An Implementation and Evaluation of WiFi Positioning Algorithms in Android

Introduction

WiFi access points (deployed throughout our campus) allow laptops, smartphones, etc to connect to a WLAN infrastructure (e.g., zephyros, newcampus, etc.) These APs (each uniquely identified by a MAC address) beam signals that can be intercepted by designated software (coined Received-Signal-Strength RSS Scanners). If you happen to have an Android or IPhone smartphone try to download one of these scanners (e.g., "Wifi Analyzer").

For this project you will be provided a real HTC Desire Smartphone. The device can be programmed with eClipse and JAVA (eClipse provides an emulator that has however no access to GPS, WiFi, etc.). After compiling your program with eClipse an APK (executable) file will emerge. The file can be copied and installed on the device (all programming info is here: http://developer.android.com/sdk/index.html). You will also be provided a skeleton implementation of an RSS Scanner.

The project aims to develop the following two APKs:

a) **RSS Logger (Simple):** You are asked to extend the RSS Scanner in a way that a number of samples (N), which contain the recorded signals from neighboring APs, are stored in a file along with the respective Geolocation (GPS) at predefined intervals. In the following example, the user sets N=3 and indicated twice, i.e. at location (`35.131141, 33.362732`) and location (`35.07356, 33.408394`) possibly by clicking a button, that he/she wants to record the following info to a file (stored on flash).

`rss-log.txt`

```
# Timestamp, Longitude, Latitude, MAC Address of AP, RSS
1297874476, 35.131141, 33.362732, 00:0b:fd:4a:71:ab, -96
1297874476, 35.131141, 33.362732, 00:0b:fd:cd:91:28, -91
1297874476, 35.131141, 33.362732, 00:0b:fd:4a:71:d2, -70
1297874476, 35.131141, 33.362732, 00:0b:fd:4a:71:89, -79
1297874476, 35.131141, 33.362732, 36:26:55:8b:65:9b, -73

# Timestamp, Longitude, Latitude, MAC Address of AP, RSS
1297874486, 35.131141, 33.362732, 00:0b:fd:4a:71:ab, -95
1297874486, 35.131141, 33.362732, 00:0b:fd:cd:91:28, -92
1297874486, 35.131141, 33.362732, 00:0b:fd:4a:71:d2, -72
1297874486, 35.131141, 33.362732, 00:0b:fd:4a:71:89, -76
1297874486, 35.131141, 33.362732, 36:26:55:8b:65:9b, -75

# Timestamp, Longitude, Latitude, MAC Address of AP, RSS
```

```
1297874496, 35.131141, 33.362732, 00:0b:fd:4a:71:ab, -94
1297874496, 35.131141, 33.362732, 00:0b:fd:cd:91:28, -90
1297874496, 35.131141, 33.362732, 00:0b:fd:4a:71:d2, -71
1297874496, 35.131141, 33.362732, 00:0b:fd:4a:71:89, -79
1297874496, 35.131141, 33.362732, 36:26:55:8b:65:9b, -72

# Timestamp, Longitude, Latitude, MAC Address of AP, RSS
1297874507, 35.07356, 33.408394, 00:0b:fd:4a:71:ab, -95
1297874507, 35.07356, 33.408394, 00:0b:fd:cd:91:28, -90
1297874507, 35.07356, 33.408394, 00:0b:fd:4a:71:d2, -69
1297874507, 35.07356, 33.408394, 00:0b:fd:4a:71:89, -88
1297874507, 35.07356, 33.408394, 36:26:55:8b:65:9b, -88
1297874507, 35.07356, 33.408394, 00:1b:11:6a:18:0f, -63

# Timestamp, Longitude, Latitude, MAC Address of AP, RSS
1297874517, 35.07356, 33.408394, 00:0b:fd:4a:71:ab, -97
1297874517, 35.07356, 33.408394, 00:0b:fd:cd:91:28, -89
1297874517, 35.07356, 33.408394, 00:0b:fd:4a:71:d2, -68
1297874517, 35.07356, 33.408394, 00:0b:fd:4a:71:89, -85
1297874517, 35.07356, 33.408394, 36:26:55:8b:65:9b, -86
1297874517, 35.07356, 33.408394, 00:1b:11:6a:18:0f, -65

# Timestamp, Longitude, Latitude, MAC Address of AP, RSS
1297874527, 35.07356, 33.408394, 00:0b:fd:4a:71:ab, -94
1297874527, 35.07356, 33.408394, 00:0b:fd:cd:91:28, -91
1297874527, 35.07356, 33.408394, 00:0b:fd:4a:71:d2, -71
1297874527, 35.07356, 33.408394, 00:0b:fd:4a:71:89, -86
1297874527, 35.07356, 33.408394, 36:26:55:8b:65:9b, -83
1297874527, 35.07356, 33.408394, 00:1b:11:6a:18:0f, -67
```

b) **Radio Map Constructor (Simple):** Build a tool implemented in JAVA, which will be used to convert `rss-log.txt` into a radiomap coined `radio-map.txt`. The radiomap is a `lx(n+2)` matrix, where `l` is the number of distinct geolocations and `n` is the total number of APs in the area (each column contains the signal values of a specific AP and the 2 extra columns contain the longitude and latitude). The detailed procedure for the radiomap creation is enumerated as follows (example follows):

