CSE 5520 Wireless Networks and Protocols

Android Project

Nanda Rampura

Table of Contents

1. Problem Definition

On android mobile platform we need to implement two applications. First application, 'Wireless Application', is for accessing and saving telephony and location information such as Base Station Cell ID, signal strength, latitude, longitude etc. Using this application we should collect data in various indoor and outdoor locations, and analyze the relationships between various metrics and recognize trends. We are also expected to obtain further data about the cell towers using services available online. Second application, 'Traceroute Application', is for generating UNIX style traceroute for different servers on Android platform. Application should also use location services to obtain the location data for the originating device, target server and hops along the way.

2. Developmental Environment

IDE used for developing the Wireless Application is Android Studio 1.4, with Java Runtime Environment 1.7.0. Target SDK version of Android used is 19. SDK version 19 corresponds to Android 4.4, codenamed KitKat. Minimum SDK version is also 19. Initial testing was done using Custom generated Android virtual device with resolution set to match the actual device used, 320 X 480. Full testing and data measurements were taken using LG L15G Sunrise phone running Android KitKat.

3. Software Architecture

3.1 Wireless Application

The application consists of two activities – activity_main and activity_tel. Layout of the UI for both the activities are implemented in XML. activity_main is the landing page of the application. It consists of a button to launch the page containing the telephony and location data, which is described by activity_tel. Here is the screenshot of the landing page:

activity_tel consists of text boxes to display the current reading of the telephony and location data measurements, as shown below:

All the measurement data are stored in read-only elements implemented as TextView. User Label is an EditText. Save Data and Main Menu are buttons. Save Data button will save the current data as displayed on the screen to an image file with current timestamp as its name. Location of screenshots is internal_storage/_nanda_wnp. Toast is used to convey successful saving of the image file. User Label field can be used to enter any string to help with tagging the current measurement. All the measurement data is also saved to a csv file at internal_storage/_nanda_wnp.

Most of the telephony and location fields were present in the original application given to us. Because the network I used for testing was an UMTS (Universal Mobile Telecommunication System) network, CID reported by Android's GsmCellLocation API was an UTRAN CID. So, I added fields for UTRAN CID and RNC ID. For obtaining the UMTS cell tower location information, we need MCC (Mobile Country Code), MNC (Mobile Network Code), and PSC (Primary Synchronization Code). So I added fields for these data too.

The backend Java class for activity_main consists of calling setOnClickListener() for Telephony and Location Manager button, in onCreate() method. Java class for activity_tel consists of private members for all the measurement data, TextView and android's TelephonyManager object. In onCreate(), we call setOnClickListener() for the two buttons, initialize the TextView objects, associate PhoneStateListener to TelephonyManager, instantiate LocationManager, and associate LocationListener to it. We also create the csv file and its headers in onCreate().

We override PhoneStateListener's onSignalStrengthsChanged() method. In this method we update the telephony data. We override LocationListener's onLocationChanged() method. In this method we update the location data. Whenever any data is changed, we add an entry to the csv file.

onResume() and onPause() methods are overridden to start and stop the measurement collection when the app goes to the foreground and background.

3.2 Traceroute Application

Traceroute application consists of one activity which contains EditText boxes to enter the Hostname or IP address to perform the Traceroute, and to enter the file name. It also consists of a button to start the Traceroute. Toasts are used to convey the progress of the Traceroute process.

Backend in Java consists of a class called AndroidTracerouteInstallerThread to install the traceroute binary, class that encapsulates the location data called Location. LocationThread class looks up the locations for the addresses in the traceroute. TracerouteThread performs the actual Traceroute operation. TraceRouteResult class stores the Traceroute data to a csv file. Location information is obtained from http://ip-api.com using the API via JSON objects. Traceroute data is saved in a csv file.

