Lab Report for EI447 - Mobile Internet

Lab 2: WiFi Scanner & WiFi Positioning

Yiwen Song

2020-4-24

Preliminaries

Development Environment

- Operating system: Windows 10, function version 1909.
- IDE: Android Studio 3.6.2 (latest version).
- Build: Gradle 5.6.4.
- JDK: Java 1.8.0_212 for AS Runtime Environment. Java > 8 not suggested for the lab project.
- Android Simulator: Pixel 2 API Q (1080*1920; Android 10.0/Pure Android; x86).
- Android Tester: OXF-AN00/Honor V30 (1080*2400; Android 10.0/MagicUI 3.0.1; ARM).

Software Information

- Support Android > 5.0 devices.
- Open-source under GNU/GPL License v3.0. Any modifications/redistributions of the project software must be restricted by the GPL license. For more information, check the *open source* of the program.
- Copyright information: During the development of the project, the following external references have been used:
  - Template for the report is *eisvogel*.
  - Based on *WiFi Scanner* given by Intelligent Internet of Things (IIoT) Lab, Shanghai Jiao Tong University.

WiFi Scanner

There is no structural change for the *WiFi Scanner* part of the program. Several changes are made to fit the current Android programming convention, which will not be talked about here (Just check the code). I only mention here that in the given code, the array *RSS_Measurement_Number_Record* is initialized to 1, which is wrong. In the published code, the variable is initialized to 0, and a check mechanism has been set to avoid 0 division. In the following part, the three questions are answered.

- Why is necessary to record all the measured value rather than only the average value? Please give your own explanation.

> For any distribution of the signal strength, the Received Signal Strength (RSS) we use is always the average value of the detected RSS. However, other parameters for the distribution, especially variation/correlation are also usually useful for further processing. In this lab, I haven't designed system to process and utilize these data. However, in order to gain more robustness and scalability, recording all the measured values is necessary.

- Besides the WiFi signal strength, what other information of the Routers can be got in the test?

> According to the official document for *ScanResult*, we can find the types of information that can be provided about the routers. It can provide *SSID* and *BSSID*; the authentication, key management, and encryption schemes (*capabilities*); center frequency for the WiFi band (*centerFreq0*, *centerFreq1*); channel width (*channelWidth*); primary frequency (*frequency*); detected signal strength (*level*, used here); Pass-point operator name published by access point (*operatorFriendlyName*); *timestamp*; and venue name (*venueName*).

- Why does the scanning need to be operated in thread "scanThread"?

> The main purpose of separating the scanning procedure in a single thread is to allow concurrency and parallelism. Other reasons include easiness for debugging, increasing stability, and providing security.

Indoor Positioning

An easy WiFi RSS indoor positioning system is designed using LANDMARC algorithm proposed by L. Ni, etc., 2003[1] and tested in a virtual environment with real WiFi signals.

Distance Estimation

According to free-space propagation model for wireless communication, we have

$$P_r = P_t G_t G_r (\frac{\lambda}{4\pi d})^2,$$

where P_r is the receive power (in mW), P_t is the transmit power, G_t is the transmit antenna gain, G_r is receive antenna gain, λ is the wavelength, and d is the distance between receiver and transmitter. By expressing RSS in dBm and excluding other parameters, we have

[1] L. M. Ni, Yunhao Liu, Yiu Cho Lau and A. P. Patil, "LANDMARC: indoor location sensing using active RFID," Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003)., Fort Worth, TX, 2003, pp. 407-415.

$$RSS = 10\log_{10} P_r = X - 20\log_{10} d,$$

where

$$X = 10\log_{10}\left[P_t G_t G_r (\frac{\lambda}{4\pi})^2\right].$$

Therefore we can express d with regard to RSS in the following equation.

$$d = 10^{\frac{1}{20}(-RSS+X)}.$$

By multiple experiments we can estimate that $X \approx 23$. Thus we can estimate the distance from the WiFi routers to our cell phones.

Positioning

According to LANDMARC algorithm, we can estimate our location, given the positions of k routers and the distances between them and our cell phones. Firstly, we define a weight vector $w = (w_1, w_2, \cdots, w_k)$ by

$$w_i = \frac{1/d_i}{\sum_{i=1}^{k} 1/d_i},$$

where d_i is the distance from us to the ith router. Then we can estimate our position by

$$(x, y) = \sum_{i=1}^{k} w_i(x_i, y_i).$$

We created a class called *Locate* to realize the LANDMARC algorithm, shown as below.

```
class Locate {

    private double loc_X, loc_Y;

    Locate(int[] RSS_Record, int[][] POS) {
        if (RSS_Record.length < 3 || POS.length != RSS_Record.length) {
            loc_X = -1; loc_Y = -1;
        }
        else {
            loc_X = 0; loc_Y = 0;

            // Location algorithm, use LANDMARC here.
            Vector<Double> dis = new Vector<>(RSS_Record.length);
            double total_dis = 0;
            for (int rss : RSS_Record) {
                double d = 1.0 / distance(rss);
                dis.add(d);
                total_dis += d;
            }
            Vector<Double> weight = new Vector<>(RSS_Record.length);
            for (double d : dis) {
                weight.add(d/total_dis);
            }
            for (int i = 0; i < RSS_Record.length; ++i) {
                loc_X += weight.get(i) * POS[i][0];
                loc_Y += weight.get(i) * POS[i][1];
            }
        }
    }

    Vector<Double> location() {
        Vector<Double> loc = new Vector<>(2);
        loc.add(loc_X); loc.add(loc_Y);
        return loc;
    }

    private double distance(int RSS_value) {
        int tmp = (-RSS_value+23)/20;
        double dis = 10^tmp;
        return dis;
    }
}
```