1. Parse `rss-log.txt` file to get all different MAC addresses
2. Calculate the average RSS value per MAC address (recall that you have N(=3) samples for each geolocation). If a MAC address is missing for a specific geolocation, i.e. the AP is not detected at that location, you can use a small RSS value, e.g. `-110` to indicate this.
3. Record this info in `radio-map.txt` file in a way that each line corresponds to the RSS values of a discrete geolocation point and each column to a discrete MAC address. Each position of this array matrix contains the averaged RSS value.

radio-map.txt (ignore anything in the parenthis)

```
(first line) # Longitude, Latitude, 00:0b:fd:4a:71:89, 00:0b:fd:4a:71:ab,
00:0b:fd:4a:71:d2, 00:0b:fd:cd:91:28, 00:1b:11:6a:18:0f, 36:26:55:8b:65:9b
(second line) 35.131141, 33.362732, -78, -95, -71, -91, -110, -73.3
```

```
(third line) 35.07356, 33.408394, -86.3, -95.3, -69.3, -90, -65, -85.7
...
```

c) **Positioning Algorithm (Moderately Difficult):** Implement an Android Client (referred to as "Client" hereafter) that connects to a C or JAVA Socket Server (referred to as "Server" hereafter) on a non-IANA reserved port (e.g., 66000). The Client and Server implement an unencrypted text protocol to download the radiomap to every interested client. In particular,

```
-- CLIENT Connects to socket 660000

SERVER: +OK READY

CLIENT: GET radiomap

-- the server now reads radio-map.txt from a file and
sends it over to the client.

SERVER: RADIOMAP firstline | secondline | ... | lastline

-- CLIENT Closes the connection after reading EOF
```

The Android UI now shows a "Downloading Radiomap Complete" message. After downloading the radiomap, the client must position itself (possibly by pressing a "Find me" button) using the currently observed RSS values. First, the following two positioning algorithms will be used and then variations of these algorithms (or other algorithms) can also be tested.

Algorithm 1: Nearest Neighbor algorithm

1. Calculate the Euclidean distance (D) between the currently observed RSS values and the RSS values for each geolocation in the radiomap
2. The geolocation that has the minimum D is returned (and possibly plotted on Google Maps)

Example

Assume that the currently observed RSS values at the unknown user location are `(-80, -92, -76, -110, -63, -70)`. Then, for the 2 geolocations in the radiomap we have

D_1 = sqrt((-78-(-80))^2+(-95-(-92))^2+(-71-(-76))^2+(-91-(-110))^2+(-110-(-63))^2+(-73.3-(-70))^2) = 51.1751

D_2 = sqrt((-86.3-(-80))^2+(-95.3-(-92))^2+(-69.3-(-76))^2+(-90-(-110))^2+(-65-(-63))^2+(-85.7-(-70))^2) = 27.3123

Because $D_2 < D_1$, the estimated user location is (`35.07356, 33.408394`).

Algorithm 2: Probabilistic algorithm

1. Calculate the probability (P) of the user being at each geolocation based on the currently observed RSS values and the RSS values for each geolocation in the radiomap. The probability P is given as the product of the probabilities p_i, $i=1,\ldots,n$ for all APs. Each probability p_i is calculated as an exponential function of the difference between the currently observed RSS value for an AP and the respective RSS value for the same AP in the radiomap (see example below).
2. The geolocation that has the maximum P is returned and plotted on Google Maps (see Android Google Maps API)

Example

Assume that the currently observed RSS values at the unknown user location are `(-80, -92, -76, -110, -63, -70)`. Then, for the first geolocation in the radiomap we have:

$p_1 = \exp(-(-78-(-80))^2/\sigma^2) = 0.8948$ (σ is a user defined parameter and in this example $\sigma = 6$)
$p_2 = \exp(-(-95-(-92))^2/\sigma^2) = 0.7788$

Similarly, $p_3 = 0.4994$, $p_4 = 4\text{x}10^{-5}$, $p_5 = 2\text{x}10^{-27}$ and $p_6 = 0.7390$.
So, $P_1 = p_1 \cdot p_2 \cdot p_3 \cdot p_4 \cdot p_5 \cdot p_6 = 2.5\text{x}10^{-32}$

For the second geolocation in the radiomap we have:

$p_1 = \exp(-(-86.3-(-80))^2/\sigma^2) = 0.3320$
$p_2 = \exp(-(-95.3-(-92))^2/\sigma^2) = 0.7390$

Similarly, $p_3 = 0.2874$, $p_4 = 1.5\text{x}10^{-5}$, $p_5 = 0.8948$ and $p_6 = 0.0011$.
So, $P_2 = p_1 \cdot p_2 \cdot p_3 \cdot p_4 \cdot p_5 \cdot p_6 = 1\text{x}10^{-9}$

Because $P_2 > P_1$, the estimated user location is `(35.07356, 33.408394)` which is the same as with the Nearest Neighbour algorithm.

Measure and plot respective Excel plots for the following parameters:

- **Execution time**: Run each positioning algorithm several times (100, 1000 or more) and measure the average time required to perform positioning.
- **Power consumption**: Download "PowerTutor" from Android Market and run each positioning algorithm several times to measure the average amount of power that was consumed to execute the positioning algorithm.
- **Positioning error:** Run each positioning algorithm several times and measure the average positioning error, i.e. distance between the location estimated with the positioning algorithm and the actual user location (available from GPS). This will give the accuracy of each positioning algorithm.

Please create an SVN repository for your project (see helpdesk) coined "AndroidRSS" (see http://helpdesk.cs.ucy.ac.cy/subjectview4.php?which=6627). Add yourselves, "dzeina" to the requested list and ask your instructor to acknowledge the request.