Android Wi-Fi Location Awareness and Data Inference Heuristic

by

Leon Wu

A Thesis Submitted in Partial Fulfillment

of the Requirements for Degree of

Master of Applied Science (MASc)

in

Electrical, Computer and Software Engineering

University of Ontario Institute of Technology

Oshawa, Ontario, Canada

(December, 2013)

Abstract

Mobile phones are becoming a primary platform for information access. More and more people use their mobile devices as one of their major communication access tools. Commuters are increasingly carrying their mobile devices with them almost everywhere. Mobile devices fit perfectly into an ideal environment for realizing ubiquitous computing. A major aspect of ubiquitous computing is context-aware applications where the applications collect information about the environment that the user is in and use this information to achieve their goals or improve performance. The location of the device is one of the most important pieces of context information. Location awareness makes certain applications possible, e.g., recommending nearby businesses and tracking estimated routes, and greatly improves the performance of other applications, for example it can be associated with automobile navigation devices. A feature available to mobile applications in the Android platform makes it possible to determine a device's location without any additional hardware or sensor mechanisms, by simply using the native capability of the built-in wireless network card. Since the release of Android systems, there have been numerous applications developed to introduce new ways of tracking locations. Recently, there have been many papers on location estimation leveraging ubiquitous wireless networks.

In this thesis, we develop an Android application to collect useful Wi-Fi information without registering a location listener with a network-based provider, such as Wi-Fi connections or data connections. Therefore it provides a passive, privacy-preserving, non-intrusive and power-saving way of achieving location awareness to Android mobile users. Accurate estimation of the location information can bring

a more contextual experience to mobile users. We save the passively collected data of the IDs of Wi-Fi access points and the received signal strengths to a database in order to help us structure the data and analyze it. We employ some heuristics to infer the location information from the data.

Our work presents a location tracking technique mainly based on Basic Service Set Identification (BSSID) and/or Received Signal Strength Indicator (RSSI) using Wi-Fi information. It falls into one of the most active fields in mobile application development --- location-based or location-aware applications.

Acknowledgements

I would like to express my deepest appreciation to my supervisor Dr. Ying Zhu for her professionalism, kindness, encouragement, efforts, and dedications. She continuously and convincingly conveyed a spirit of adventure in regard to research and an excitement to teaching. Without her guidance and persistent help, this dissertation would not be possible.

My special appreciation goes to Dr. Ken Pu for his friendship, guidance and numerous fruitful discussions. He has been extremely helpful and supportive during my research.

Last but not the least, I would like to thank my family in Canada, US, and China for supporting me throughout my entire study period.

Table of Contents

List of Figures

Chapter 1 Introduction

1.1 Problem Definition:

Location awareness refers to devices that can passively or actively determine their locations [1]. Mobile devices can provide such capability without requiring a device to actively register with a Wi-Fi network. In this thesis, we develop an Android application that can assist and remind commuters where they are while on the road. For example, due to the increase of home prices and air pollution, many commuters adapt to use public transit such as the subway for travelling. Most of them are either resting or working/studying during the trip. There is a chance that they could miss their stops, but the notifications from their mobile devices would remind them for various purposes. Location awareness applications would support such a feature on the mobile devices for commuters to manage their road trips more efficiently.

Most navigation equipment associated with location tracking or awareness methods use Global Positioning System (GPS) technology to get Ephemeris information such as their current orbital position and almanac data. There are probably thirty-one satellites zipping around the world with nothing better to do than help us find ways to the grocery store. This is why GPS, originally developed by the military but then converted to civilian use, becomes highly populated because it beams precise time signals to Earth-based receivers such as mobile devices. Because of the slow data rate of the transmission—in some cases, as low as 50 bits per second [2]—it can take proximately 30 seconds for a standalone GPS receiver to get the complete information required from each satellite used in establishing a fix, which requires at least three satellites. During this process, GPS receivers consume a lot of power doing the signal processing needed to lock onto

satellites signals and adjust those signals as satellites zoom past in their orbits. Location awareness application service based with Wi-Fi information can save at least 30% battery consumption compared with GPS. Anyone who makes use of GPS functionality on a smartphone understands the impact such connectivity has on the limited battery supply that's packed inside the device. The phone device can even become over heated from an extended period of use.

GPS devices can be an excellent navigation aid in low visibility situations. Buy a unit if you partake in activities where a GPS can provide valuable location-based information that you cannot get with your own eyes. However, GPS requires a clear sky to work and hence does not always work perfectly indoors or where satellites cannot penetrate, such as a tunnel or subway system. Wi-Fi technology has been deployed throughout urban areas such as the Greater Toronto Area. Does it make us think that Wi-Fi is on its way to replace or at least substitute of GPS?

1.2 Background and Motivation:

Mobile devices and personal digital assistants have received more attention in recent years around the world. Even the most famous Microsoft Chairman Bill Gates has been recently under pressure by investors and some major shareholders to step down because he was accused that the company has blocked the adoption of new strategies to make substantial changes to mobile devices. Most mobile devices weigh around 100 grams, and all the top of the line mobile devices can offer advanced web browsing and sophisticated touch-screen or speech interface features. Users can easily enter input queries with or without touching the screen of the mobile

phones. For example, an application can respond a voice query from the user about their locations.

Wi-Fi technology provides convenience for mobile users to access information. For example, coffee shops, shopping malls, airports, public transits and other venues increasingly offer wireless access to computer networks, referred to as hotspots, for users who bring their own wireless-enabled devices such as a laptop or mobile device. These services may be free to all, free to customers only, or fee-based.

Designing and developing products today in an increasingly globalized market has introduced new challenges for manufacturers across a range of industries – from automotive, medical devices, and aerospace to electronics and high-tech giants. There are several leading solutions for developing mobile services in 90s till now. One of the major serious players is Microsoft Windows CE.net with Embedded Visual C++ 4.0. However, the complexity of the product such as development cycle hurts the profitability. Consumers are demanding a broader range and selections of products, requiring businesses to innovate and rapidly deliver due to market demands. About a decade ago, Bell Fleet Solutions have strived to push their Telematics products equipped with Windows CE technology to the markets. The end result was not very successful partially due to the high cost their Window CE base Telematics product. In recent years, Apple IOS and Google Android developments dominated the mobile application market. The source model of Apple is closed source; therefore the closed and proprietary nature of iOS has garnered criticism. Google Android has particularly become more popular partially due to their open source platform. The following table show the comparison among these three giants.

	Google	Apple	Microsoft
Number of users(in millions)	900	600	12
Number of apps(in thousands)	800	1250	160
Number of developers (in thousands)	150	235	45
Number of downloads(in millions)	48	50	65

Table 1.1 Comparison among major high tech giants [3]

Location awareness using Wi-Fi information on mobile applications can provide significant help for travellers to manage their time on the road and improve travel efficiency that reduces time waste. Most importantly it can reduce the cost of purchase data plan to obtain the real time information. For example, there is no need to use data service from major service provider to keep the mobile device connected with Internet. It can also be used to track their devices if the location based database information is stored on a dedicated server or the cloud. Further enhancements can include location determination such as calculating distance while on the road.

Using Wi-Fi information to determine or estimate locations on mobile devices can also significantly reduce battery consumption compared to using GPS. For example, the service based Android application can run up to more than 12 hours, the battery consumption is only 50% from fully-charged. However if the GPS is turned on to collect location information, the battery will die within 5 hours.

1.3 Objective:

The main focus of this thesis is to develop a method and an Android application that provides mobile devices such as smart phones the capability of determining their locations using surrounding Wi-Fi network information.

The objective of this thesis is to develop a method and an application for mobile devices to collect and store information of nearby Wi-Fi access points (including ID and signal strength) into a database, and analyze the data using some heuristics to infer the location of the device.

1.4 Contributions:

We develop a privacy-preserving, power-saving and economical way of collecting Wi-Fi information without registering with Internet Service Providers (ISP)s. Users are not giving up their private information by registering their devices to the network. As we all know, purchasing a data plan with any of the major wireless providers such as Rogers, Bell or TELUS is typically costly. The method we use to get location information totally avoids the data usage charge and does not require data to be transferred to and from the Internet, therefore it provides a clean bill of health for mobile users to use the application on their mobile devices without worrying that their personal information may be leaked out.

1.4 Thesis Organization:

There are five chapters including in this thesis. Chapter 1 is the introduction to overall research about mobile application development in the current market, the objective of the research, and benefits of Google Android using Wi-Fi information. Chapter 2 related work about location awareness using different approaches.

Chapter 3 and Chapter 4 present methodologies of Android development using Java and Python using DB-API 2.0 interface with SQLite Database.

Finally, Chapter 5 summarizes what has been done in the thesis, future improvement, and market potentials.

Chapter 2 Review of Literature

2.1 Evolution of Location Sensing with Wi-Fi Information

More and more location awareness studies are accepting Wi-Fi technology as a subject worthy of business consideration. At its earlier stage, some related work is adopting the infrared (IR) sensors. However in order to obtain useful data, all the nodes in the network must equip with IR sensors, which adds additional overhead expense. Bell Fleets Solutions is one of the enterprises business modules using Kyocera wireless cards to send out the vehicle data including GPS information. However, one of the reasons Bell Fleet Solutions could not attract many customers is because the high cost of their Telematics products.

2.2 Location Fingerprinting

The method of using Wi-Fi based on the signal strength to determine the location in the buildings has been one of hot research topics in the past few years.

Location fingerprinting has been widely focused on indoor facilities even without much commercial success yet. It could be difficult to construct an accurate mapping using signal strength patterns and actual physical locations.

2.3 Related Work

An active part of the research in ubiquitous computing has been in location-aware computing [27]. There has been a great deal of interest in context-aware applications [32] and the location of devices on which the applications are executing is one of the main components of “context” for an application.

There have been many works, e.g., [5, 6, 7, 8, 9, 10, 11, 12, 13, 28], in the inference of in-door locations of mobile devices using the signal measures surveyed by the device in the environment of a wireless

LAN. Et al.[9] presents an error-minimization technique is used to infer locations from the measured signal strengths. But such methods suffer the weakness of lacking in robustness in the presence of noisy data. The works of [6] and [9] use the technique of multilateration whereby an object infers its location by calculating its range from beacons with known locations using the signal measure of radio frequency (RF). This is accurate in environments where there are no obstructive objects like walls which make such calculations from signal measures difficult. In [5], the authors designed and implemented a Wi-Fi location service to enable applications and devices to detect and make use of their dynamic environmental context, by inferring the room location of a mobile device in a wireless networked environment. As the device roams, it periodically surveys the received signal strength of the surrounding wireless access points. The observed Wi-Fi signal strength data are compiled into a Bayesian network. Once trained, the Bayesian network was used for inference of room location; due its probabilistic nature, this method is robust in the presence of (necessarily) noisy signal strength measurements.

Motivated by tracking children in a playground, the authors of [14] studied two methods of Wi-Fi localization using RSSI (received signal strength index) readings. The first is the triangulation method which maps RSSI as a function of distance which can then be used with live RSSI values as input to output a location prediction. The second is the fingerprinting method which creates a radio map of a given area based on RSSI data from a set of access points and also a probability distribution of RSSI values for each location, then the live RSSI values matched to the closest fingerprint to infer a location. The fingerprinting method is found to be much more accurate than the triangulation method. An experimental study is conducted in [15] to investigate the performance and deployment issues of location fingerprinting schemes.

The aim of the work in [10] is to investigate the quality of three different indoor localization methods based on the measured received signal strength index (RSSI) data on smart phones. The localization

algorithms studied and compared are minimal Euclidean distance, intersection of RSSI-Isolines (essentially a type of triangulation method) and a Bayesian stochastic model. The test environment was inside a single seminar room. Their methods depend on an initial calibration phase in which a radio map of RSSI values of all access points reachable from the room is developed with *known* coordinates. This map is subsequently compared with observed values to find the positions of the smart phone inside the room.

Several works, e.g., [11, 28, 31], have developed Bayesian-based probability models for received signal strengths to infer locations. The authors in [11] used fingerprints and a Bayesian inference algorithm to calculate the position of a mobile laptop with a modified Linux kernel driver for the wireless Ethernet card. The modification is to support the scanning and sampling of MAC addresses and signal strengths of packets. Experiments are also carried out in an indoor environment. In [28], experiments are conducted over a multi-floor office building with highly accurate results. A topological model of the physical environment is used to divide the indoor space into cells, and the mapping of the device location is to a cell instead of to a point --- which substantially reduced computation time of the training phase. An indoor positioning system was developed in [12] using RSSI fingerprints as well, in an office building environment, focusing on determining whether the user is inside or outside a room. Teuber et al. [13] combined the minimal Euclidean distance method together with fuzzy logic post-processing to find positions of mobile users inside an airport hangar.

All these related works are interesting and offer accuracy in localization. However, they differ from this thesis work in that they are only concerned with localization of a device in an indoor space, usually just one room. Some of them even require the prior knowledge of exact coordinates of access points which is a stringent requirement that we do not impose for our service to work.

Location algorithms are proposed in [8] to be used in conjunction with classical location algorithms such as triangulation. These algorithms infer user location by selectively fusing location information from multiple wireless technologies (Wi-Fi, Bluetooth) and multiple classical location algorithms in an optimal way. The performance evaluation of the algorithms is conducted in an indoor environment.

Recent work on radio map-based techniques may be categorized into deterministic ones, e.g., [6, 16] and probability distribution-based ones, e.g., [5, 11, 17, 18].

In both [6] and [16], the *a priori* knowledge of the physical positions of all access points in a given area are used to generate a function that maps signal strengths to physical distance. This function is obtained empirically from a set of trained points and observations. The function is then applied to real-time observed signal strengths to generate predictions of locations.

A WLAN location inference technique is proposed and implemented in [18] that uses a combination of signal strength probability distributions and clustering of locations. The probability distributions make the technique robust in the noisy wireless channel. The location clustering reduces the computational complexity of radio map searching. The method was implemented in an indoor space and showed high accuracy of locating the user.

The authors of [17] take a machine learning approach to the problem of location estimation. The approach is similar to that of [5], but is more generalized than the study in [5] which only focused on determining the room that the user is in. In [17], a Bayesian-based probabilistic model is developed to describe the distribution of received signal strengths at various locations, which can then be used to infer the location from real-time observed signal strengths.

Another recent work [29] implemented a real location estimation system called LEASE. It takes the interesting approach of installing a small number of low-power wireless transmitters and sniffers

throughout the region of interest to help with locating client devices. The emitters send a few packets periodically and the sniffers simply gather and forward the collected information of received signal strengths. The localization system collects all these data from the sniffers which are then used to develop and refine a signal strength model.

A recent work [19] investigates the problem of fingerprinting localization based on RSSI observations in the environment of a Wi-MAX network in an (outdoor) urban area. This work proposes a clever improvement to the classical fingerprinting method. The classical method tries to match measured signal strengths to the closest fingerprint of a base station, but ignores the ones that do not match at all. The proposed improvement is to make use of the negative information of non-matching fingerprints simultaneously. The reasoning is that if a base station's fingerprint does not match a measured value at all, then this base station must be far from the desired location. Given enough such information, the accuracy of locating the measured value could be increased.