3.3 Android APIs

a) **TelephonyManager** Class – Provides access to information about the telephony services on the device.

b) **PhoneStateListener** Class – Listener for monitoring changes in specific telephony states on the device, like signal strength, service state etc.

c) **LocationManager** Class – Provides access to the system location services, which allow applications to obtain periodic updates of the device's geographic location.

d) **LocationListener** Class – Used for receiving notifications from LocationManager when the location has changed.

e) **public void onSignalStrengthsChanged (SignalStrength signalStrength)**

   Callback invoked when network signal strength changes.

f) **public abstract void onLocationChanged (Location location)**

   Called when the location has changed.

g) **public int getGsmSignalStrength ()**

   Get the GSM Signal Strength, valid values are (0-31, 99) as defined in TS 27.007 8.5

h) **GsmCellLocation** Class – Represents the cell location on a GSM phone.

   Callback invoked when network signal strength changes.

i) **public int getCid ()**

   Returns GSM Cell ID, which is -1 if unknown, and 0xffff is the max legal value. On UMTS networks the Cell ID returned is an UTRAN CID, which is concatenation of 12-bits of RNC (Radio Network Controller) ID and 16-bits of Cell ID.

j) **public int getLac ()**

Returns GSM Location Area Code, which is -1 if unknown, and 0xffff is the max legal value.

k) **public int getPsc ()**

On a UMTS network, returns the primary scrambling code of the serving cell.

l) **public String getNetworkOperator ()**

Returns the numeric name (MCC+MNC) of current registered operator.

m) **public static Enumeration<NetworkInterface> getNetworkInterfaces ()**

Gets a list of all network interfaces available on the local system or null if no interface is available.

n) **public Enumeration<InetAddress> getInetAddresses ()**

Returns an enumeration of the addresses bound to this network interface.

o) **public String getHostAddress ()**

Returns the numeric representation of this IP address (such as "127.0.0.1").

p) **public double getLatitude ()**

Get the Latitude, in degrees.

q) **public double getLongitude ()**

Get the Longitude, in degrees.

r) **public double getAltitude ()**

Get the altitude if available, in meters above the WGS 84 reference ellipsoid. If this location does not have an altitude then 0.0 is returned. Note that this altitude is not the altitude above sea level. To obtain sea-level altitude based on WGS 84 altitude, conversion formula needs to be applied.

s) **public float getBearing ()**

Get the bearing, in degrees. Bearing is the horizontal direction of travel of this device, and is not related to the device orientation. It is guaranteed to be in the range (0.0, 360.0] if the device has a bearing. If this location does not have a bearing then 0.0 is returned.

t) **public float getAccuracy ()**

Get the estimated accuracy of this location, in meters. We define accuracy as the radius of 68% confidence. In other words, if you draw a circle centered at this location's latitude and longitude, and with a radius equal to the accuracy, then there is a 68% probability that the true location is inside the circle.

In statistical terms, it is assumed that location errors are random with a normal distribution, so the 68% confidence circle represents one standard deviation. Note that in practice, location errors do not always follow such a simple distribution. This accuracy estimation is only concerned with horizontal accuracy, and does not indicate the accuracy of bearing, velocity or altitude if those are

included in this Location. If this location does not have an accuracy, then 0.0 is returned. All locations generated by the LocationManager include an accuracy.

u) **public float getSpeed ()**

Get the speed if it is available, in meters/second over ground. If this location does not have a speed then 0.0 is returned.

4. Data and Analysis

Wireless service used for data measurements is TracFone Wireless, which is mobile virtual network operator (MVNO) which uses AT&T's 3G GSM (UMTS) network. Cellular data and WiFi were enabled during data measurements.

4.1 Indoor Signal Change

The Building I chose for indoor signal change measurements is Discovery Park at UNT.