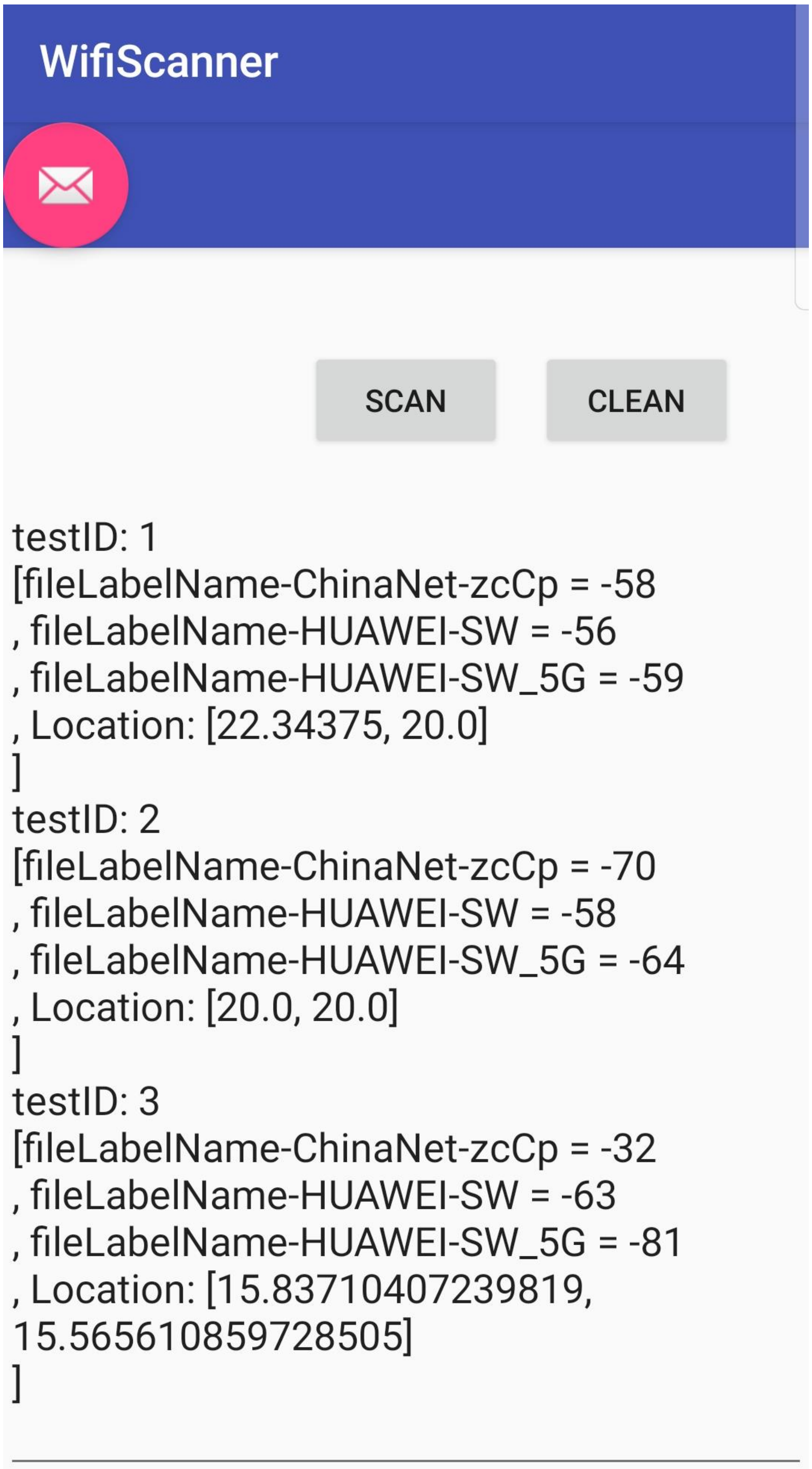
After calculating the location in the initialization function, we can use the *location()* function to find our current position. And in *SuperWiFi*, we just need to add a line to show our position.

```
Locate l = new Locate(RSS_Value_Average, POS);
Vector<Double> loc = l.location();
scanned.add("Location: "+loc.toString()+"\r\n");
```

In our positioning part, we have not considered signal fading due to terrain or inter-signal interference that may cause additional noise. However, the positioning algorithm still can work well in open environments with weak terrain.

Demonstration

We created a virtual environment with 3 routers “ChinaNet-zcCp”, “HUAWEI-SW”, “HUAWEI-SW_5G”, and their respective positions {0, 0}, {35, 20}, {25, 40}. Then by the real detected WiFi signals we can achieve the following result.



```
testID: 1
[fileLabelName-ChinaNet-zcCp = -58
, fileLabelName-HUAWEI-SW = -56
, fileLabelName-HUAWEI-SW_5G = -59
, Location: [22.34375, 20.0]
]
testID: 2
[fileLabelName-ChinaNet-zcCp = -70
, fileLabelName-HUAWEI-SW = -58
, fileLabelName-HUAWEI-SW_5G = -64
, Location: [20.0, 20.0]
]
testID: 3
[fileLabelName-ChinaNet-zcCp = -32
, fileLabelName-HUAWEI-SW = -63
, fileLabelName-HUAWEI-SW_5G = -81
, Location: [15.83710407239819,
15.565610859728505]
]
```

Figure 1: Demo for indoor positioning