An interesting study is presented in [30] where the authors considered the problem of wide-area Wi-Fi localization, and compared a number of radio signal-based location estimation algorithms with the aim of minimizing calibration efforts. They used algorithms originally proposed for indoor Wi-Fi localization. Their experiments showed that Wi-Fi localization can achieve reasonable accuracy in the wide-area deployment. Skyhook Wireless is a company that has collected more than a billion Wi-Fi access points information. It has the database systems that contain most coverage in the United States, Canada, Western Europe and selected Asian countries.

Chapter 3 Methodology

The nature of open source of Android development environment provides the incentives for amateurs and professional developers to make contributions to their Java-based application development and design their own versions of Android towards their needs. The size of the Android developer community has rapidly increased since the first Android powered phone released in 2008. Android applications are written primarily in the Java language, and it attracted a huge number of Java developers to join forces to extend the functionalities of the devices.

Overall this research includes two phases. The first one is the mobile application development that collects Wi-Fi information and stores in the database on the mobile devices. The second portion is the data inference heuristics including database recreations, data plot, data manipulation and data query. Both parts are closely related to complete the location awareness using Wi-Fi information.

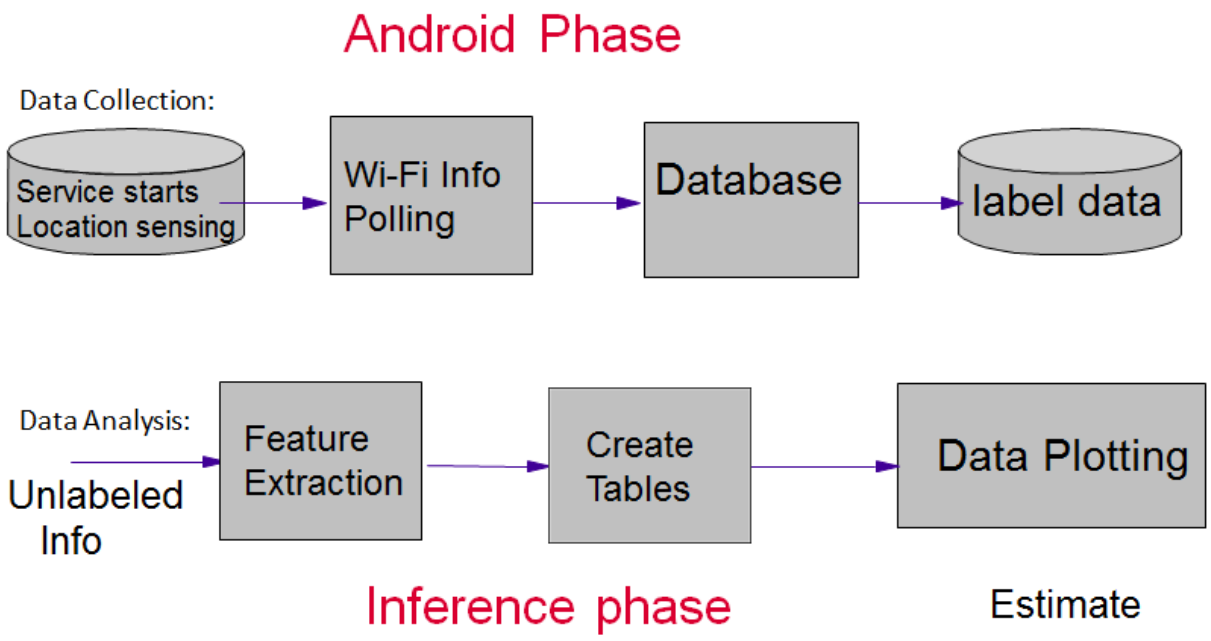


Figure3.1Overall Methodology

There are two modes that the wireless network interface controllers (WNIC) can operate. Devices including Notebook PC, Desktop PC with wireless cards and mobile devices in a wireless network are set up to either communicate with each other directly or through a central place, called an Access Point (AP), indirectly. The first mode is called “Ad hoc” mode that uses peer-to-peer (P2P) can act as a client or server allowing sharing various resources.

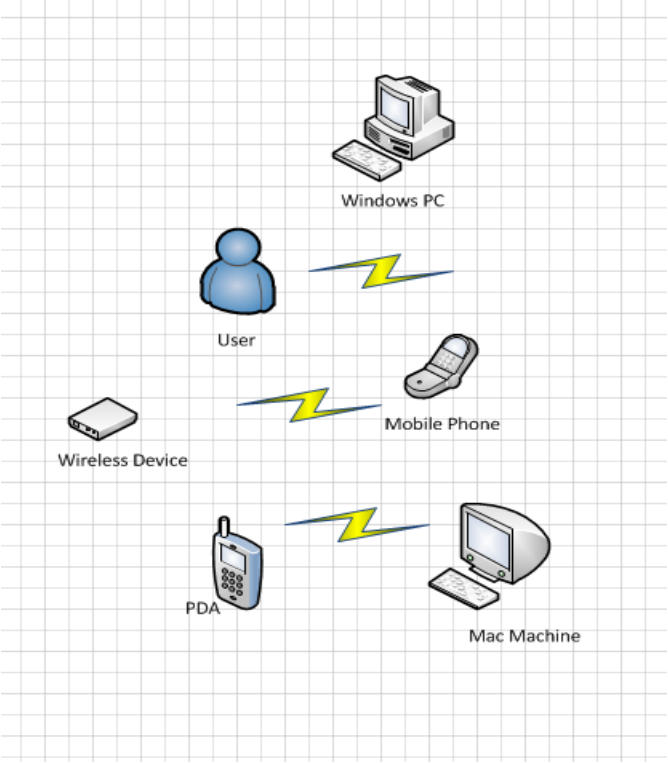


Figure3.2 Wireless Ad Hoc Network

The latter mode is called infrastructure mode and it is what this thesis is focused on. As the name of the network describes, the AP, as the central point of the network, controls the wireless communication. It has several advantages:

1. The AP determines the range of Basic Service Set (BSS) [20] coverage and communication.

2. There is no need for surrounding APs to keep track of its neighbour's relationship such as broadcasting periodically their MAC addresses; therefore it reduced the routing complexity in principle.

3. It is more robust than Ad hoc Networks especially when the traffic of the communication is heavy.

4. It is easy to manage the entire system by configuring the AP at real time.

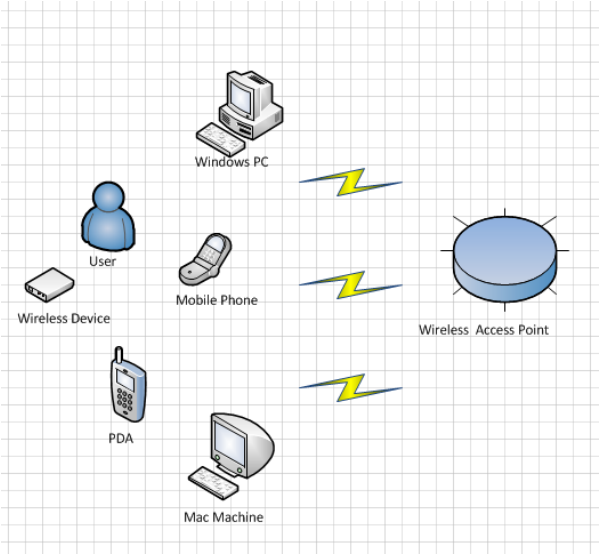


Figure3.3Infrastructure Wireless Networks

3.1 Android Development Overview

The majority of mobile application development has branched out in several major approaches, Google Android, Apple iPhone, RIM Blackberry, and Microsoft Windows 8. Android is an operating system based on the Linux operating system. The Android system supports background processing, provides a rich user interface library, supports 2-D and 3-D graphics using the OpenGL libraries, access to the file system and provides an embedded SQLite database.

Google provides graphical development environments based on the Eclipse IDE to develop applications. Android Developer Tool (ADT) is based on the Eclipse IDE and provides additional functionality to develop Android applications. Android platform and developer tools are excellent for programming mobile devices. Android applications are primarily written in the Java

language. The compiled Java code is compressed with an archive file with a suffix of .apk. This is the file that can be distributed in many ways for installing applications on mobile devices.

Application components are essential building blocks of an Android application. There are four major application components, activity: services, broadcast receivers and content providers.

The majority of Android applications use activities as the central feature because it has rich visual user interface for users to undertake. One of most popular features on mobile devices is the text messaging application, especially for young adults as they text each other frequently. A text messaging application might have one activity for showing the list of contacts and another activity to review the history of the sent folder and so on. All those activities work together to form a cohesive user interface. In this research, the application only consists of one activity because the focus is not on user interface. Figure 3.4 shows the simplified lifecycle of an Activity at important states.

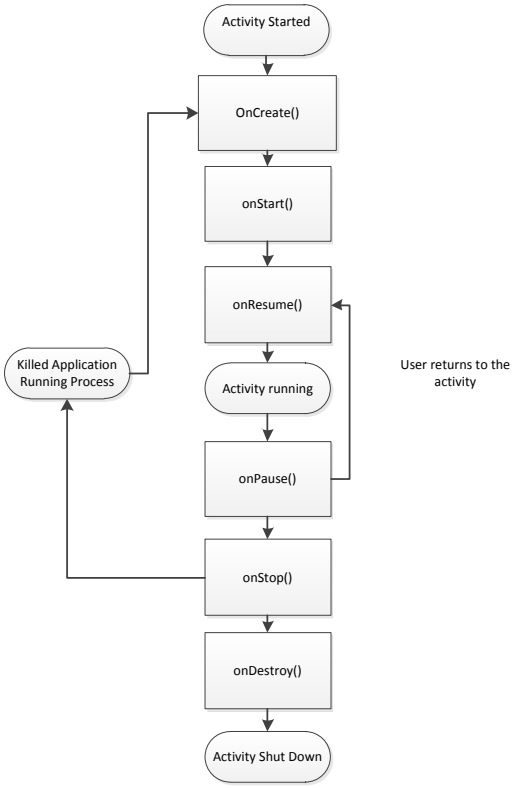


Figure 3.4 Activity Lifecycle [24]

Service also has a lifecycle, but it leads much simpler fashion than activity's. In general, an activity can start and stop a service. A service lifecycle can be represented as follows. For security reason, both service and broadcast which would be explained in the next paragraph receivers are needed to enforce permissions to access as shown in the *Mainfest.xml* file.

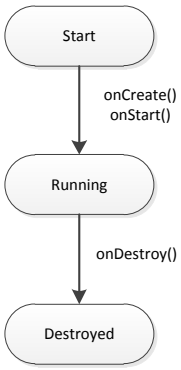


Figure3.5Lifecycle of Service

```
<receiver android:enabled="true" android:name =".mLocationReceiver"/>
 <service android:enabled="true" android:name=".mService" />
```

Broadcast receivers allow Android developers to register for system or application events. Registered receivers for that particular event will be notified during the Android runtime once the event happens. Its lifecycle could be explained through Figure 3.6.

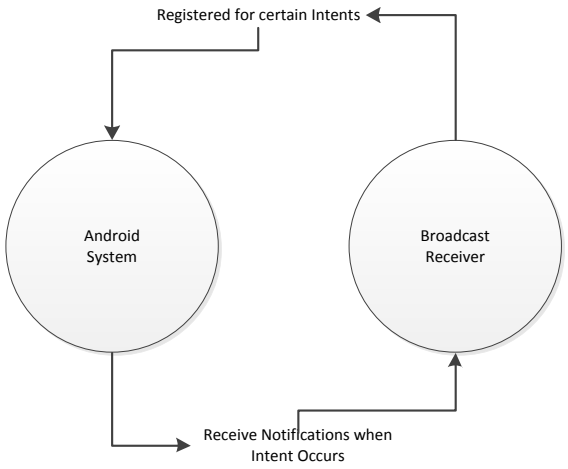
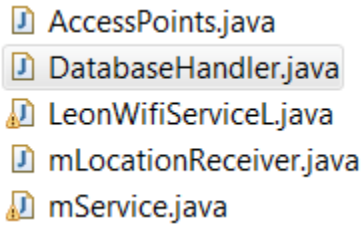


Figure3.6 Lifecycle of Broadcast Receivers

Finally content providers store and retrieve a structured set of data and make it accessible to applications. In most cases, this data is stored in a SQLite database. Content Providers provide a convenient way to share data information with other applications based on a structured data interface.

3.2 Android Application Architecture

We have developed 5 classes during the Android programming development.



- AccessPoint handles all initializing objects with constructors of the Wi-Fi information such as Service Set ID (ssid), Business Service Set ID (bssid) and signal strength. Ssid is a string that indicated as access point's name. Bssid is a 6 byte vendor assigned unique MAC address. It also sets the methods to set and retrieve all these Wi-Fi information.

- DatabaseHandlder takes care of database creation and upgrade. It executes basic SQL statements such as insertion, retrieval and updating.

- LeonWifiServiceL is the main activity piece that created simple button for users to start the service.

- mLocationReceiver contains the add-ons that sensing the location changes and store them into the database.

- mService is the main force to start the service running in the background and collect Wi-Fi information periodically and store them into the database file on the SD card.

In this Android development research, the application has involved most of the above components with different approaches. A flow diagram is presented in Figure 3.7 to demonstrate how the Android development flows from the start till the end. The application has one activity to integrate with service if it is not already running. The user can communicate with the service through a simple user interface that service exposes. The android application is developed using API 10 Gingerbread Android 2.33. Citing from some unofficial reports a few months ago, the majority of android mobile phone users are still using the older Android OS in the current market. Gingerbread has arrived during the peak time of Android development.

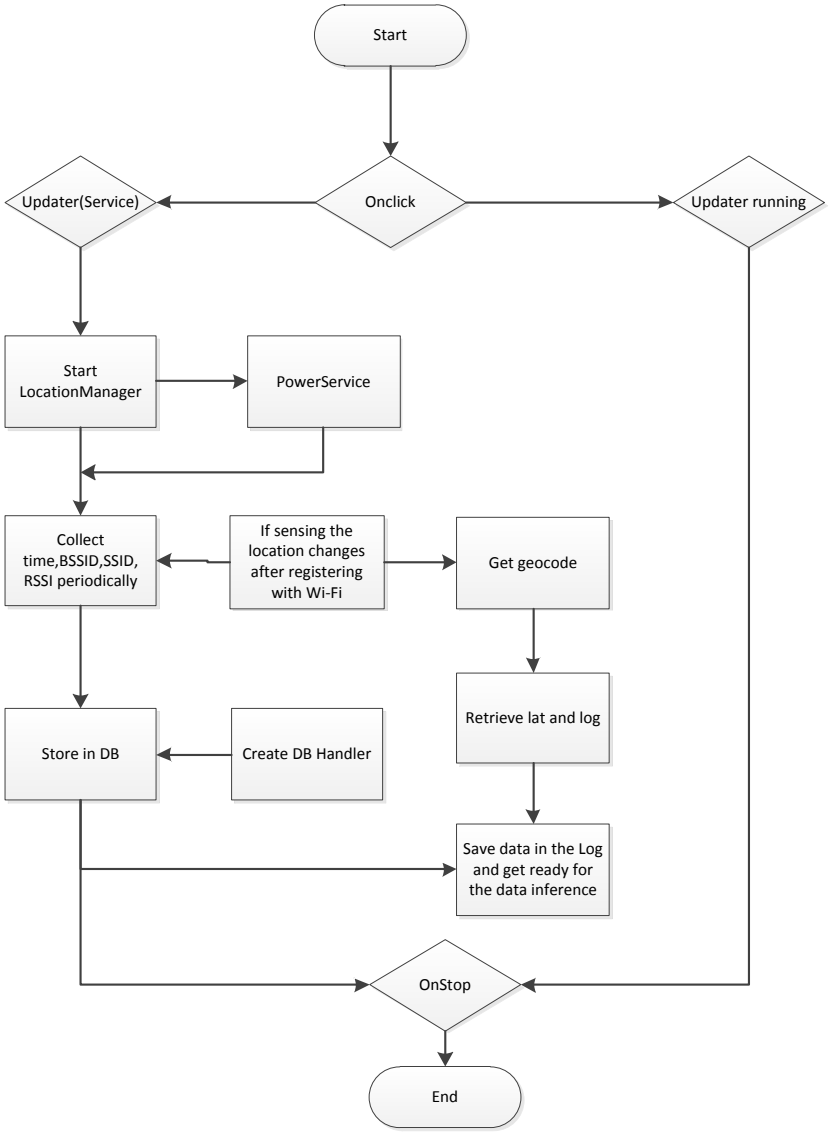


Figure3.7Flow Chat of Android Application

3.3 Detailed Approaches:

1. **Button creation in Activity.**

Most mobile applications could be developed with user interface. It could be very entertaining and enjoyable while users interact with the applications. Android provides an extensible input method framework that allows applications to provide users

alternative input methods, such as swipe, keyboards or even gesture and eye movement with current Samsung S4 platform.