Table 1 – Tower Information

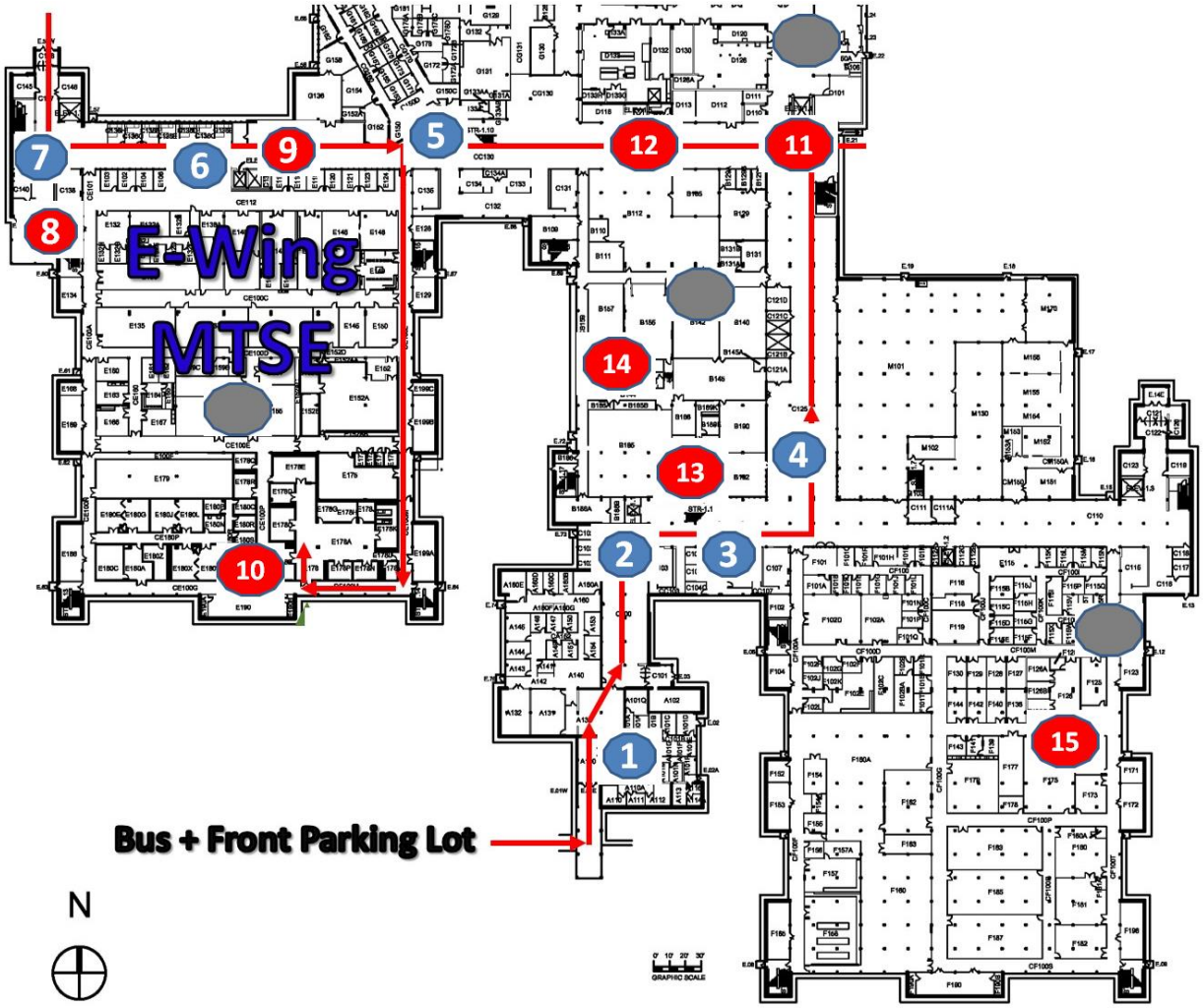
	Location	Cell ID	LAC	Internal IP	External IP	Server Location
1	A101	60574	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)
2	C102C	2025	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)
3	C104	14901	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)
4	B192	2025	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)
5	G140	116	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)
6	C136B	22192	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)
7	C141	12383	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)
8	C241	113	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)
9	C236	12383	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)
10	E292	29215	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)
11	D206A	60574	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)
12	D210A	2025	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)
13	B280	113	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)
14	B270	14901	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)
15	F230	60574	52717	10.124.12.115	107.77.169.7	Lat: 29.7633, Long: -95.3633 (Houston, TX)

The table contains two kinds of IP addresses – Internal IP address and External IP address. Internal IP address is the address assigned by the local network router because of NAT (Network Address Translation). External IP address is the address assigned to the network router, and is used by external hosts on the internet to reach the said router and all devices behind it.

I also collected measurement data over WiFi. On WiFi, Internal IP was 10.124.12.129, and external IP was 129.120.2.131. The location of the external IP is UNT, Denton (Lat: 33.2148, Long: -97.1331).

