LAB 2

MEASUREMENT OF WIFI SIGNAL STRENGTH

April 25, 2020

Name:Chenyu Yang
Student ID: 517030910386
Class: F1703015

Contents

1 INTRODUCTION

1.1 Requirement

Get used to the WiFi system and accomplish the sampling and measuring of WiFi signal strength through programming in Android on smartphone.

1.2 Background

Researchers are generally required to complete mobile terminal scanning to indoor WiFi Routers, especially in indoor positioning, WiFi access point selection and other popular fields. As the result of the dynamic characteristics of the wireless channel, the wireless signal received by mobile terminals is always unstable. The signal measured at a settled position from a certain router is distributed in a specific distribution(not a fixed value). Therefore, the signal should be measured repeatedly, though the position is fixed. Then, the result needs to be analyzed as random data in a certain distribution.

1.3 Environment

1. OpenJDK 64-bit;
2. Android 9.0 (pie);
3. Android Studio 3.3.2;
4. MI 6; (for test)

2 QUESTIONS

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

A: Recording all values helps to inspect the distribution of the signal strength at every time. If the signal strength is not stable,it will have a negative impact on the final accuracy. So we can choose the more stable source according to all measured values. Apart from this, the detail data can also help us to debug the program and go on further implemention.

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

A: The class *scanResult* includes:

Type	Name	Explanation
String	BSSID	The address of the access point.
String	SSID	The network name.
String	capabilities	Describes the authentication, key management, and encryption schemes supported by the access point.
int	centerFreq0	Not used if the AP bandwidth is 20 MHz If the AP use 40, 80 or 160 MHz, this is the center frequency (in MHz) if the AP use 80 + 80 MHz, this is the center frequency of the first segment (in MHz)
int	centerFreq1	Only used if the AP bandwidth is 80 + 80 MHz if the AP use 80 + 80 MHz, this is the center frequency of the second segment (in MHz)
int	channelWidth	AP Channel bandwidth.
int	frequency	The primary 20 MHz frequency (in MHz) of the channel over which the client is communicating with the access point.
int	level	The detected signal level in dBm, also known as the RSSI.
CharSequence	operatorFriendlyName	Indicates passpoint operator name published by access point.
long	timestamp	timestamp in microseconds (since boot) when this result was last seen.
CharSequence	venueName	Indicates venue name (such as 'San Francisco Airport') published by access point; only available on Passpoint network and if published by access point.

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

A: Because the scanning process takes long time to scan and calculate the average. If it's not operated in a separate thread, it will block the main thread and the UI(for example, the *SCAN* botton) will make no responce to user's action until a scanning process is finished.

3 POSITIONING SYSTEM DESIGN

In this part, we will design an indoor positioning system based on WiFi signal strength by using the signal strength measuring function of Android mobilephone. Any positioning algorithm is available.

3.1 introduction

At present, the main indoor positioning algorithms are based on triangular positioning algorithm, fingerprint positioning algorithm and centroid positioning algorithm. The core idea of the first two algorithms is to get the signal strength RSSI (Received Signal Strength Indication) sent by each AP (Access Point) through the SDK of the mobile phone system. Received Signal Indicator Strength) and AP address, locating according to signal attenuation model (signal strength is related to distance from the source), triangular and fingerprint locations use different methods to locate the location, while centroid locations use network connectivity to locate.

3.1.1 Triangular positioning algorithm

Triangular positioning algorithms place three AP's in the space where they need to be positioned, and the positions of three AP's are known. If you know the signal strength of a location in the space, you can build a signal attenuation model. Based on the signal strength, you can calculate the distance from the location to the three AP's, use the signal strength of a mobile device at a location to estimate the distance from the nearby AP, and if you can determine the location of several AP'sThe location of the mobile device can be determined. Obviously, the triangular location algorithm is only suitable for fixed location, and there will be some error in the signal attenuation model in practical use due to the presence of obstacles.

3.1.2 Fingerprint positioning algorithm

Fingerprint positioning algorithm is a set of algorithms based on the complex indoor environment and different signal strength information formed by reflected refraction. The fingerprint algorithm can make good use of the signal information formed by reflected refraction. Offline, the fingerprint signal strength database is formed, and the location distance is calculated by a set of RSSI values measured in the online location.

3.1.3 Centroid positioning algorithm

The centroid algorithm is a centroid location algorithm based on network connectivity. Its advantage over other algorithms is that it does not need to measure distance. The basic idea is that unknown nodes take the geometric centroid of all anchor nodes within their communication range as their own estimated location.The traditional centroid algorithm is low in cost, small in computation and easy to implement.However, because positioning accuracy depends too much on node density, it is also less accurate.For this reason, many researchers have improved the centroid location algorithm in different ways, such as centroid weighting based on reciprocal distance, centroid weighting based on reciprocal RSSI, and weighted centroid location based on maximum likelihood estimation.