There are a number of ways to create buttons for a user interface. Menus are the common user interface component in the majority of applications. It provides options such as menu and action bar. The user interface is for users to control the start and stop time. It used layout resource to simplify the button creation. The *setContentView* is used to set layout for simple button creation. The buttons can created from Widgets.

```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  //use layout main to set buttons
  setContentView(R.layout.main);
```

In the *main.xml* file, it can be even more customized with specified the button locations showing on the screen.

```
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" android:text="LeonWifi Service L" android:gravity="center" android:textSize="20sp" andro
<Button android:layout_width="wrap_content" android:layout_height="wrap_content" android:id="@+id/buttonStart" android:text="Sta
<Button android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Stop" android:id="@+id/buttonStop"
</LinearLayout>
```

The graphical layout showing on the mobile device is identical as it has been defined in the development stage.

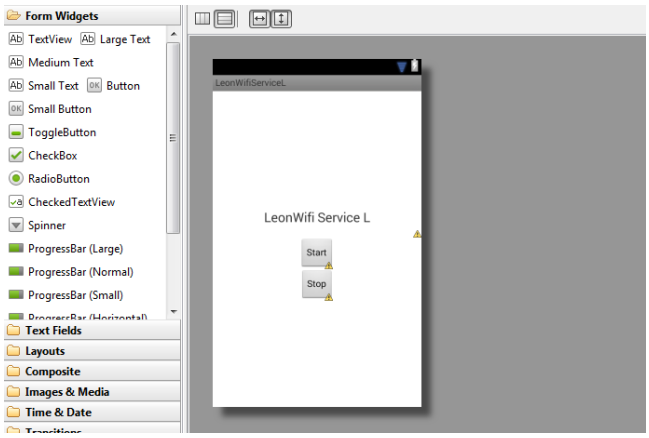


Figure3.8User Buttons for User Interface

2. **Service Based Application.**

The main goal of the first part of this research on mobile devices is to collect Wi-Fi information while travelling without any interruption. There are a number of ways to develop the location based applications reaching this goal. However, to be able better collect the useful information “quietly”, a method performing such a task could be running in the background while mobile users can still use their smart phones regularly such as playing games or making phone calls. The Service based programming method is adapted to meet this requirement.

There are a number of reasons why Service based programming is suitable for this research. First, the application should handle operations such as collecting the Wi-Fi information silently without a User Interface. Android accords services a higher priority than inactive or invisible Activities [21]. The nature of this research offers a perfect environment for service based application. Second, unlike activities which present a rich graphical interface, services run in the background, such as updating the Content

Providers, firing Intents and triggering Notifications. It would prolong battery life so that large amount of data could be collected. Service type of applications can run without a dedicated Graphical User Interface (GUI), and it still can execute in the main thread of the application process. Third, the Service is an application component representing either an application's desire to perform a longer-running operation while not interacting with the user or to supply functionality for other applications to use [5]. In this research, the application is required to run a number of hours for collecting enough data information for data inference. Therefore, the service based approach is best suited for our purpose.

A service runs by default in the main threads of the application. In this research, the custom service is initiated by calling *startService(new Intent(this. mService.class)* from the a defined Service Class, *mService*.

```
public void onClick(View src) {
  switch (src.getId()) {
  case R.id.buttonStart:
    Log.d(TAG, "onClick: starting srvice");
    startService(new Intent(this, mService.class));
    break;
  case R.id.buttonStop:
    Log.d(TAG, "onClick: stopping srvice");
    stopService(new Intent(this, mService.class));
    break;
  }
}
```

The service needs to be declared in the Manifest.xml file and implanting class must extend the Service class.

```
<service android:enabled="true" android:name=".mService" />
```

The following code shows how the custom service declaration and its implementation are done in this particular android development.

```
public class mService extends Service {
    private static final String TAG = "mService";
    WifiManager wifi;
    String sp,signal,Addr;
    //adding time
    String reportDate;
    public Boolean sendData = false;
```

Then the system will retrieve the *onCreate()* method, and then call its *onStart(Intent, int)* method with the arguments supplied by the client. At this point, service has started its life cycle from *onStart()* until the *onDestroy()* method is called. After field testing, the service based application can run as long as over 12 hours consistently without charging the battery on Galaxy S.

3. **Wakelock.**

In this research project, Galaxy S has been used as the primary mobile device. Samsung Galaxy S is used during the development period. From the default setting, the screen time out can only set to maximum 10 minutes. After that time of period, the phone will go to “sleep” mode unless it has been waked up by the users. However, it will introduce a challenge for the service based application collecting data, which cannot guarantee the process is running. After many field trips and testing, the data collection cannot be responded while the phone is in the sleep mode. The log data file indicated the gap of missing data starting from the phone went into sleep mode.

Because Android smart phone Galaxy S has the screen timeout that prevent the service base tracking application collecting data while the screen is blackout, therefore the power management mechanism must kick in to allow the service based application collecting Wi-Fi information. A wake lock mechanism is used to make sure the device stay on otherwise the service based application won't be able to collect the Wi-Fi information. This feature is extremely useful for Android OS 2.33 because there is no option to keep the screen from blackout which causes the phone goes into sleep mode.

```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  //to prevent the service goes to sleep or standby mode, still show a 4 minutes gap
  PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
  wl = pm.newWakeLock(PowerManager.FULL_WAKE_LOCK, "DoNjfdhotDimScreen");
  buttonStart = (Button) findViewById(R.id.buttonStart);
```

To be able to put the smart phone at awake mode, the wakelock should be put at *onResume*() during the visible lifetime of activity, not the entire lifetime of the activity.

```
@Override
protected void onResume() {
    super.onResume();
    //Call acquire() to acquire the wake lock and force the device to stay on
    //at the level that was requested when the wake lock was created.
    wl.acquire();
}
```

With Android OS version 4.2 Jelly Bean, the "stay awake" has just been introduced under the development options. However, in order to "stay awake", the phone is needed to be charging. Therefore it still didn't provide convenience to commuters.

4. **Simple thread.**

The main goal of this android application is to collect Wi-Fi information periodically. It means the service will pull data from surrounding Wi-Fi environment every once a while. How can the application perform such as a task periodically? This is a core Java development question. It could be timer, scheduler or even the Alarm Service to wake the device up to perform the certain task. In this research topic, a new simple thread has been chosen to pull data from Wi-Fi environment in order to focus on the need to interact with network. Static method of *Thread.sleep()* provided the solution to one of main objectives in this application – periodically polling the Wi-Fi data.

```
Thread.sleep(DELAY);
```

The new thread Updater has been created to handle data collection periodically with the *run()* method. As shown below, a flag was used to keep track on the thread is started or interrupted and synchronize data collection with the thread.

```
//Updater Thread
class Updater extends Thread{
    static final long DELAY = 10000; //10 seconds
    private boolean sendData = false;
    private Location mLastLocation;
    private Object mLocationReceiver;
    @Override
    public void run() {
        sendData = true;
        WifiManager myWifi = null;
        FileOutputStream fileOutputStream = null;
```

Therefore the synchronized is added in front of *onStart()* and *onDestroyed()* to improve the concurrency with fine-grained synchronization. The use of synchronized methods or

statements provides access to the implicit monitor lock associated with every object [22].

The Boolean type flag is used in different places to ensure the all operations on the same page.

```
@Override
public synchronized void onDestroy() {
    Toast.makeText(this, " mService Stopped", Toast.LENGTH_LONG).show();
    lm.removeUpdates(pendingIntent);
    super.onDestroy();
            //Stop the updater
    if (this.sendData) {
        updater.interrupt();

    }
    //Garbage collect
    updater = null;
    Log.d(TAG, "onDestroy");

}

@Override
public synchronized void onStart(Intent intent, int startid) {
    Toast.makeText(this, " mService Started", Toast.LENGTH_LONG).show();
    //Use the LocationManager class to obtain the location data...
    Intent i = new Intent(this, mLocationReceiver.class);
    pendingIntent = PendingIntent.getBroadcast(this, 0, i, PendingIntent.FLAG_UPDATE_CURRENT);
    updater = new Updater();
    //Start the updater
    if(! this.sendData){
        updater.start();
        this.sendData = true;
    }
    Log.d(TAG, "onStart");
}
```

5. **SQLite Database.**

Android platform gives users a few options to store data. One of most conventional way is using text files – either store in the application own file system directories or to external storage space on mobile devices such as SD cards. However, carrying out more complex operations on persistent data requires a more efficient way of management than flat text files. This is where the mobile database approach comes into the picture in this project.

Android system equips with SQLite which has been available on the platform so the application could manage its own private database. SQLite is a lightweight transactional database that comes with Android Operating System. Database engine occupies a small amount of disk storage and memory; therefore it has also been adopted by other mobile operating systems like IOS, windows 8 because of this feature.

To begin with SQLite, an extended class from SQLiteOpenHelper was created as using the following line of code

```
public class DatabaseHandler extends SQLiteOpenHelper {
```

SQLiteOpenHelper comes with a constructor and two methods, *onCreate()* and *onUpgrade().* As the constructor is instantiated, there are four important piece of data has to be provided: context, Database name, Cursor factory, and Database version.

```
public DatabaseHandler(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}
```

Context is useful to set this in the constructor and store it locally for later use in other methods. Database name is the file name that can be accessed from SQLite Manager later for data analysis. Database version is initially set to 1 conventionally, and it can be updated for later use.

OnCreate() method is called to create new database with a SQL creation script as follows,

```
// Creating Tables
@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_WIFI_TABLE = "CREATE TABLE " + TABLE_CONTACTS + "("
            + KEY_ID + " INTEGER PRIMARY KEY,"
            + KEY_TIME + " TEXT,"  //adding time
            + KEY_NAME + " TEXT,"
            + KEY_ADDR + " TEXT,"  //adding address
            + KEY_SIG_NO + " TEXT" + ")";
    db.execSQL(CREATE_WIFI_TABLE);
}
```

To update the database is shown below as the standard method to insert the data into the database in different columns.

```
// Adding new contact
void addContact(AccessPoints contact) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    //adding time
    values.put(KEY_TIME, contact.getTime()); //Contact time
    values.put(KEY_NAME, contact.getName()); // Contact Name
    values.put(KEY_ADDR, contact.getAddr()); //Contact Address
    values.put(KEY_SIG_NO, contact.getSignal()); // Contact Signal

    // Inserting Row
    db.insert(TABLE_CONTACTS, null, values);

    db.close(); // Closing database connection
}
```

In this research, database is used for storing information of time, SSID, BSSID and RSSI at this stage of development. Different fields or columns in this case were added to the database so the useful information is stored in the database for offline data analysis. As shown below, partial information is stored in the relational table structure.

id	time	name	addr	signal
1	Mon Dec 10 23:43:44 EST 2012	Rogers02303	e8:40:f2:6b:f4:e7	-49
2	Mon Dec 10 23:43:46 EST 2012	Rogers02303	e8:40:f2:6b:f4:e7	-46
3	Mon Dec 10 23:43:46 EST 2012	city17	00:26:82:a5:cb:97	-93
4	Mon Dec 10 23:43:47 EST 2012	Rogers02303	e8:40:f2:6b:f4:e7	-47
5	Mon Dec 10 23:43:47 EST 2012	city17	00:26:82:a5:cb:97	-93
6	Mon Dec 10 23:43:48 EST 2012	Rogers02303	e8:40:f2:6b:f4:e7	-47
7	Mon Dec 10 23:43:48 EST 2012	city17	00:26:82:a5:cb:97	-92
8	Mon Dec 10 23:43:49 EST 2012	Rogers02303	e8:40:f2:6b:f4:e7	-46
9	Mon Dec 10 23:43:50 EST 2012	Rogers02303	e8:40:f2:6b:f4:e7	-46
10	Mon Dec 10 23:43:51 EST 2012	Rogers02303	e8:40:f2:6b:f4:e7	-46
11	Mon Dec 10 23:43:52 EST 2012	Rogers02303	e8:40:f2:6b:f4:e7	-47
12	Mon Dec 10 23:43:53 EST 2012	Rogers02303	e8:40:f2:6b:f4:e7	-47
13	Mon Dec 10 23:43:54 EST 2012	Rogers02303	e8:40:f2:6b:f4:e7	-47
14	Mon Dec 10 23:43:55 EST 2012	Rogers02303	e8:40:f2:6b:f4:e7	-48
15	Mon Dec 10 23:43:55 EST 2012	city17	00:26:82:a5:cb:97	-92
16	Mon Dec 10 23:43:55 EST 2012	Sharpe	64:0f:28:09:d9:79	-96
17	Mon Dec 10 23:43:55 EST 2012	BELL682	00:25:3c:c4:27:11	-94
18	Mon Dec 10 23:43:57 EST 2012	Rogers02303	e8:40:f2:6b:f4:e7	-47
19	Mon Dec 10 23:43:57 EST 2012	city17	00:26:82:a5:cb:97	-92
20	Mon Dec 10 23:43:58 EST 2012	Rogers02303	e8:40:f2:6b:f4:e7	-46

Figure3.9Raw Data Collection

6. **Data collection.**

Archiving the data is one of the crucial steps for data analysis for this project. It is the bridge that is closely linked to application performance. There are few ways to store the data, therefore for the data logging technique is applied to store the data or get the database file to retrieve the table information.

There are two methods to obtain the database information from the device .First, like iPhone OS, Android stores these databases in an area where is only accessible by root users. To access the caches, an Android device needs to be "rooted," which removes most of the system's security features. Unlike iPhone OS, though, Android phones aren't typically synced with a computer, so the files would need to be extracted from a rooted device directly.

This distinction makes the data harder to access for the average users, but easy enough for an experienced developer or forensic expert. The information can be retrieved from various android forums and developer websites.