Internal IP address can be obtained by using Android API's **getInetAddresses()** function. External IP can be obtained by using http://ip-api.com. The server location was also obtained from the same website.

The locations where the measurements were taken are marked in the indoor map below:



Locations marked with blue are on the first floor and the locations marked with red are on the second floor. Locations marked with grey are the spots where there were coverage holes. The locations where

measurements were taken are the locations where handoff happened. As we can see there are quite a few towers that provide coverage to the building. The reason we see many handoffs in such a small area is because the signal strength inside the building was low (between -80 and -110 dbm), and the handoff threshold was reached in many places. There were a few places inside the building which had coverage holes (less than -111 db), mostly in stairwells and interior corridors.

Screenshots of the measurement data at all the locations is included in IndoorLocationsData folder. csv file containing continuous measurements is also included in IndoorLocationsData.csv. From the csv file, we can see the exact locations where handoffs happen.

4.2 Outdoor Signal Change and GPS Reading

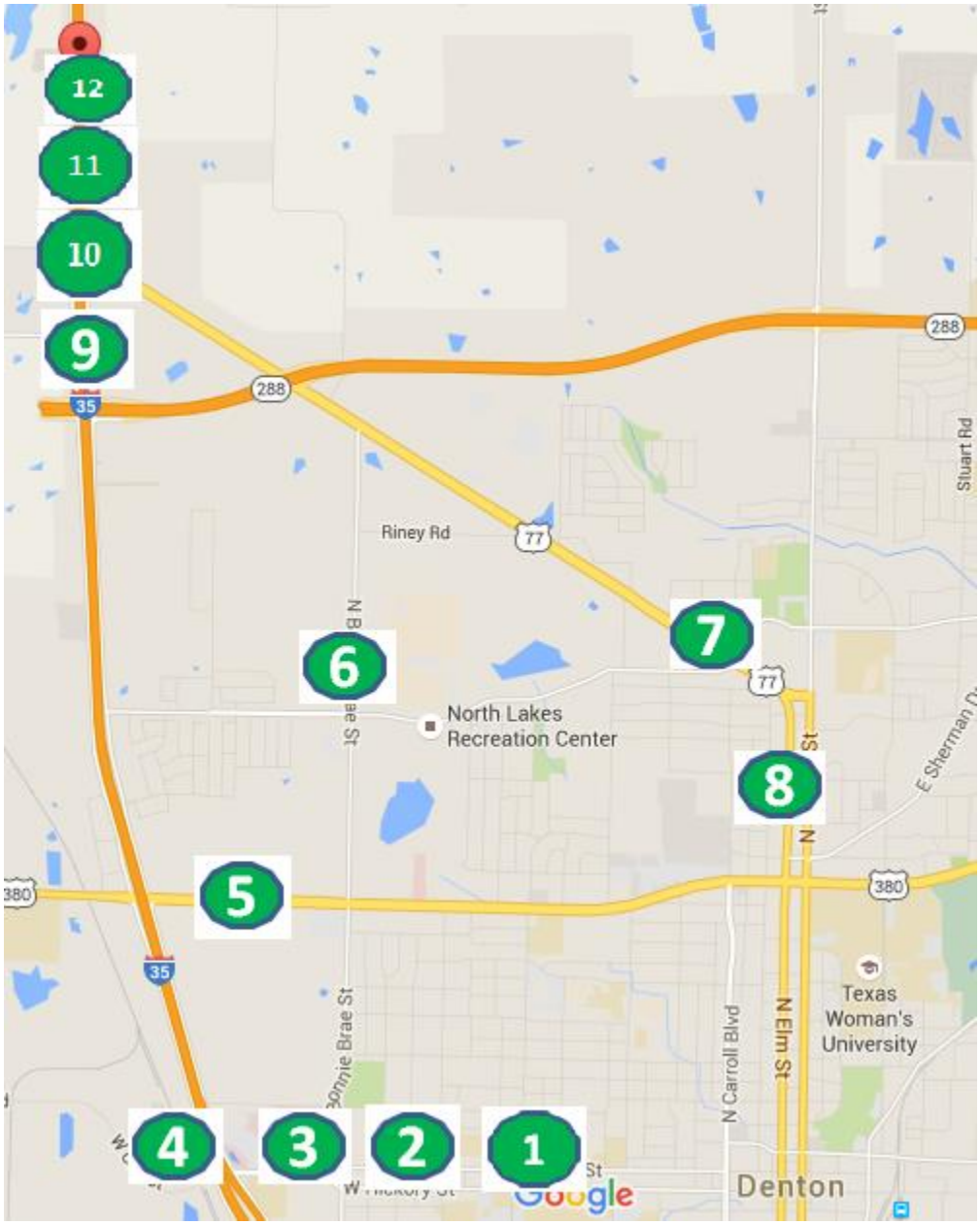
Table 2 – GPS Information