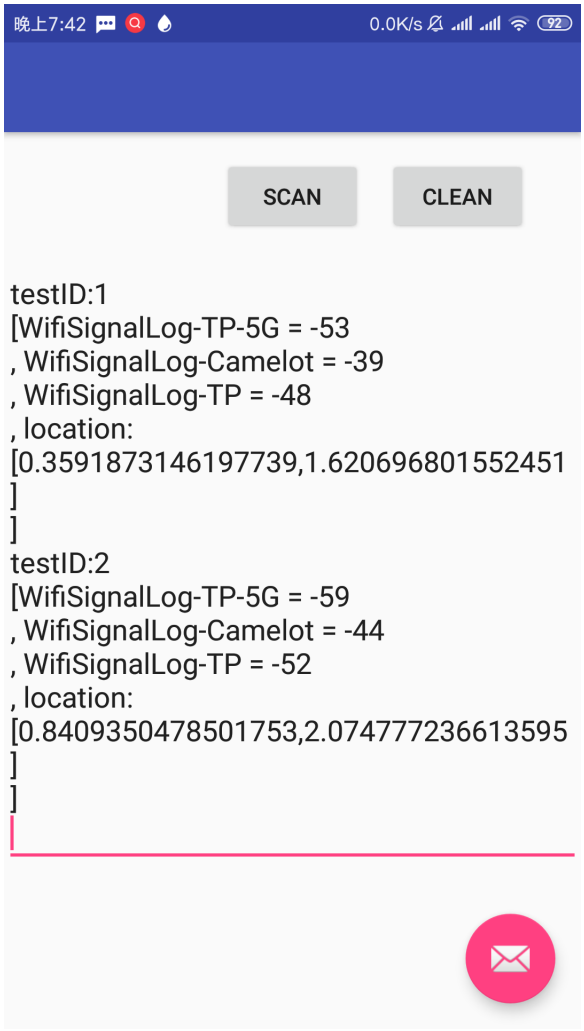
3.2 Procedure

Considering we have not enough test devices, the Centroid Algorithm is not available. And Fingerprint Algorithm takes a lot of time to collect the signals at defferent locations. We finally decide to use Triangular Algorithm, though it may not be as accurate as others. This algorithm has two steps, the first is to calculate the distance based on RSSI. Here we use the formula $d = 10^{\frac{abs(RSSI)-A}{10*n}}$, where A,d is two constants for different APs. The second step is to lacate based on the distance of each AP. This is a geomatric problem and we can treat it as solving equaltions.

```
public class ThreePointLocation {

  // d=10^((ABS(RSSI)-A)/(10*n)) A:45~49; d: 3.25~4.5
  public class NodeInfo {
    public double location[];
    public int A;
    public double n;

    NodeInfo(int A_in, double n_in, double[] loc) {
      A = A_in;
      n = n_in;
      location = loc;
    }

    public double calculate_distance(int RSSI) {
      double distance = Math.pow(10, (Math.abs(RSSI) - A) / (10 * n));
      return distance;
    }
  }

  public NodeInfo node1;
  public NodeInfo node2;
  public NodeInfo node3;

  ThreePointLocation() {
    node1 = new NodeInfo(45, 4, new double[]{0, 0});
    node2 = new NodeInfo(46, 4, new double[]{0, 2.37});
    node3 = new NodeInfo(46, 4, new double[]{1.41, 0});
  }

  // This function refers to https://blog.csdn.net/u013780605/article/details
     /52673223
  private double[][] intersect(double r1, double r2) {
    double x1 = node1.location[0], y1 = node1.location[1], x2 = node2.location[0], y2
        = node2.location[1];

    double A = (x1 * x1 - x2 * x2 + y1 * y1 - y2 * y2 + r2 * r2 - r1 * r1) / (2 * (y1
        - y2));
    double B = (x1 - x2) / (y1 - y2);

    double a = 1 + B * B;
    double b = -2 * (x1 + (A - y1) * B);
```

```
    double c = x1 * x1 + (A - y1) * (A - y1) - r1 * r1;

    double delta = b * b - 4 * a * c;

    double x_1, x_2, y_1, y_2;
    if (delta <= 0)
      delta = 0;
    x_1 = (-b + Math.sqrt(delta)) / (2 * a);
    x_2 = (-b - Math.sqrt(delta)) / (2 * a);
    y_1 = A - B * x_1;
    y_2 = A - B * x_2;

    return new double[][]{{x_1, y_1}, {x_2, y_2}};
  }

  public double cal_distance(double[] p1, double[] p2) {
    double res = (p1[0] - p2[0]) * (p1[0] - p2[0]) + (p1[1] - p2[1]) * (p1[1] - p2
        [1]);
    return Math.sqrt(res);
  }

  public double[] location(int RSSI1, int RSSI2, int RSSI3) {
    double r1 = node1.calculate_distance(RSSI1);
    double r2 = node2.calculate_distance(RSSI2);
    double r3 = node3.calculate_distance(RSSI3);

    double intesect_points[][] = intersect(r1, r2);

    double d1 = cal_distance(intesect_points[0], node3.location);
    double d2 = cal_distance(intesect_points[1], node3.location);

    if (Math.abs(d1 - r3) < Math.abs(d2 - r3))
      return intesect_points[0];
    else
      return intesect_points[1];
  }
}
```

3.3 result



4 References

- https://blog.csdn.net/xx326664162/article/details/49385761
- https://www.jianshu.com/p/2d071581b468
- https://blog.csdn.net/u013780605/article/details/52673223

5 Complete Code

https://github.com/Achronferry/SJTU-EE447-Lab/tree/master/lab2/WiFi_Scanner_IIoT