Another way to retrieve the data is using the string to store the database information then save the file into certain folder on the mobile device. *geAllContact()* basically perform a SQL command “select * from table” to retrieve all information from the table WifiInfoL. Then a FileOutputStream was created for the file object handling the data information to store the data on the SD card where it can be easily accessed.

```
try {
    fileOutputStream = new FileOutputStream(String.format("sdcard/tmp/%s LeonWifi.txt", System.currentTimeMillis()));
} catch (FileNotFoundException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
```

Then retrieve all database information into the text files.

```
db.addContact(new AccessPoints(i,reportDate, sp, Addr,signal));

//Reading all record
Log.d("Reading: ", "Reading all contacts..");
List<AccessPoints> contacts = db.getAllContacts();

for (AccessPoints ap : contacts) {
    //String log = "Id: "+ap.getID()+"  ,Time: " + ap.getTime() +"  ,Address: " + ap.getAddr() +"
    String log = ap.getID()+"|" + ap.getTime() +"|" + ap.getAddr() +"|" + ap.getName() +"|" + ap.g
    byte[] LeonLog = log.getBytes();
    try {
     // Writing Contacts to log
        fileOutputStream.write(LeonLog);
        Log.d("Name: ", log);
    }
         catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

}   //end of for loop AccessPoints ap
```

Using data type byte to get all bytes of a UTF-8 encoded text file into a byte array, then store the data file into tmp folder on SD card. It is the plain text file without any

database features. It can be easily translate back to Database file format as needed at the data analysis stage.

7. **Location Sensing**.

This portion of the development is a necessary step for the verification purpose for the offline data analysis. When mobile users move to different locations, the application can sense new location in order to reflect the moving direction. There is one condition applied in this scenario - the mobile device has to register with trusted and open Wi-Fi network.

Listening for location changes is a costly affair, it can consume more battery usage. The heat dissipation on the back of the mobile devices could be noticeable when the listening update is running intensively. Therefore, rather than using the class of LocationListener constantly to obtain the location information, the method of using *pendingIntent* object with a BroadcastReceiver class is added to keep track on the changing location information such as latitude and longitude obtained from registered Wi-Fi connection.

```
public synchronized void onStart(Intent intent, int startid) {
    Toast.makeText(this, " mService Started", Toast.LENGTH_LONG).show();
    //Use the LocationManager class to obtain the location data...
    Intent i = new Intent(this, mLocationReceiver.class);
    pendingIntent = PendingIntent.getBroadcast(this, 0, i, PendingIntent.FLAG_UPDATE_CURRENT);
```

The *PendingIntent* allows the mLocationReceiver class to use the application permission to execute the defined code as follows in the mLocationReceiver class. In this case, it used the LocationManager to obtain the location data.

```
//Updater Thread
class Updater extends Thread{
    static final long DELAY = 10000; //10 seconds
    private boolean sendData = false;
    private Location mLastLocation;
    private Object mLocationReceiver;
    @Override
    public void run() {
        sendData = true;
        WifiManager myWifi = null;
        FileOutputStream fileOutputStream = null;

        try {
            fileOutputStream = new FileOutputStream(String.format("sdcard/tmp/%s LeonWifi.txt", System.currentTimeMillis())
        } catch (FileNotFoundException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        //Use the LocationManager class to obtain the location data...
        lm = (LocationManager)getSystemService(Context.LOCATION_SERVICE);

        lm.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 20000, 0, pendingIntent);
```

In BroadcastReceiver class, the location information is stored in the intent argument, and this information can be identified as follows. Then the context can be added to the database as known address for later data analysis.

```
public void onReceive(Context context, Intent intent) {
    // TODO Auto-generated method stub
    String locationKey = LocationManager.KEY_LOCATION_CHANGED;
    String providerEnableKey = LocationManager.KEY_PROVIDER_ENABLED;
    if(intent.hasExtra(providerEnableKey))
    {
        if(!intent.getBooleanExtra(providerEnableKey, true)){
            Toast.makeText(context, "Provider Disabled", Toast.LENGTH_SHORT).show();
        }
        else {
            Toast.makeText(context, "Provider Enabled", Toast.LENGTH_SHORT).show();
        }

    }
    //to check the locationKey has changed
    if(intent.hasExtra(locationKey)){
        Location loc = (Location)intent.getExtras().get(locationKey);

        Geocoder geocoder = new Geocoder(context, Locale.getDefault());
```

It certainly must be declared in the manifest.xml file for Broadcast Receiver for the permission.

```
<receiver android:enabled="true" android:name =".mLocationReceiver"/>
```

Because it is still part of the service routine, the Location Manager and *pendingIntent* object should be at the beginning of data collection.

```
//Use the LocationManager class to obtain the location data...
lm = (LocationManager)getSystemService(Context.LOCATION_SERVICE);

lm.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 20000, 0, pendingIntent);
//Intent i = new Intent(this,this.mLocationReceiver.class);
if(myWifi==null)
{
    myWifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);
}
while (sendData) {
    //put try and catch because it can be interrupted
    try{
        Log.d(TAG, "Updater running");
        //Date now = null;
```

The *PendingIntent* Object is passed when there is a change in location, the *onReceive()* method is triggered. Also it should be removed after the location sensing is done.

```
@Override
public synchronized void onDestroy() {
    Toast.makeText(this, " mService Stopped", Toast.LENGTH_LONG).show();
    lm.removeUpdates(pendingIntent);
    super.onDestroy();
            //Stop the updater
    if (this.sendData) {
        updater.interrupt();
    }
    //Garbage collect
    updater = null;
    Log.d(TAG, "onDestroy");

}
```

The Toast Class is used to display the location information to the end users when the location change takes place.

8. **Location Report**.

In order to verify the physical location from collected data, the location Geocode is introduced to interpret latitude and longitude information into address information.

```
,
//to check the locationKey has changed
if(intent.hasExtra(locationKey)){
    Location loc = (Location)intent.getExtras().get(locationKey);
    Geocoder geocoder = new Geocoder(context, Locale.getDefault());
```

Android provides the class of Geocoder to transform the information of latitude and longitude into the street address. The purpose of this portion of the coding is to assist the data analysis at the second stage of the research. The following coding shows how longitude, latitude and street address can be retrieved and stored back to the existing database.

```
 try {
    addresses = geocoder.getFromLocation(loc.getLatitude(), loc.getLongitude(), 1);
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

   if (addresses != null && addresses.size() > 0) {
       Address address = addresses.get(0);
       // Format the first line of address (if available), city, and country name.
       //String addressText = String.format("%s, %s, %s",
                Address4 = String.format("%s, %s, %s",
               address.getMaxAddressLineIndex() > 0 ? address.getAddressLine(0) : "",
               address.getLocality(),
               address.getCountryName());

   }

//Puting time, lat, log, addr in the database

Double lat = loc.getLatitude();
Double lng = loc.getLongitude();
Date now = new Date();
//reportDate = now.toString();
Address1 = now.toString();
Address2 = Double.toString(lat);
Address3 = Double.toString(lng);
////////////////////////////////////////////////////////////
Toast.makeText(context, "Location Changed : Lat: " + loc.getLatitude()+ " Lng: "+loc.getLongitude(),
        Toast.LENGTH_SHORT).show();
DatabaseHandler db = new DatabaseHandler(context);
db.addContact(new AccessPoints(Address1, Address2,Address3,Address4));
```

Chapter 4 Data Inference

Data Inference is second stage of this thesis project. After collecting all the Wi-Fi information including ssid, bssid and rssi, the data can be analyzed offline. The purpose of the data inference is to plot the history path of the corresponding nodes and connect the Wi-Fi logical space to physical space which is the known addresses.

Using the system Command in Windows 7 shown below can display the surrounding or neighbouring wireless access points

>netsh wlan show networks mode=bssid

A screen shot can be retrieve as shown below after executing the dos command at any given on UOIT campus.

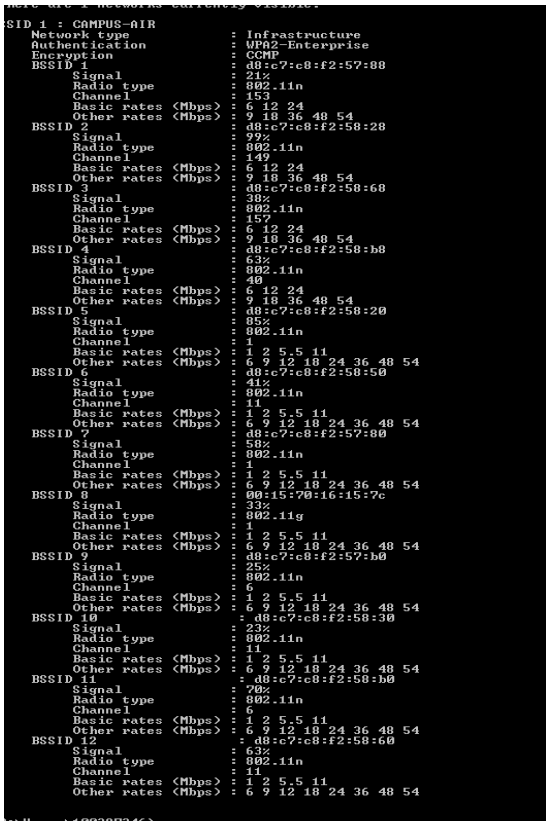


Figure4.10 Network Scan for Bssids

Inference, or model evaluation, can be described as the process of updating probabilities of outcomes based upon the relationships in the model and the evidence known about the situation at hand. A probability can be mistaken, but there is little ambiguity about what it means, or how evidence should be combined to calculate it.

4.1 Wi-Fi location of Data inference:

As discussed earlier, there are two ways of polling the database information from the mobile device. One is to register as the super users to the mobile device, and then the super user would easily reuse the database information or translate it into text format. The second approach is the Java implementation at application level to store the database information as

text format on the SD card of the mobile device. This research is opted for both options as the mobile application development has gone to some levels in depth.

In order to select meaningful data and then interpret them as different nodes, the techniques of grouping and calculation of similarity between nodes are used. The following diagram can be referred as the flow chart for data manipulation from the data collection. There are steps that could be key points to determine the readings from the data file.

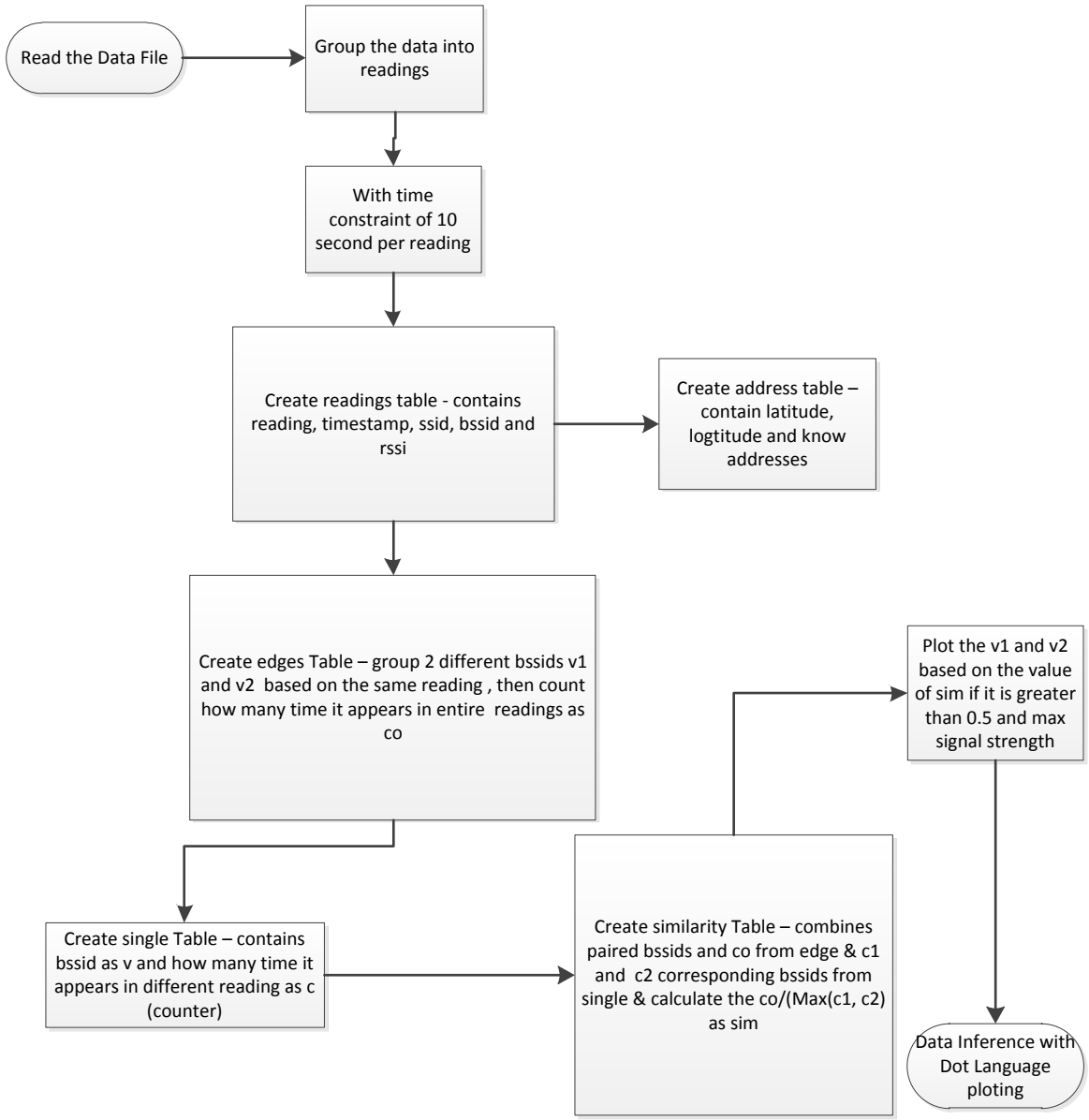


Figure 4.2 Flowchart for Data Analysis

The programming language used for data inference is Python using DB-API 2.0 interface with SQLite Database. There are 5 tables created in the database from the raw data for the data inference. The future step can be developed to the stage of regrouping the database at runtime.

Raw data collection file has one large table in the database. It contains five columns id, time, ssid, bssid and rssi. In order to efficiently use “raw” information from the database, the data inference is used at this stage to manage the data collection.

A new database is generated for easy access data inference with the manipulation and calculation based on time stamps. The algorithm of these tables could be explained in a nutshell. The readings table is created first from the raw data that contains id, timestamp, ssid, bssid and rssi. The algorithm of creating the readings table is to group the data into different blocks based every 10 seconds using the timestamp. Next, the edge table and single table are generated based on readings table to select 2 unique access points and count them both and individually how many times they appear in the entire readings table. Similarity table is the key table for calculating how similar these two access points in the entire readings table. Android application is capable of obtaining physical address after registering with Wi-Fi connections; therefore the useful physical address table is also created from the raw data. In order to connect the Wi-Fi logic distances to physical known addresses, the heuristics of distance and multi-hop are explained in greater details in the subsections.

1. **Readings Table**

The following shows partial one of the original data in the text format. It contains id, time, ssid, bssid and signal strength in row.

