Signal Finder Design Documentation

Team Donut

Sadaf Amouzegar
Alejandro Carbonara
Angela Gong
Kalpana Suraesh
Jessica Yu

1 Introduction

Everyone has faced the common problem of not having cell-phone service when they need it. When a person is inside a building that has poor reception, it would be very convenient to be able to find the place where his or her reception would be best in a short amount of time. With this in mind, our project aims to solve this problem.

1.1 Goal

Our goal is to create an application that can conveniently inform the user where their cell phone reception. In order to do so, we will collect signal data via distributed methods from various users and phone carriers using an Android application. Once we have gathered enough data points, we will be able to see where reception is best by filtering and computing upon our data set. We will proceed to use that information to our advantage, and if we are in need of better cell phone signal, we would be able to use the Android application and a web UI to find the optimal spot to use our cell phone.

2 Data Design

2.1 Data Format

Data points will be stored in the phone memory and on a database in the following format.

Parameter	**Storage Format**	**Description**
`clientId`	`String/VARCHAR(40)`	An opaque, hashed string of the phone's IMEI. This is constant for all requests from a single device.
`carrier`	`String/VARCHAR(40)`	The phone carrier of the client (e.g. “ATT”)

Parameter	Storage Format	Description
latitude	double/NUMERIC(7, 4)	The latitude, in degrees.
longitude	double/NUMERIC(7, 4)	The longitude, in degrees.
accuracy	double/NUMERIC(7, 4)	Accuracy of the measurement in meters.
phoneType	int/INTEGER	The phone network type, which is one of [0, 1, 2, 3]. (e.g. CDMA, GSM)
time	java.sql.Timestamp/ TIMESTAMP	The timestamp at which the measurement was taken.
signal	int/INTEGER	Signal strength at the location in dBm.

Table 1. Data storage format

2.2 Data Storage on Android App

Within the Android app, data points are stored in a text file called `data.txt`. After data has been uploaded to the server, it will be deleted from the phone to save memory. Current data is stored in the cache and is written to memory as soon as it is processed by the phone. Up to 500 data points will be stored on the phone at once, and if more data points are inserted, then those 500 points will be written to the database before new points are recorded.

2.3 Data Storage on Database

2.3.1 MySQL Database

For our design we will use a MySQL database to store the data points. The benefits of using a MySQL database is the ability to use primary keys to uniquify data points, as well as foreign keys to limit the kind of data inserted into the database. MySQL also allows for different ways to filter the data which will be beneficial for both the mobile and web interfaces.

The data will be stored on the server at mysql.super-magician.com at `cs3donut`.

2.3.2 Data Storage

Data is stored in two tables on the database. `data` contains all of the data points stored as tuples. Each tuple contains (`clientId`, `carrier`, `latitude`, `longitude`, `accuracy`, `phoneType`, `time`, `signal`). Its primary key is (`clientId`, `latitude`, `longitude`). This makes it such that each user can only submit one signal per location, since we do not want to overflow the database with unnecessary data. It contains a foreign key to the table `valid_clients`.

`valid_clients` contains tuples with all of the (`clientId`).

3 Architecture Design

Data is processed in the following format, as can be seen by Figure 1. First collection of signal information and GPS location is done by the mobile app. It is then uploaded to the server and inserted into the database. Finally, the results are retrieved from the database, processed, and displayed in a heat map interface.

Figure 1. The data collection process.

3.1 Data Collection

Collection of signal strengths is done via an extension of the `PhoneStateListener` and an implementation of the `LocationListener`. Using the listeners, the phone is able to retrieve information about the current signal strength as well as its GPS location. When the user's location changes (via a prompt to the `onLocationChanged` method) or the signal changes (via a prompt to the `onSignalStrengthChanged` method), then the app will collect the new data. In the event that the GPS is disabled (a call to `onProviderDisabled`), then no data will be collected until the GPS is activated (a call to `onProviderEnabled`).

At each instance a `Location` and signal strength (an `int`) is collected, which will be further processed by the mobile app to be uploaded.

3.2 Data Transfer

3.2.1 Client Registration

Prior to uploading data to the server, the client must register their hashed clientId with the database. This is done via HTTP POST request to the server at http://www.anjoola.com/donut/register.php.

