internet for rapid community intensity maps, *Seismol.* ..., 1999.

[8] About Community Internet Intensity Maps, Oct. 2000.

[9] AWARE | Android Mobile Context Instrumentation Framework, 2015. [Online]. Available: http://www.awareframework.com/ .

[10] Southern California Earthquake Center(SCEC).Available: http://www.scec.org/.

SUMMARY

Name	student ID	Design	Implementation	Evaluation	Final report	Total
Zeyun	2382881	65 26%	127 40%	73 44%		
Haejong	2292191	68 27%	92 29%	47 29%		
Perttu	2261724	53 21%	98 31%	44.5 27%		