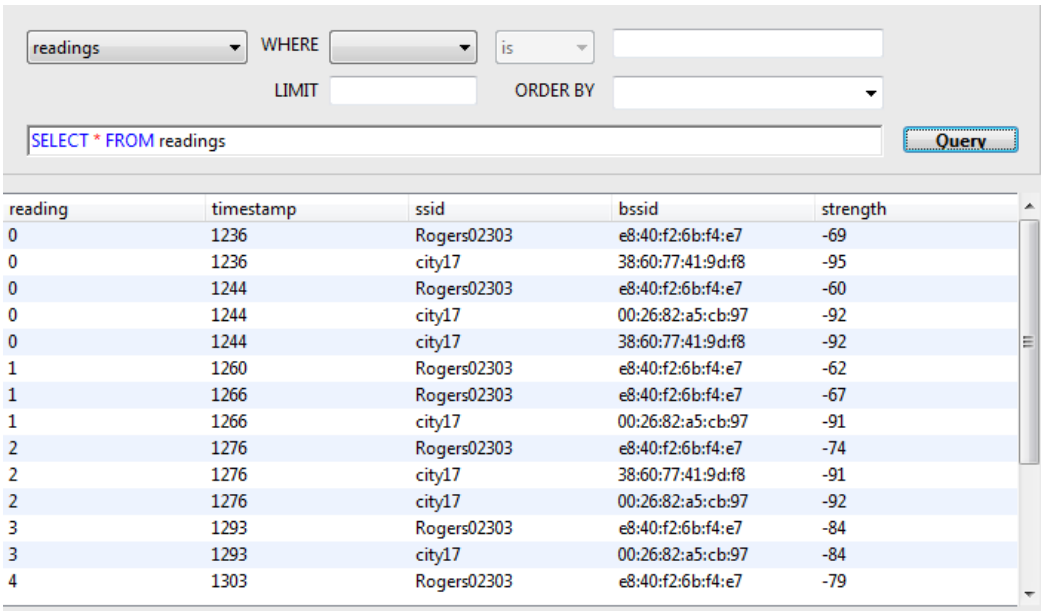
```
586|Sun Sep 01 12:11:04 EDT 2013|CAMPUS-AIR|d8:c7:c8:e2:e3:e0|-71
587|Sun Sep 01 12:11:05 EDT 2013|CAMPUS-AIR|d8:c7:c8:e2:e3:e0|-77
588|Sun Sep 01 12:11:05 EDT 2013|CAMPUS-AIR|00:15:70:4d:bd:34|-82
589|Sun Sep 01 12:11:07 EDT 2013|CAMPUS-AIR|d8:c7:c8:e2:e3:e0|-76
590|Sun Sep 01 12:11:07 EDT 2013|CAMPUS-AIR|d8:c7:c8:e2:dc:70|-80
591|Sun Sep 01 12:11:07 EDT 2013|CAMPUS-AIR|00:15:70:4d:af:0c|-88
592|Sun Sep 01 12:11:10 EDT 2013|CAMPUS-AIR|d8:c7:c8:e2:e3:e0|-77
593|Sun Sep 01 12:11:10 EDT 2013|CAMPUS-AIR|d8:c7:c8:e2:dc:70|-86
594|Sun Sep 01 12:11:10 EDT 2013|CAMPUS-AIR|00:15:70:4d:bd:34|-91
```

Because the Android application is taking the Wi-Fi information every second periodically, it is more logical to regroup them every 10 seconds as one reading to better understand the data allocation. The date time format has been converted to integer as shown below

```
timestamp = datetime.strptime(timestamp, "%a %b %d %H:%M:%S EST %Y")
timestamp = timestamp.hour * 24 + timestamp.minute * 60 + timestamp.second
```

```
if lastTimestamp == None or timestamp - lastTimestamp < 10:
    readingGroup.append(row)
```

Therefore, instead of hundreds, thousands of the sequential data information appending on each time stamps with one second apart, the readings block table could be used systematically as the first step to group the data every 10 seconds per reading in the readings table in the new database file. The following is what the readings table looks like after regrouping data according to the time constraint.

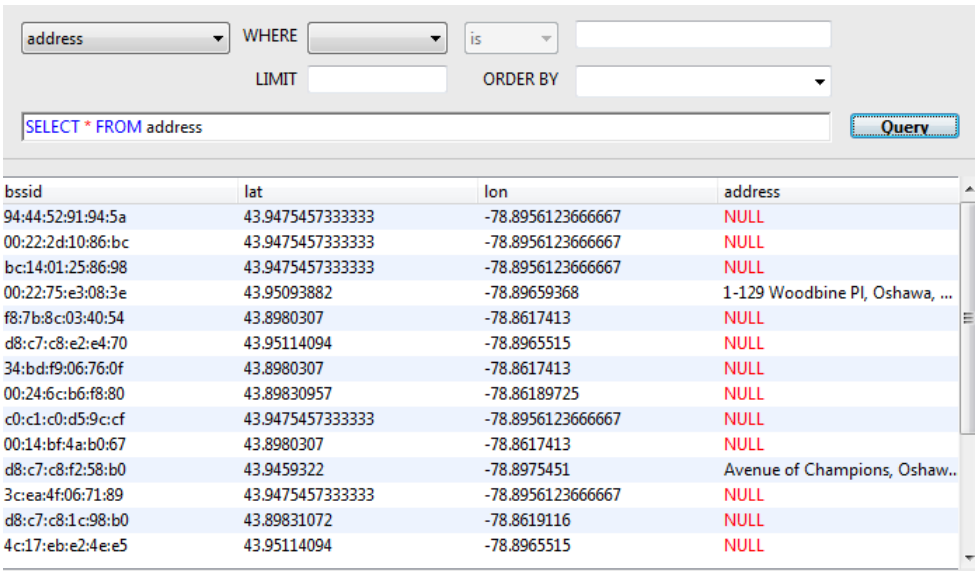


reading	timestamp	ssid	bssid	strength
0	1236	Rogers02303	e8:40:f2:6b:f4:e7	-69
0	1236	city17	38:60:77:41:9d:f8	-95
0	1244	Rogers02303	e8:40:f2:6b:f4:e7	-60
0	1244	city17	00:26:82:a5:cb:97	-92
0	1244	city17	38:60:77:41:9d:f8	-92
1	1260	Rogers02303	e8:40:f2:6b:f4:e7	-62
1	1266	Rogers02303	e8:40:f2:6b:f4:e7	-67
1	1266	city17	00:26:82:a5:cb:97	-91
2	1276	Rogers02303	e8:40:f2:6b:f4:e7	-74
2	1276	city17	38:60:77:41:9d:f8	-91
2	1276	city17	00:26:82:a5:cb:97	-92
3	1293	Rogers02303	e8:40:f2:6b:f4:e7	-84
3	1293	city17	00:26:82:a5:cb:97	-84
4	1303	Rogers02303	e8:40:f2:6b:f4:e7	-79

Figure 4.3Readings Table

2. Address Table

The Address table contains information of bssid and corresponding longitude, latitude and physical address, converted from the Android application when the mobile device is registered with the Wi-Fi connection. It could be useful for data inference especially for data retrieval between Wi-Fi space and physical space and then the data plotting.

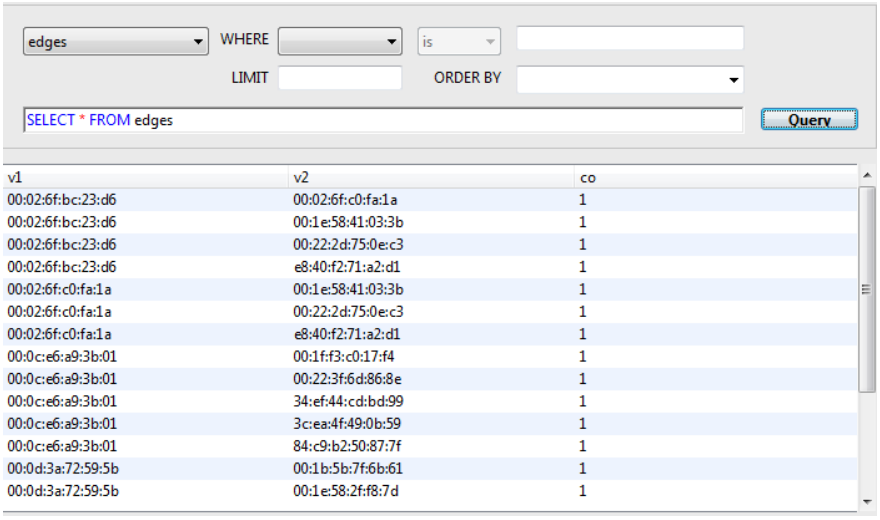


bssid	lat	lon	address
94:44:52:91:94:5a	43.9475457333333	-78.8956123666667	NULL
00:22:2d:10:86:bc	43.9475457333333	-78.8956123666667	NULL
bc:14:01:25:86:98	43.9475457333333	-78.8956123666667	NULL
00:22:75:e3:08:3e	43.95093882	-78.89659368	1-129 Woodbine Pl, Oshawa, ...
f8:7b:8c:03:40:54	43.8980307	-78.8617413	NULL
d8:c7:c8:e2:e4:70	43.95114094	-78.8965515	NULL
34:bd:f9:06:76:0f	43.8980307	-78.8617413	NULL
00:24:6c:b6:f8:80	43.89830957	-78.86189725	NULL
c0:c1:c0:d5:9c:cf	43.9475457333333	-78.8956123666667	NULL
00:14:bf:4a:b0:67	43.8980307	-78.8617413	NULL
d8:c7:c8:f2:58:b0	43.9459322	-78.8975451	Avenue of Champions, Oshaw..
3c:ea:4f:06:71:89	43.9475457333333	-78.8956123666667	NULL
d8:c7:c8:1c:98:b0	43.89831072	-78.8619116	NULL
4c:17:eb:e2:4e:e5	43.95114094	-78.8965515	NULL

Figure 4.4Address Table

3. Edge Table

In order to group a pair of different bssids as nodes for data inference especially for data plotting, the edge table was created for this purpose. The edge table contains a pair of different bssids referred as v1 and v2 co-occurring in the same reading and then count how many times they co-exist in entire readings as co. Therefore, the same reading of 2 different bssids is paired in one row joining with the number of times they appear from different readings.

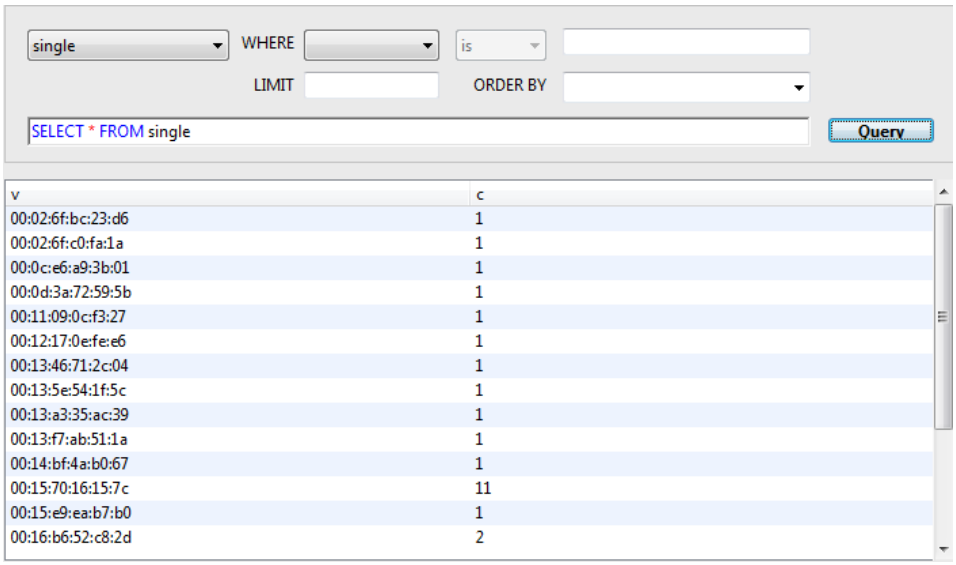


v1	v2	co
00:02:6f:bc:23:d6	00:02:6f:c0:fa:1a	1
00:02:6f:bc:23:d6	00:1e:58:41:03:3b	1
00:02:6f:bc:23:d6	00:22:2d:75:0e:c3	1
00:02:6f:bc:23:d6	e8:40:f2:71:a2:d1	1
00:02:6f:c0:fa:1a	00:1e:58:41:03:3b	1
00:02:6f:c0:fa:1a	00:22:2d:75:0e:c3	1
00:02:6f:c0:fa:1a	e8:40:f2:71:a2:d1	1
00:0c:e6:a9:3b:01	00:1f:f3:c0:17:f4	1
00:0c:e6:a9:3b:01	00:22:3f:6d:86:8e	1
00:0c:e6:a9:3b:01	34:ef:44:cd:bd:99	1
00:0c:e6:a9:3b:01	3c:ea:4f:49:0b:59	1
00:0c:e6:a9:3b:01	84:c9:b2:50:87:7f	1
00:0d:3a:72:59:5b	00:1b:5b:7f:6b:61	1
00:0d:3a:72:59:5b	00:1e:58:2f:f8:7d	1

Figure 4.5Edge Table

4. Single Table

The reason table single was named is to eliminate the redundancy of a bssid appear multiple times in the entire readings. Single table contains each bssid as v in readings and counter as c that counts how many time v appears in different reading in entire readings table. It collects statistical information of each individual bssid.



v	c
00:02:6f:bc:23:d6	1
00:02:6f:c0:fa:1a	1
00:0c:e6:a9:3b:01	1
00:0d:3a:72:59:5b	1
00:11:09:0c:f3:27	1
00:12:17:0e:fe:e6	1
00:13:46:71:2c:04	1
00:13:5e:54:1f:5c	1
00:13:a3:35:ac:39	1
00:13:f7:ab:51:1a	1
00:14:bf:4a:b0:67	1
00:15:70:16:15:7c	11
00:15:e9:ea:b7:b0	1
00:16:b6:52:c8:2d	2

Figure 4.6 Single Table

5. Similarity Table

The Similarity table combines edge and single tables and a few other additional important heuristics calculations and definition columns. The first three columns are the pair of bssids as v1 and v2 and co-occurrence counter from edge table. The fourth and fifth columns are the counters of c1 and c2 counters that count how many corresponding v1 and v2 appear in reading in the entire readings table. The sixth column is the similarity heuristics calculation as sim about the paired v1 and v2. The seventh column is the Wi-Fi space distance heuristic calculation as dist between v1 and v2.

1) Similarity Heuristics Calculation

   This portion is mainly focus on the calculation of computing a measure between 0 and 1 to determine how similar v1 and v2 are with respect to the readings. If they appear together all the time, then they are very similar (sim = 1), but if they never appear together, then they are not similar at all, so sim=0. The threshold is set to 0.5 to increase the clarity of the plotting. The formula to achieves this goal use as (co)/max(c1,c2).