3.2.2 Transfer to Database

Upon accumulation of data, the data points in the file `data.txt` will be uploaded to the database. First the app calculates how many data points are stored in the file and sets a variable `numData` to that number.

Data is submitted via HTTP POST request to the server at http://www.anjoola.com/donut/submit.php. The HTTP parameters will be formatted as strings, the following are first submitted: (`clientId, carrier, numData`) using an array of `NameValuePairs`. This will indicate to the server the source of the data and the number of data points to expect.

For each data point to transfer, the following parameters are sent: (`latitude$i, longitude$i, accuracy$i, phoneType$i, time$i, signal$i`), where the `$i` indicates the index of the data point (ranges from 0 to `numData - 1`).

3.2.3 Server Response

Depending on the type of input it receives, the server will respond with either an "HTTP/1.1 200 OK" header, indicating that the upload of data was a success, or a "HTTP/1.1 400 Bad request" header, indicating that the upload was a failure. The following will also be outputted:

```
SignalFinderAPI=1.0
<errorCode>
<message>
```

Error Code	**Message**
0	OK, no error
1	Bad clientId token
2	Timestamp out of bounds
3	Parse error
4	Malformed parameters (missing record or fields)
5	Other error

Table 2. Possible error codes and messages.

A 200 response means that the request was recorded successfully. A 400 error code implies that an error prevented the request from being handled. After this error code, the user can try to resubmit data again.

Errors can occur if some field is not specified, fields are out of bounds (e.g. the latitude entered is -300, which is impossible), invalid parameters are entered (e.g. "foo" for the signal strength instead of an integer), or the `clientId` is not in the list of `valid_clients`.

A list of the parameters and their ranges can be seen on the next page in Table 3.

Parameter	Valid Range	Parameter	Valid Range
`latitude`	-90 to 90	`phoneType`	0, 1, 2, 3
`longitude`	-180 to 180	`time`	A valid UNIX timestamp
`accuracy`	Greater than 0	`signal`	-113 to -51

Table 3. Parameters and valid ranges.

3.2.4 Privacy Concerns

There may be concerns involving using the phone's IMEI as a way to identify users. This problem is resolved in the following process. At the first installation of the app, the app gets the phone's IMEI, prepends it with a random salt, and converts it to base 64. This is then stored in a file called phone_id.txt, and the real IMEI is never stored in any way. This new clientId is then uploaded to the server.

During data transfers, this hashed phone ID (clientId) is sent to the server along with data points, who then checks it against their list of valid clientIds to see if this user is allowed to upload data. If so, then this user's data is inserted into the database. Otherwise, the data is rejected.

This is a secure form of data uploads because the client's real IMEI is never transferred in any form, and invalid users are not allowed to enter any data.

4 User Interface

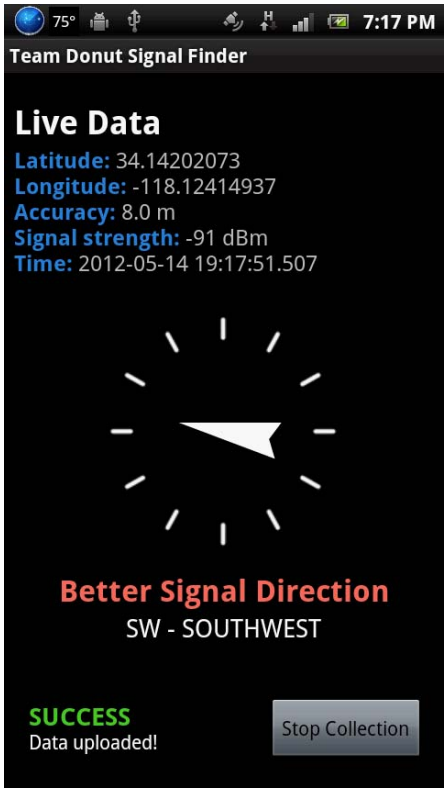
4.1 Mobile UI

The mobile interface consists of two parts: one to show the user the current collected data, and one to show the direction of better signal strength.

4.1.1 Data Collection

Upon the start of the application the user is able to choose a sleep interval and press the Start Collection button, as can be seen by Figure 2. An Ongoing notification will appear in the Android status bar indicating that the running service has begun (see Figure 4). This signal collection service then runs in the background, and the user may run other processes in the meantime.

