MIS Project Documentation

Jan Frederick Eick, Benjamin Burse

August 17, 2017

1 Introduction

In their paper "MyShake: A smartphone seismic network for earthquake early warning and beyond" the authors are discribing how many countries in the world are still suffering from insufficient to no earthquake prewarning at all. The consequences are thousands of people still dying in earthquakes every year. There approach to the problem was the validation and implementation of a low cost, highly available prewarning system based on modern smartphones and mobile wireless communication. The idea was to develop a smartphone application which makes use of the already built-in accelerometer of smart devices, to detect earthquakes. In case of a positive detection, a short message with all relevant data is sent over to a server. The servers task is to collect all data sent from the mobile devices and decide, based on the number of occurrences, if a real earthquake is on the way. After an earthquake is being detected, the server notifies all users in the region.

2 Implementation

2.1 Original implementation

The authors of MyShake designed the application in a way that it constantly records accelerometer sensor data using a background service. These sensor readings are stored in a circular buffer which holds 250 sensor samples in the default settings (25 Hertz for 10 seconds). With each new updated sensor reading three features of the signal are calculated:

1. The interquartile range
2. The maximum zero-crossing rate
3. The cumulative absolute velocity of the acceleration vector sum

These features are used as inputs to a trained Artificial Neural Network which classifies the sensor data. If the output reaches a threshold of 0.9, the state of the application is changed to "streaming" and the application starts streaming measurement data to the server.

The server can then decide if it notifies nearby devices to start streaming as well or if an earth quake warning should be issued.

The authors of the paper trained their neural network using a shake table and a wide variety of mobile devices. They also used sensor readings of real earth quakes from seismic research centers which have been filtered and downsampled to resemble sensor readings from mobile phones.

The server is able to further classify different sensor readings as earth quakes and can calculate the approximate magnitude and location of the seismic event.

2.2 Mobile Application

The heart of the application is a reliable classifier for the sensor data. Our initial approach was to rebuild the ANN using training data from our own phones and manually inducing shocks to a table and shaking it. We experimented with TensorFlow and NeuPy to train a model with rougly 80 sensor readings of different sampling rates. After testing our classifier on new data it demonstrated subpar performance for our application.

We experimented with a different approach of performing a fast fourier transformation on the signal data and tried to find a threshold of high frequencies for which we would detect a potential seismic event. Unfortunately this approach would need a much bigger sampling rate of the accelerometer sensor, which we deliberately tried to avoid to not put too much strain on the battery. With our desired sampling rate of 25 Hz we were unable to identify high frequencies of shakes which are characteristic for seismic events.

After several failed attempts to recreate a reliable local classifier we decided to try to reverse engineer the MyShake APK using JADX - Dex to Java Decompiler. The Java code was not obsfuscated and we quickly identified the class `com.dt.myshake.algorithms.ann.ANNImplV1` as the ANN used by the original authors. The class contained hardcoded values for the feature scales, intercepts for the first layer of the network, the intercept for the first hidden layer and weights. With very little adjustments we were able to use the code of the original authors. Further investigation of the reversed source code revealed that the application is using a very interesting mechanism to update this ANN without relying on a complete update of the APK. In the class `com.dt.myshake.service.ann.ANNLoader` the method `downloadAndStoreNewClass(Context context)` is used to download a new JAR-File from the server of the authors. The file is stored in the system data directory and later with the method `loadNewClass(Context context)` converted to a dex.

```
private boolean loadNewClass(Context context) {
    ...
    try {
        File f = context.getDir("/outdex", 0);
        ...
        String receivedJarPath = "ANNImplV" + this.latestANNVersion;
        this.loadedDetectionAlgorithmClass =
                      new DexClassLoader(f.getAbsolutePath()
                        + "/" + receivedJarPath, f.getAbsolutePath(),
                        null, getClass().getClassLoader())
                          .loadClass("com.dt.service.ann.algorithms."
                             + receivedJarPath);
        return true;
    } catch (Exception e) {
        ...
    }
}
```

We decided to intercept the traffic of the mobile app using Portswigger Burp Suite and tried to find out if the application is updating its ANN. During a two day observation phase we could not detect any activity of the application to update the ANN.

Our implementation used the ANN output and additionally calculated the peak ground acceleration to detect local seismic events. For demonstration purpose we chose very low thresholds of output certainty and used a low threshold for the PGA to trigger network events to the backend. As soon as the thresholds are reached the state of the application is changed to the `StreamingState`. In our demonstration we did not actually stream sensor data to the backend.

In case of a received earth quake warning, we display an alert by using a notification with the highest possible priority and play a warning tone.

2.3 Backend

Our implementation of the backend was done using Python 2.7. We chose an XMLRPC interface for the task at hand using the `SimpleXMLRPCServer` module of Python's standard library. We used a very simple algorithm to determine the trigger of an earth quake warning by setting a low threshold of the number of devices that reported a seismic event in a timeframe of 10 seconds. We notify the mobile clients using the `Firebase Cloud Messaging` service from google. The backend can decide to notify clients in proximity of another seismic event to switch to the `StreamingState`, or it can issue an actual earth quake warning.

2.4 Libraries

On the client we used the following libraries:

- androidplot for the plotting of the sensor data
- Google Support Library (v7 appcompat, constraint-layout, design)
- Firebase Cloud Messaging to receive push notifications from the backend
- aXMLRPC for the communication with the XMLRPC backend server

On the backend we used:

- SimpleXMLRPCServer
- PyFCM to send push notifications using the FCM service

3 Results

3.1 What we achieved

With our implementation we were able to set up a working front- and backend. As already described, the frontend collects the earthquake data and the backend does the registration of unique devices, evaluation and notification. We were able to detect a possible earthquakes on a device and process the received information on the server side. Afterwards we were also able to notify all registered devices of a possible earthquake via googles FCM API (Firebase Cloud Messaging, successor of GCM - Google Cloud Messaging).

3.2 What we did not achieve

As also already explained, the original implementation of the application used an ANN classifier (Artifical Neuronal Network) to separate real earthquakes from other events like hitting the table. We were not able to construct an ANN on our own, because of the lack of access and understanding of real earthquake data. We first tried to use a kind of activity detection as a classifier, with very bad results, but then switched over to the reverse engineered ANN classifier of the authors.

In difference to the original backend, which was able to calculate the center and spread direction of the earthquake, our backend is limited to a very basic data analysis (via threshold) and notification functionality. This could be a possible point for further development.