2) Wi-Fi Space Distance Heuristic Calculation

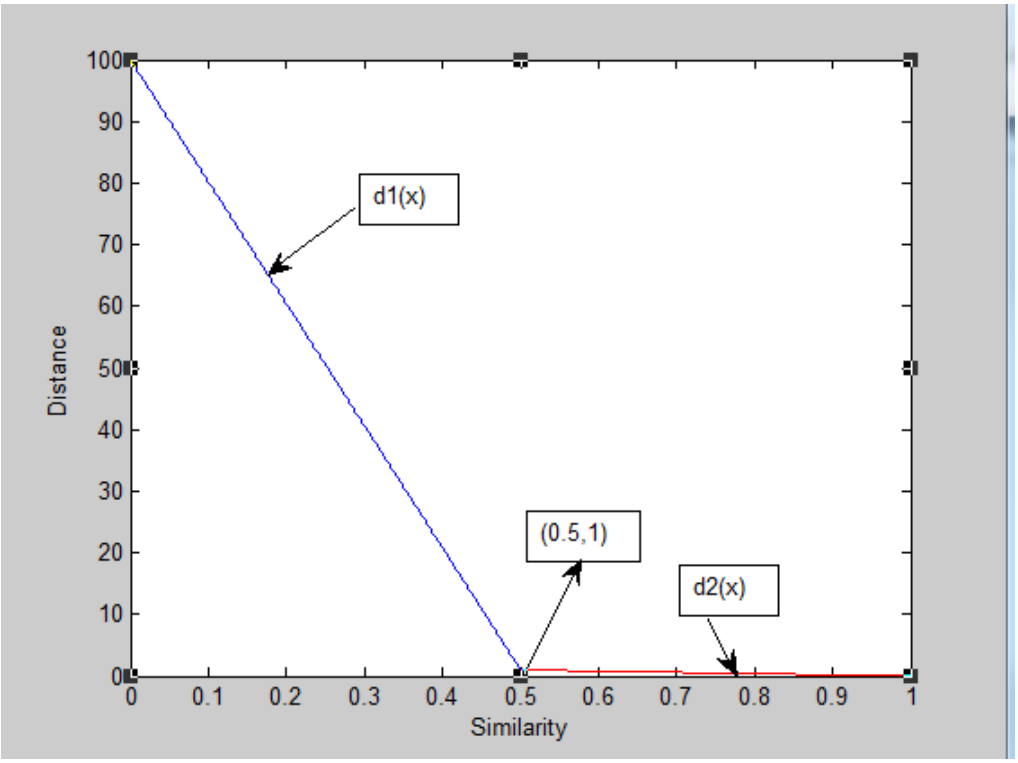


Figure 4.7 Distance Heuristic Diagram

The node distance calculation is done based on similarity calculation. The distance d1 can be presented as

$d1(x) = (1-\alpha_1 X)*K1$ where K1 is user define value as 100 refers as far away (1)

$d2(x) = (1-\alpha_2 X)*K2$ (2)

where d1 and d2 are distance between two nodes v1 and v2 and X is the similarity between the two nodes v1 and v2.

Substitute K1= 100 and X = 0.5 into equation (1),

$d1(X) = 100(1- \alpha_1) = 1$, so α_1 can be easily obtained 99/50, therefore

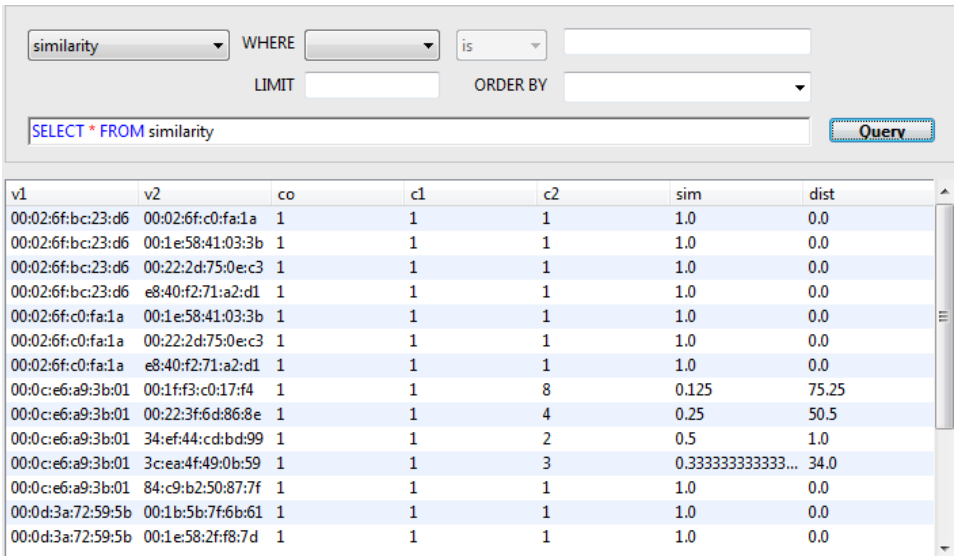
$d1(X) = 100*(1-99/50 * X)$ (3)

Similar technique can applied to obtain α_2 using the graph shown above

$1 = (1-\alpha_2 X)*K2$ and $0 = (1- \alpha_2)* K2$

Therefore K2 and α_2 can be obtained as 2 and 1 respectively.

$d2(x) = 2*(1-X)$ (4)



v1	v2	co	c1	c2	sim	dist
00:02:6f:bc:23:d6	00:02:6f:c0:fa:1a	1	1	1	1.0	0.0
00:02:6f:bc:23:d6	00:1e:58:41:03:3b	1	1	1	1.0	0.0
00:02:6f:bc:23:d6	00:22:2d:75:0e:c3	1	1	1	1.0	0.0
00:02:6f:bc:23:d6	e8:40:f2:71:a2:d1	1	1	1	1.0	0.0
00:02:6f:c0:fa:1a	00:1e:58:41:03:3b	1	1	1	1.0	0.0
00:02:6f:c0:fa:1a	00:22:2d:75:0e:c3	1	1	1	1.0	0.0
00:02:6f:c0:fa:1a	e8:40:f2:71:a2:d1	1	1	1	1.0	0.0
00:0c:e6:a9:3b:01	00:1f:f3:c0:17:f4	1	1	8	0.125	75.25
00:0c:e6:a9:3b:01	00:22:3f:6d:86:8e	1	1	4	0.25	50.5
00:0c:e6:a9:3b:01	34:ef:44:cd:bd:99	1	1	2	0.5	1.0
00:0c:e6:a9:3b:01	3c:ea:4f:49:0b:59	1	1	3	0.333333333333...	34.0
00:0c:e6:a9:3b:01	84:c9:b2:50:87:7f	1	1	1	1.0	0.0
00:0d:3a:72:59:5b	00:1b:5b:7f:6b:61	1	1	1	1.0	0.0
00:0d:3a:72:59:5b	00:1e:58:2f:f8:7d	1	1	1	1.0	0.0

Figure 4.8 Similarity Table

The definition on similarity and calculation of “distance” between 2 nodes can be obtained from equations (1),(2),(3), and (4). The K1 value could be chosen differently, but the plotting results remain the same because the history path is not based on the distance between nodes. The logical distance is built on to show users how far from the position to a known address. With more extensive data collection, the real distance values could be introduced in the future development stage.

According to the calculation of the similarity between 2 nodes, it seems that the limit of similarity can be set to 0.5 so that the plotting can be fine-tuned by filtering out some noise.

The Wi-Fi distance calculation can be connected to physical space by building the multi-hop distance index as the following diagram showing 2 clusters of nodes with user define distances.

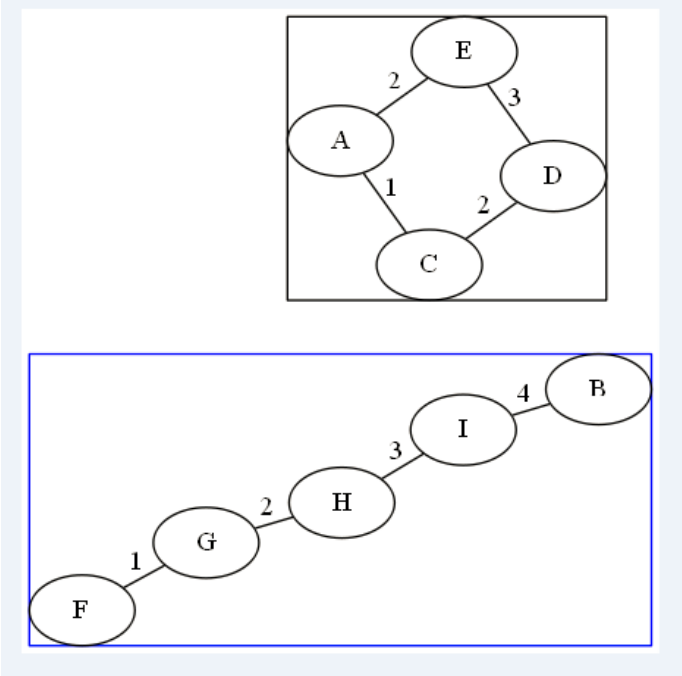


Figure 4.9 Multiple Hop Index Diagram

Starting table D1, it transfers the Wi-Fi space information into physical address space!

D1 table includes distance between adjacent hops. D1 jointed both Similarity table and Address table. v1 and v2 are selected from the similarity table with the condition of v2 matches bssid showing in the address table. The dist presented as distance between the pair retrieved from the similarity table. This mechanism presents the distance between v1 and v2 with a known address.

From Figures of D1, D2, D3 and D4, tables are developed based on multiple hop heuristics table. Assume A and B contain the physical addresses and they are used as the reference points for multi-hop development heuristic. The distances in table D1 as shown below are recorded for adjacent nodes. For example, the distance between A and E is 2, but the distance between D and A can be recorded because D is not directly connected to A.

Node	Node	Distance
C	A	1
E	A	2
D	-	-
F	-	-
G	-	-
H	-	-
I	B	4

Table 4.2 D1 simplified version

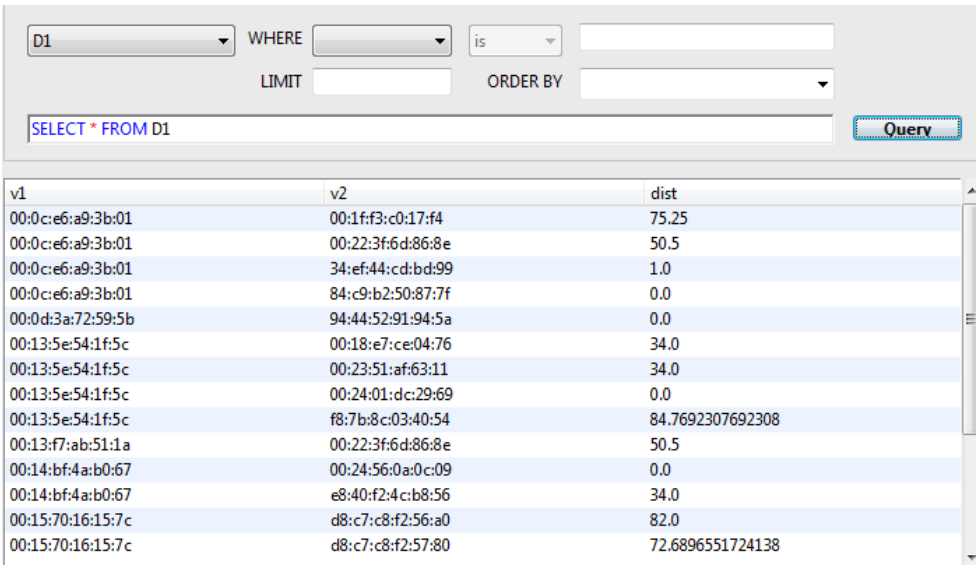
Building on table D2 includes the distance between 2 nodes that are connected indirectly with one node in between. It takes v1 from Similarity table and v2 from D1 with the condition of v2 matches v1 from D1 table. The dist represents the minimum distance between these v1 and v2. It can be explained as the following simplified table referring Figure 4.9 two clusters. The values

of distance remain the same as D1 for adjacent nodes. For example, the distance between C and A is still 1, but the distance between D and A is recorded as 3 between it is the minimum distance between them via the route of D->C->A. The distance between H and B is recorded as 7 in the second cluster. The logic of developing tables of D3 and D4 are the same as D2 by extending the minimum distances between two nodes via 2 and 3 nodes respectively. When there are 3 nodes in between 2 nodes, the signal would become very weak therefore there is no need to continue to calculate further.

Hop	Hop	Distance
A	A	0
B	B	0
C	A	1
D	A	3
E	A	1
F	-	-
G	-	-
H	B	7
I	-	-

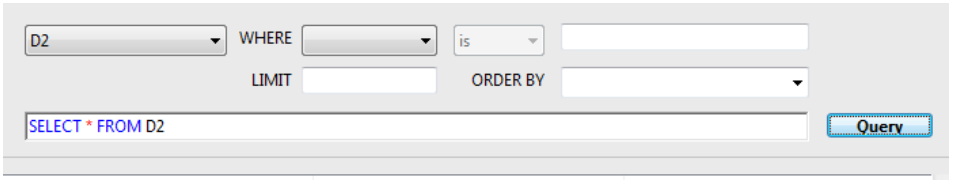
Table 4.3 D2 simplified version

All D1, D2, D3 and D4 are represented in the similar format as shown below.



v1	v2	dist
00:0c:e6:a9:3b:01	00:1f:f3:c0:17:f4	75.25
00:0c:e6:a9:3b:01	00:22:3f:6d:86:8e	50.5
00:0c:e6:a9:3b:01	34:ef:44:cd:bd:99	1.0
00:0c:e6:a9:3b:01	84:c9:b2:50:87:7f	0.0
00:0d:3a:72:59:5b	94:44:52:91:94:5a	0.0
00:13:5e:54:1f:5c	00:18:e7:ce:04:76	34.0
00:13:5e:54:1f:5c	00:23:51:af:63:11	34.0
00:13:5e:54:1f:5c	00:24:01:dc:29:69	0.0
00:13:5e:54:1f:5c	f8:7b:8c:03:40:54	84.7692307692308
00:13:f7:ab:51:1a	00:22:3f:6d:86:8e	50.5
00:14:bf:4a:b0:67	00:24:56:0a:0c:09	0.0
00:14:bf:4a:b0:67	e8:40:f2:4c:b8:56	34.0
00:15:70:16:15:7c	d8:c7:c8:f2:56:a0	82.0
00:15:70:16:15:7c	d8:c7:c8:f2:57:80	72.6896551724138

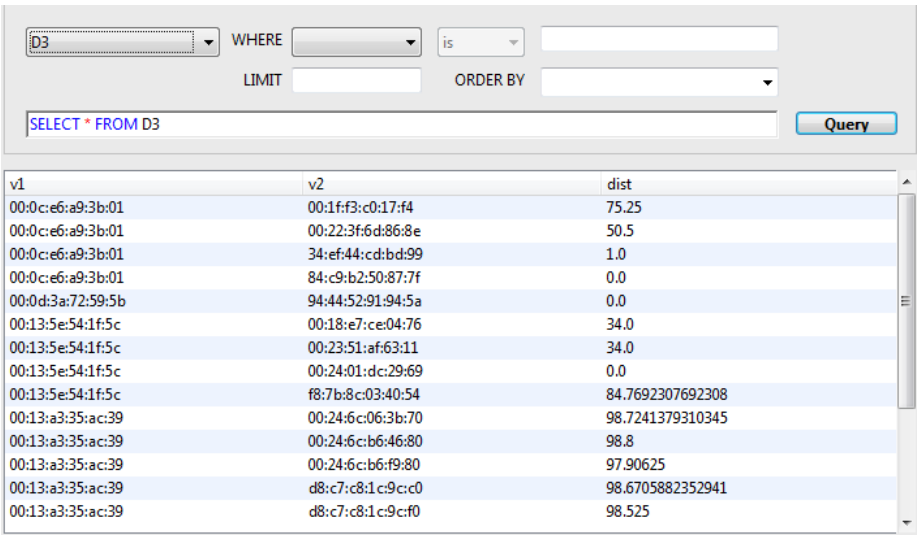
Figure 4.10 D1 Table



v1	v2	dist
00:0c:e6:a9:3b:01	00:1f:f3:c0:17:f4	75.25
00:0c:e6:a9:3b:01	00:22:3f:6d:86:8e	50.5
00:0c:e6:a9:3b:01	34:ef:44:cd:bd:99	1.0
00:0c:e6:a9:3b:01	84:c9:b2:50:87:7f	0.0
00:0d:3a:72:59:5b	94:44:52:91:94:5a	0.0
00:13:5e:54:1f:5c	00:18:e7:ce:04:76	34.0
00:13:5e:54:1f:5c	00:23:51:af:63:11	34.0
00:13:5e:54:1f:5c	00:24:01:dc:29:69	0.0
00:13:5e:54:1f:5c	f8:7b:8c:03:40:54	84.7692307692308
00:13:a3:35:ac:39	00:24:6c:06:3b:70	98.7241379310345
00:13:a3:35:ac:39	00:24:6c:b6:46:80	98.8
00:13:a3:35:ac:39	00:24:6c:b6:f9:80	97.90625
00:13:a3:35:ac:39	d8:c7:c8:1c:9c:c0	98.6705882352941
00:13:a3:35:ac:39	d8:c7:c8:1c:9c:f0	98.525