As the `SignalStrengthListener` listens for new information about the signal strength and GPS location, various components in the UI are updated accordingly. A `TextView` displaying the live data is updated with the newest information, and another `TextView` displays the current database status and whether or not the previous data points were successfully uploaded to the server. This can be seen in Figure 3.



Figures 2 and 3. Mobile app upon startup, and collecting data.

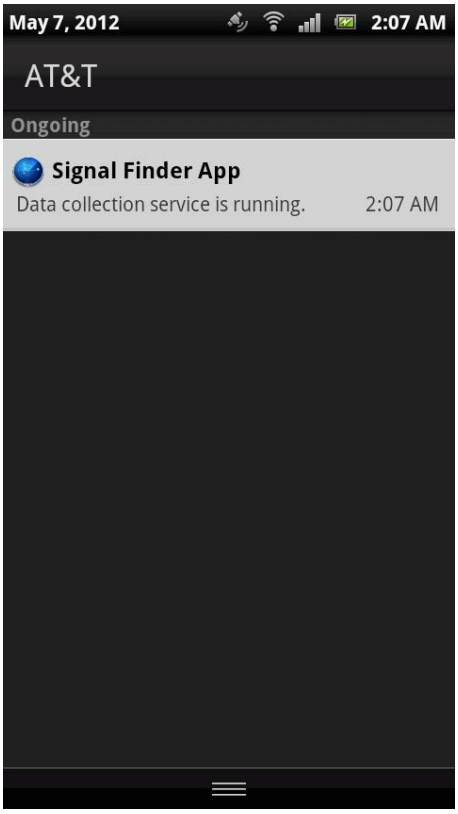


Figure 4. Ongoing app notification.

4.1.2 Better Signal Finder

While collecting data, the app will also display the direction where a better signal can be found. This is based on previously collected data in the database. The mobile app queries the database for signals and locations in an area 22m by 22m around the user, and finds the location with the best signal strength within that area. Using some calculations, the app will point to the direction of better signal and indicate this to the user.

The latitude and longitude of the area around the user is calculated as (`current_latitude ± 0.0001, current_longitude ± 0.0001`). An example of this display can be seen in Figure 3.

4.1.3 Battery Considerations

Collecting data and querying the server for better signal strength can cause a drain on the battery of the phone. In order to remedy this, users are able to, from a drop-down menu (Figure 5), choose how often they want to collect signal data.

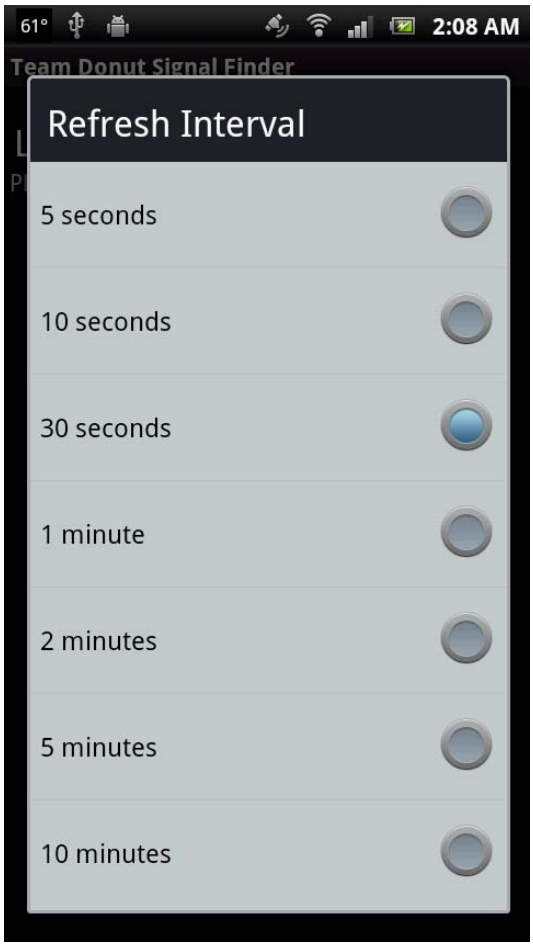


Figure 5. Drop-down menu to select interval between data collection.

In between data collection periods, the app will go into a low-power mode in which it does nothing until a timer interrupt causes it to wake up and begin collecting data.

The calculations for the better signal is done only every minute, since it is unlikely that the user will move far or fast enough such that the signal strength can change significantly within a minute.

The user may elect to stop all data collection and better signal calculations by pressing the Stop Collection button.

4.2 Website Frontend

4.2.1 Raw Data Dump

The website queries the MySQL database, pulls the results, and outputs it into a table. As data collection occurs, data is constantly being uploaded onto the database. Data points will therefore appear live on the data table. Columns that will be displayed include the time, carrier, latitude, longitude, and signal strength.

The user may also choose to filter the data by a minimum or maximum latitude and/or longitude, and can sort the table in order of carrier, location, or signal strength.

4.2.2 Heat Map

Data points will be displayed in a heat map format using the Google Maps API. Depending on the strength of the signal, various locations on the map will be colored. The user is then able to click on these shaded points to view more detailed information on the average signal strength at that point and the coordinates at which the data was taken from.

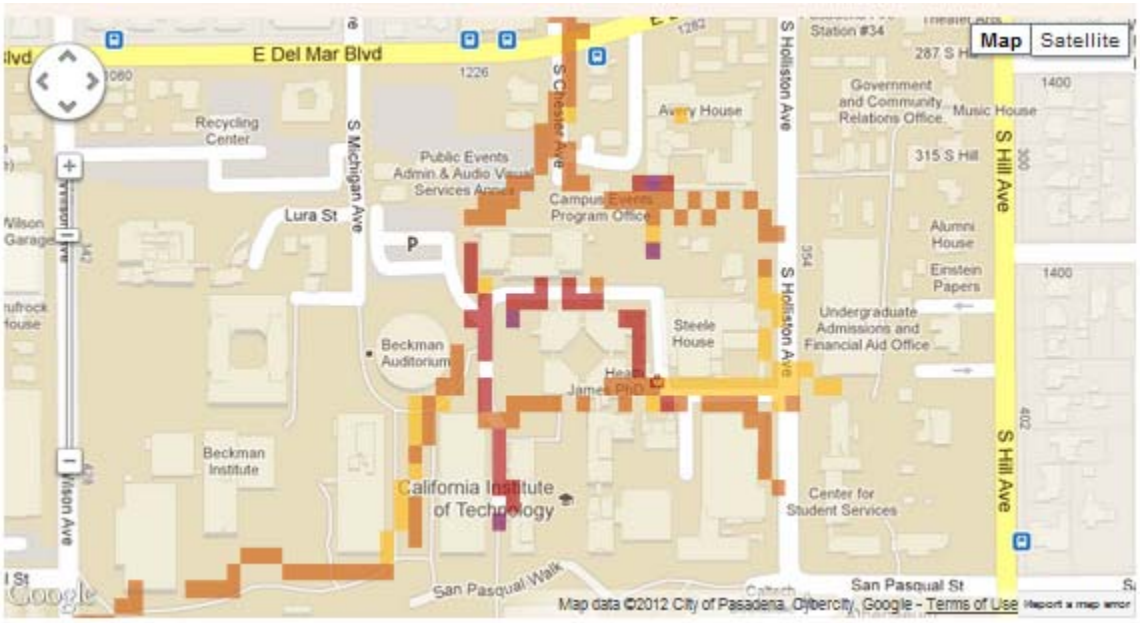


Figure 6. Heat map interface on website.

The data can also be filtered by making HTTP GET requests to http://www.anjoola.com/donut/data.php with the following parameters:

Parameter	**Details**
`minLatitude`	Lesser latitude of the retrieval area, in degrees.
`minLongitude`	Lesser longitude of the retrieval area, in degrees.
`maxLatitude`	Greater latitude of the retrieval area, in degrees.
`maxLongitude`	Greater longitude of the retrieval area, in degrees.
`carrier`	(Optional parameter). The carrier.
`phoneType`	(Optional parameter). The phoneType, one of 1, 2, 3.
`clientId`	(Optional parameter). Records from only this client retrieved.

Table 4. Parameters for HTTP GET request.

The user may also choose to filter the data by a minimum or maximum latitude and/or longitude, and can sort the table in order of carrier, location, or signal strength.