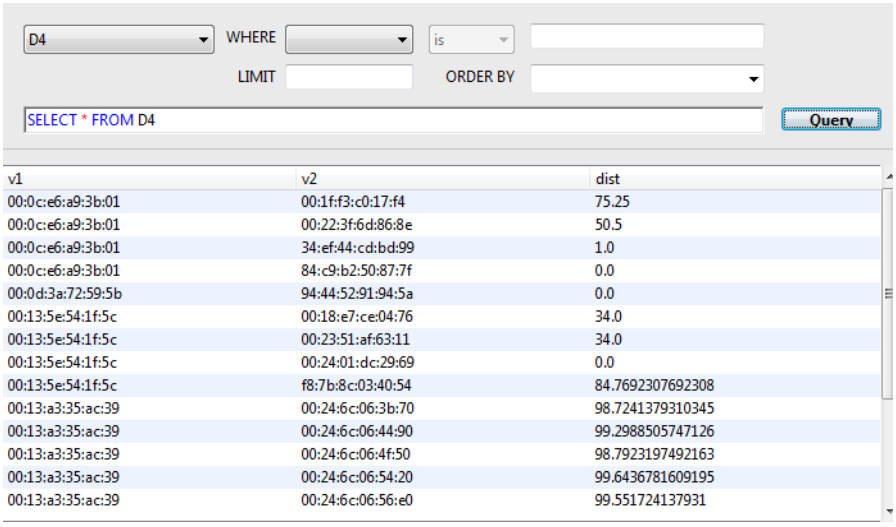
Figure 4.11 D2 Table

The same technique applied to D3 and D4 that include distance between 3 hops and 4 hops.



v1	v2	dist
00:0c:e6:a9:3b:01	00:1f:f3:c0:17:f4	75.25
00:0c:e6:a9:3b:01	00:22:3f:6d:86:8e	50.5
00:0c:e6:a9:3b:01	34:ef:44:cd:bd:99	1.0
00:0c:e6:a9:3b:01	84:c9:b2:50:87:7f	0.0
00:0d:3a:72:59:5b	94:44:52:91:94:5a	0.0
00:13:5e:54:1f:5c	00:18:e7:ce:04:76	34.0
00:13:5e:54:1f:5c	00:23:51:af:63:11	34.0
00:13:5e:54:1f:5c	00:24:01:dc:29:69	0.0
00:13:5e:54:1f:5c	f8:7b:8c:03:40:54	84.7692307692308
00:13:a3:35:ac:39	00:24:6c:06:3b:70	98.7241379310345
00:13:a3:35:ac:39	00:24:6c:b6:46:80	98.8
00:13:a3:35:ac:39	00:24:6c:b6:f9:80	97.90625
00:13:a3:35:ac:39	d8:c7:c8:1c:9c:c0	98.6705882352941
00:13:a3:35:ac:39	d8:c7:c8:1c:9c:f0	98.525

Figure 4.12 D3 Table



v1	v2	dist
00:0c:e6:a9:3b:01	00:1f:f3:c0:17:f4	75.25
00:0c:e6:a9:3b:01	00:22:3f:6d:86:8e	50.5
00:0c:e6:a9:3b:01	34:ef:44:cd:bd:99	1.0
00:0c:e6:a9:3b:01	84:c9:b2:50:87:7f	0.0
00:0d:3a:72:59:5b	94:44:52:91:94:5a	0.0
00:13:5e:54:1f:5c	00:18:e7:ce:04:76	34.0
00:13:5e:54:1f:5c	00:23:51:af:63:11	34.0
00:13:5e:54:1f:5c	00:24:01:dc:29:69	0.0
00:13:5e:54:1f:5c	f8:7b:8c:03:40:54	84.7692307692308
00:13:a3:35:ac:39	00:24:6c:06:3b:70	98.7241379310345
00:13:a3:35:ac:39	00:24:6c:06:44:90	99.2988505747126
00:13:a3:35:ac:39	00:24:6c:06:4f:50	98.7923197492163
00:13:a3:35:ac:39	00:24:6c:06:54:20	99.6436781609195
00:13:a3:35:ac:39	00:24:6c:06:56:e0	99.551724137931

Figure 4.13 D4 Table

4.2 History Path Plotting

Data plotting is the connection of trace of combination of printing v1 and v2 from the table of similarity and taking the maximum signal strength from the table of readings. In order to clean up the unwanted nodes for the plotting, the path is taken from the selective pair of v1 and v2 after comparing their similarity values are great than 0.5. It is in line with the heuristic Wi-Fi

distance calculation as the user defined value. If value of sim is chosen only greater than 0.0, the majority of the pair of v1 and v2 will be added into the path.

The following plots are that derived one particular data file called drive.txt that has been collected early. Blue line is the link between nodes. Black dot represents the selected node. Those black lines and dots are noise from the surrounding environments such as passing by mobile users, and they are not part of the history path.

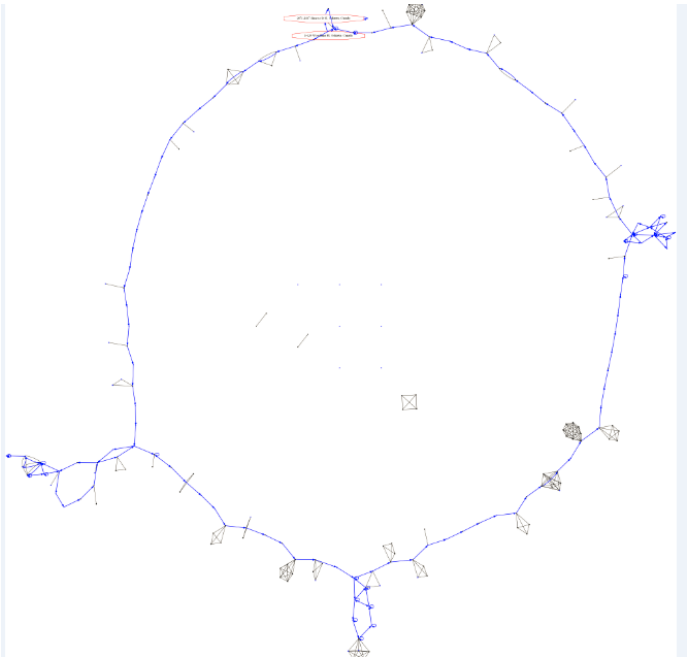


Figure 4.14 Data Collection Plotting of Route 1

The plotting of route 1 shown above is a round trip from Oshawa UOIT area to Hwy 7, McCowan at Markham then all the way come back via Taunton road.

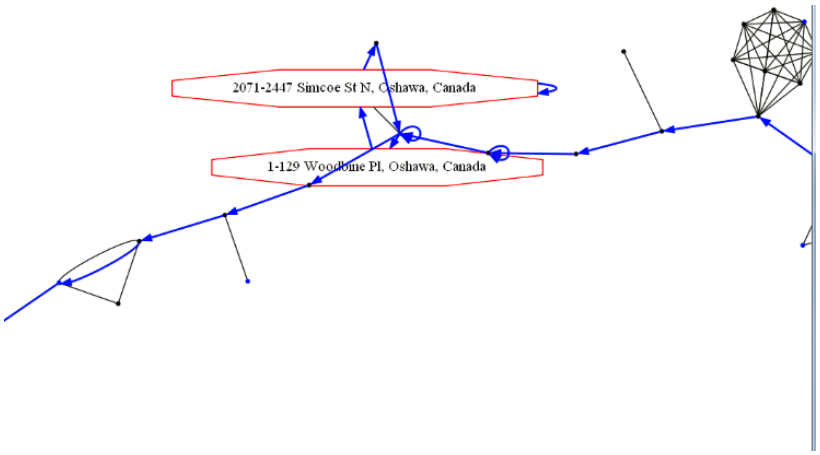


Figure 4.15 Route 1 with Labels of Known Addresses Zoomed In

The above figure shows the partial plot zoomed in with label of the known address.

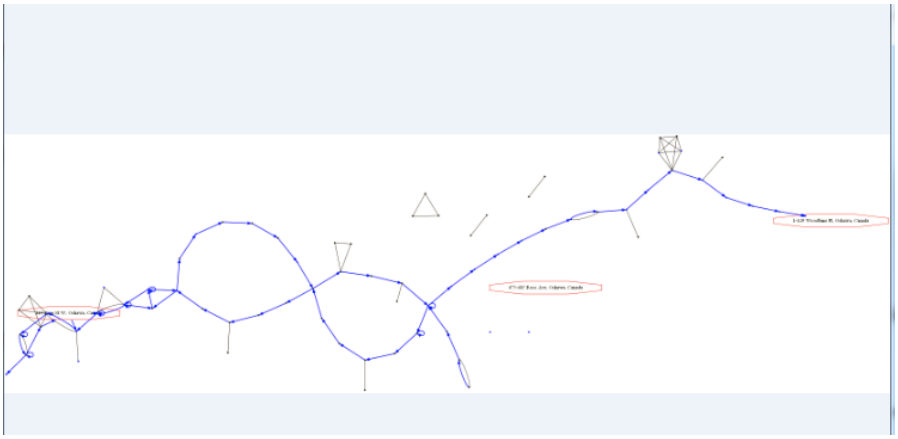


Figure 4.16 Data Collection Plotting of Route 2

It is a trip from Oshawa Town Centre at 419 King st W, Oshawa to 475-485 Ross Ave, Oshawa then all come back to UOIT area along Taunton road with selective labels.

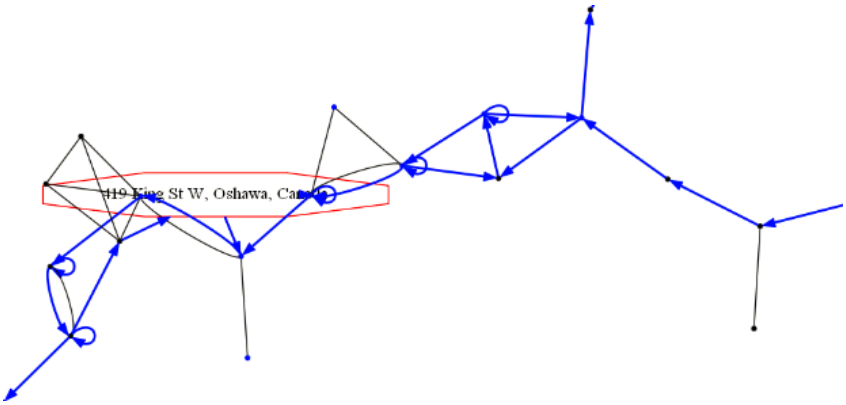


Figure 4.17 Route 2 with One of Known Address Labelled Zoomed In

Figure 4.17 shows the zoomed in partial plot from the entire plot of route 2. It could lead to a too many labelling in the plot if all the known addresses are shown, therefore only a limited number of labels are displayed in the zoomed in figures.

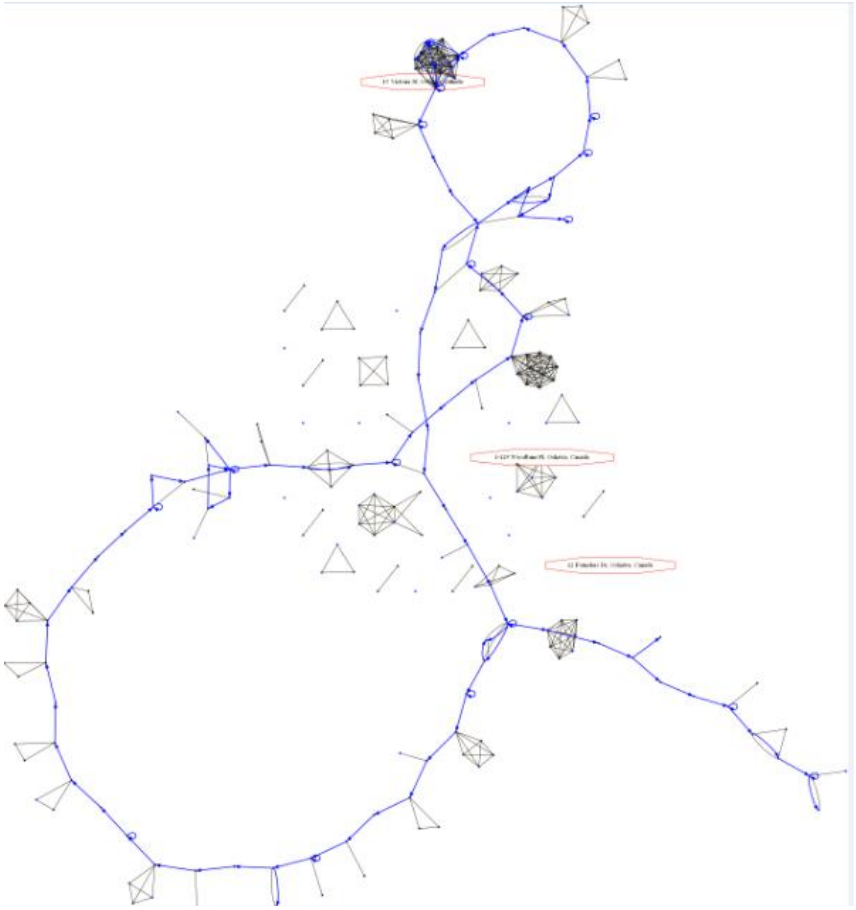


Figure 4.18 Data Collection Plotting of Route 3

The above trip is from the UOIT Campus to Regent Centre, Downtown Oshawa and then come back to UOIT campus in various buildings with selective labels.

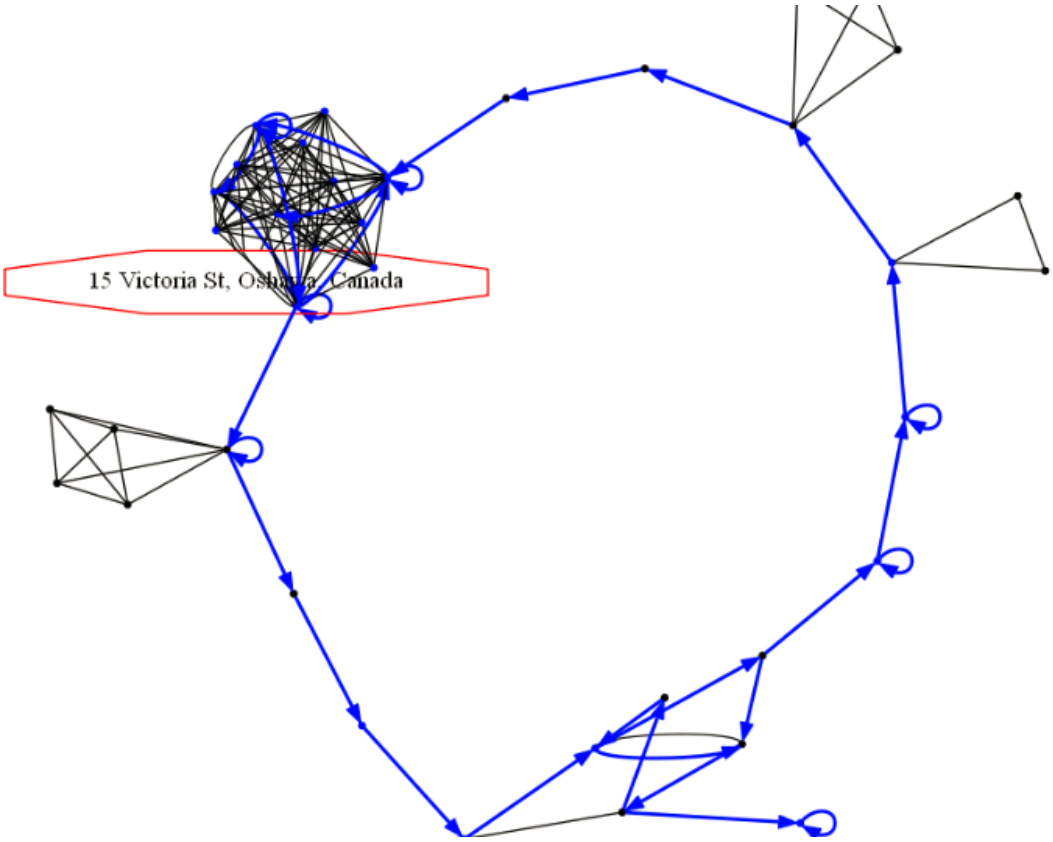


Figure 4.1911Route 3 with one of Known Address Labelled Zoomed In Top View

Figure 4.19 Route 3 with One of Known Address Labelled Zoomed In Top

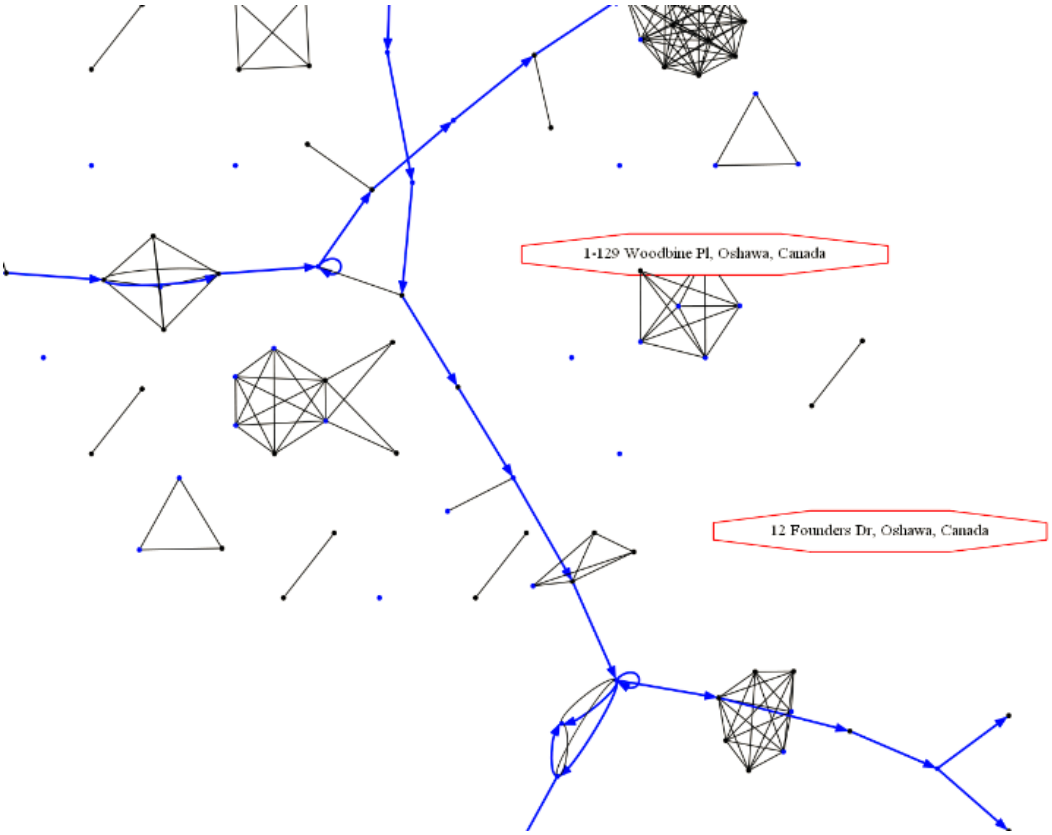


Figure 4.20 Route 3with One of Known Address Labelled Zoomed In Bottom View

4.3 Data Query

The text based data query based on joint address table and D4 table is developed for better demonstration of data inference. It can be used as the database query for any bssid and returning a series of bssid and corresponding Wi-Fi space distances, longitude, latitude and the physical address in ascending order. It tells minimum distance from the users location to a known address stored in the database.

```
C:\Grads\Thesis\Latest DB(Lin)\leon>python query-one.py db/all.sqlite3 00:26:82:
a5:cb:97
Possible locations: (0.001 seconds)
---------------------------------
[e8:40:f2:6b:f4:e7] 0.49 (43.95088483, -78.89648250, 1-129 Woodbine Pl, Oshawa,
Canada)
[38:60:77:41:9d:f8] 16.00 (43.95091420, -78.89642500, 2-96 Woodbine Pl, Oshawa,
Canada)
[c0:c1:c0:f9:c9:f4] 40.00 (43.95101410, -78.89692090, 1 Woodbine Pl, Oshawa, Can
ada)
[4c:17:eb:e2:4e:e5] 76.00 (43.95114094, -78.89655150, )
[64:0f:28:e1:6c:e1] 82.00 (43.95092628, -78.89667787, 1-129 Woodbine Pl, Oshawa,
 Canada)
[5c:d9:98:64:d5:62] 94.00 (43.95114094, -78.89655150, 96 Woodbine Pl, Oshawa, Ca
nada)
[e0:46:9a:50:97:ba] 104.35 (43.95084850, -78.89712740, 2071-2447 Simcoe St N, Os
hawa, Canada)
```

Data query returns logic distance values from the known addresses in an ascending order. It has not been developed at this stage to explore how accurate the distance values are in terms of real distance, therefore there is no unit for the distance values. By injecting difference threshold of similarity, distance values might return differently from the data query, but it is insignificant at this stage because the logical distance values have not been measured with real distance values. However, it has demonstrated the huge potential to link the logical distance values with the real distance values using Google maps if further investigation and development are put on the agenda.

Chapter 5 Conclusion and Future Development

Location awareness is one of most popular researches among all mobile application development. It brings the convenience to mobile users especially on the go. However, the lack of systematically managed architecture using existing well established Wi-Fi technologies has brought attentions. With the existing android platform, the developers can use the user friendly Eclipse IDE to develop applications within corresponding Android SDK. The Java development and open source concept provided rich experience for developers to extend their existing applications. The android application collecting Wi-Fi information provides the one of most economical ways to present the location awareness mechanism. As we know, mobile devices equipped GPS sensors are the power source driven for location display to perform efficiently and accurately. This research approach can significantly resolve the issue of utilizing power resource on the mobile devices by sensing only Wi-Fi information. It could lead to be used in many areas such as in the areas of police investigations, mobile commuters and phone tracking and so on.

The research can be extended extensively into next stages with a few approaches. First, the data analysis can be done at real time. It might include the data compression and integration with Google map display on the mobile devices. Modern mobile devices equip with more advanced hardware specifications, such as quad-core processors with several gigabytes of RAM, to increase the performance of devices. It could lead to more applications can perform sophisticated tasks such as database implementation and data inference and plotting at real time. Second, effective routing such as finding shortest distance algorithm using Ant colony

heuristic with current android application extracting Wi-Fi information for route optimization could lead to extensive research. The offline data analysis could be performed in a way to measure the shortest path metrics incorporating the ant colony optimization that the path pheromones scent technique for mobile users in the mobile Ad-hoc Networks [25].

Third, the data synchronization method can be considered by implementing the middleware data synchronization layer [26]. GPS is considered one of the most reliable navigation tools for people travel among cities, provinces or even across the board, but the time of exclusively using GPS technology has almost come to an end. Therefore, the combination of using GPS, Wi-Fi and Cloud technology can be next step for the future mobile and data layer development. Like GPS technology 5 to 10 years ago, the relative younger cloud technology has also been seen a wider acceptance in the industry as people have adopted cloud computing data storage. Cloud managed Wi-Fi architecture maybe becomes one of the future developments in years to come.

The future development can be considered practical using the current methodology and algorithms expanding on to the daily activities. For example, one of future projects can be developed for subway systems. It can be also used with client/server based mechanism. Database development could be implemented for large data collection.

There are few advantages using Cloud based Location awareness devices, such as Cloud based GPS:

1. Cloud based GPS can provide always up to date service. Any coding patches or system and content updates will be available on the cloud. Users would enjoy the hassle free services and get updated information.

2. All devices setting, personal files and configuration information can be accessible via the cloud service. For example, if the mobile device is broken, the customers can always get a new one by not to worrying about the lost data. The hardware failure is no longer an issue affecting the software hosted on the cloud.

3. Because the cloud is a centralized service, cloud based the application can be protected by the vendors or third-party.

4. Cloud based the application can be shared with each other facilitated by the vendor and support staff. Through the sharing systems, waypoints, maps, traffic updates, and other content can be easily sent among customers.

For the cloud based service, the internet connection has been considered the fundamental to all. Wi-Fi technology is inevitably and necessarily deployed in this future architecture. Therefore the Internet especially Wi-Fi service will the tunnel between local devices, such as mobile devices, GPS, and cloud services.

References

[1]. Location awareness http://en.wikipedia.org/wiki/Location_awareness Last time visited: July 2013

[2]. Global Position System http://en.wikipedia.org/wiki/Global_Positioning_System Last time visited: July 2013

[3]. How Much Do Average Apps Make http://www.forbes.com/sites/tristanlouis/2013/08/10/how-much-do-average-apps-make/ last time visit: Sep 2013

[4]. Alisa Marshall, Signal-Based Location Sensing using 802.11b

[5]. Castro, Paul, et al. "A probabilistic room location service for wireless networked environments." Ubicomp 2001: Ubiquitous Computing. Springer Berlin Heidelberg, 2001.

[6]. Bahl, Paramvir, and Venkata N. Padmanabhan. "RADAR: An in-building RF-based user location and tracking system." INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. Vol. 2. IEEE 2000.

[7]. Yang, Jie, and Yingying Chen. "Indoor localization using improved rss-based lateration methods." Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE. IEEE, 2009.

[8]. Gwon, Youngjune, Ravi Jain, and Toshiro Kawahara. "Robust indoor location estimation of stationary and mobile users." INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies. Vol. 2. IEEE, 2004.

[9]. Small, J., et al. Determining a user location for context aware computing through the use of a wireless LAN infrastructure. Project Aura report, 2000. http://www.cs.cmu.edu/~aura/docdir/small00.pdf.

[10]. Grossmann, Uwe, Markus Schauch, and Syuzanna Hakobyan. "RSSI based WLAN indoor positioning with personal digital assistants." Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2007. IDAACS 2007. 4th IEEE Workshop on. IEEE, 2007.

[11]. Ladd, Andrew M., et al. "Robotics-based location sensing using wireless ethernet." Wireless Networks 11.1-2 (2005): 189-204.

[12]. Retscher, Guenther, et al. "Performance and Accuracy Test of the WLAN Indoor Positioning System" ipos." Proceedings of the 3rd Workshop on Positioning, Navigation and Communication. 2006.

[13] Teuber, Andreas, and Bernd Eissfeller. "WLAN indoor positioning based on Euclidean distances and fuzzy logic." Proceedings of the 3rd Workshop on Positioning, Navigation and Communication. 2006.

[14]. Quan, Michael, Eduardo Navarro, and Benjamin Peuker. "Wi-Fi Localization Using RSSI Fingerprinting," 2010.

[15]. Prasithsangaree, Phongsak, Prashant Krishnamurthy, and Panos Chrysanthis. "On indoor position location with wireless LANs." Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium on. Vol. 2. IEEE, 2002.

[16]. Smailagic, Asim, and David Kogan. "Location sensing and privacy in a context-aware computing environment." Wireless Communications, IEEE 9.5 (2002): 10-17.

[17]. Roos, Teemu, et al. "A probabilistic approach to WLAN user location estimation." International Journal of Wireless Information Networks 9.3 (2002): 155-164.

[18] Youssef, Moustafa A., Ashok Agrawala, and A. Udaya Shankar. "WLAN location determination via clustering and probability distributions." Pervasive Computing and Communications, 2003.(PerCom 2003). Proceedings of the First IEEE International Conference on. IEEE, 2003.

[19]. Bshara, Mussa, et al. "Fingerprinting localization in wireless networks based on received-signal-strength measurements: A case study on WiMAX networks."Vehicular Technology, IEEE Transactions on 59.1 (2010): 283-294.

[20]. Basic Service Set http://en.wikipedia.org/wiki/Service_set_(802.11_network)#Basic_service_set Last time visited: June 2013

[21]. Michael Negnevitsky, Artificial Intelligence: A Guide to Intelligent Second Edition, Published by Pearson Education 2008

[22]. Lock http://developer.android.com/reference/java/util/concurrent/locks/Lock.html Last time visited: Oct 2013

[24]. Activity Life Cycle

http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle Last time visited: Oct 2013

[25] Ajay C Solai Jawahar, Ant Colony Optimization for Mobile Ad-hoc Networks

[26]. Yanxin Xue, The research on data synchronization of distributed real-time mobile network

[27] Hightower, Jeffrey, and Gaetano Borriello. "Location systems for ubiquitous computing." *Computer* 34.8 (2001): 57-66.

[28] Haeberlen, Andreas, et al. "Practical robust localization over large-scale 802.11 wireless networks." *Proceedings of the 10th annual international conference on Mobile computing and networking*. ACM, 2004.

[29] Krishnan, Parameshwaran, et al. "A system for LEASE: Location estimation assisted by stationary emitters for indoor RF wireless networks." *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*. Vol. 2. IEEE, 2004.

[30] Cheng, Yu-Chung, et al. "Accuracy characterization for metropolitan-scale Wi-Fi localization." *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM, 2005.

[31] Letchner, Julie, Dieter Fox, and Anthony LaMarca. "Large-scale localization from wireless signal strength." *Proceedings of the national conference on artificial intelligence*. Vol. 20. No. 1. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[32] Schilit, Bill, Norman Adams, and Roy Want. "Context-aware computing applications." *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*. IEEE, 1994.