	Recorded Speed	Cell ID	LAC	IP	Accuracy	Bearing	Latitude	Longitude	Speed of Movement from API	Altitude
1	20 mph	60572	52717	10.117.103.111	4.0	270.10	33.21579	-97.15580	20.132	186.0
2	20 mph	60572	52717	10.117.103.111	3.0	269.90	33.21581	-97.15877	20.132	189.0
3	20 mph	60572	52717	10.117.103.111	4.0	256.0	33.21577	-97.16171	20.132	179.0
4	20 mph	14903	52717	10.117.103.111	8.0	270.90	33.21546	-97.16370	20.69	178.0
5	40 mph	60573	52717	10.117.103.111	4.0	88.50	33.22981	-97.16887	40.26	189.0
6	40 mph	60571	52717	10.117.103.111	3.0	1.60	33.24114	-97.16068	39.71	185.0
7	40 mph	60021	52717	10.117.103.111	9.0	115.40	33.24217	-97.13630	40.27	185.0
8	40 mph	60022	52717	10.117.103.111	4.0	183.80	33.23482	-97.13336	40.27	188.0
9	60 mph	29212	52717	10.117.103.111	3.0	357.30	33.25969	-97.17770	60.40	191.0
10	60 mph	12383	52717	10.117.103.111	3.0	359.70	33.26985	-97.17784	60.40	184.0
11	60 mph	29215	52717	10.117.103.111	3.0	359.50	33.27058	-97.17785	60.4	183.0
12	60 mph	113	52717	10.117.103.111	5.0	359.80	33.27204	-97.17786	59.84	184.0

As we saw in the Accuracy API description, lower accuracy number means better accuracy. Apart from one or two anomalous measurements, we see that the accuracy is better at lower speeds, which is what one would expect.

The screenshots of all the measurements and the csv files are included in OutdoorLocationsData folder.

The outdoor locations are marked on the map below:



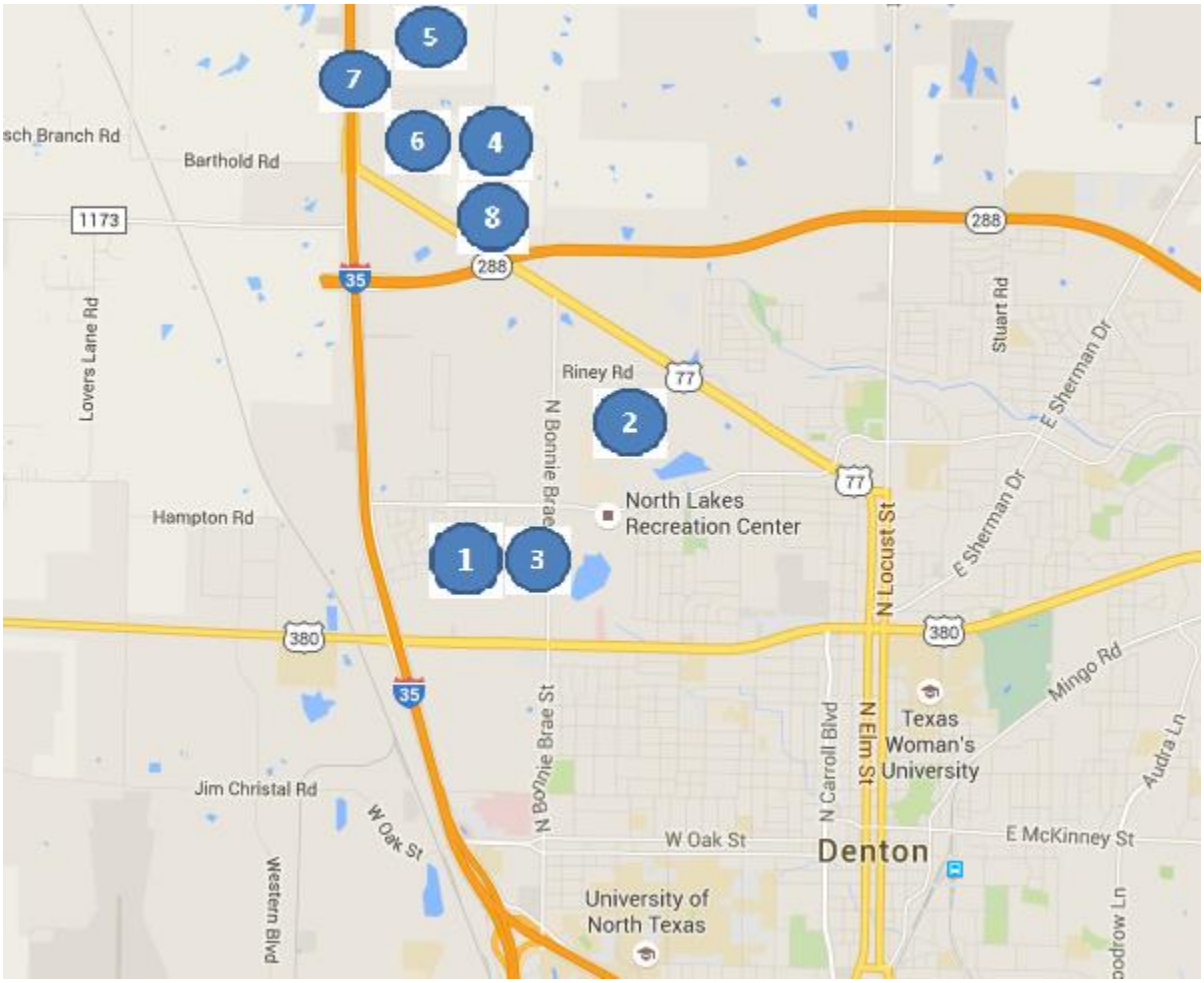
As we can see from the data, handoff happened only once when driving at 20 mph, since the distance covered is small. At higher speeds, handoff happened between each data point. At 60 mph on the highway, handoff happened within 1-2 miles.

4.3 Physical Locations

Table 3 – Physical Locations

	Cell ID	GPS Location		Tower Location		IP Location	
		Latitude	Longitude	Latitude	Longitude	Latitude	Longitude
1	**60574**	33.25345473	-97.15331493	33.236206	-97.166649	29.7633	-95.3633
2	**2025**	33.25358737	-97.15302959	33.244643	-97.151138	29.7633	-95.3633
3	**14901**	33.2535783	-97.15303937	33.236267	-97.164719	29.7633	-95.3633
4	**116**	33.25435453	-97.15271468	33.265799	-97.165741	29.7633	-95.3633
5	**22192**	33.25440848	-97.15290541	33.281022	-97.169739	29.7633	-95.3633
6	**12383**	33.25477459	-97.15353905	33.268764	-97.171717	29.7633	-95.3633
7	**113**	33.25478169	-97.15355204	33.269202	-97.173996	29.7633	-95.3633
8	**29215**	33.25463645	-97.15331545	33.268187	-97.161050	29.7633	-95.3633

The tower locations are shown in the map below:



GPS location is the location of mobile device. Tower Location is the location of the BTS obtained from https://unwiredlabs.com. All towers are located in Denton, TX. IP Location is the location of the network router obtained from http://ip-api.com. IP Location is in Houston, TX. We see that for each CID, tower location is different, as expected. IP location is different from Tower location since, tower location is the location of the BTS, and IP location is the location of the network router, which could be in MSC. Using these locations, we can develop location based services like search results based on current location, auto detection of location in maps, crowdsourcing of location specific data like weather, traffic conditions etc. Service providers can also provide internet access via WiFi using the location data.

4.4 Traceroute

I obtained the Traceroutes for the following businesses:

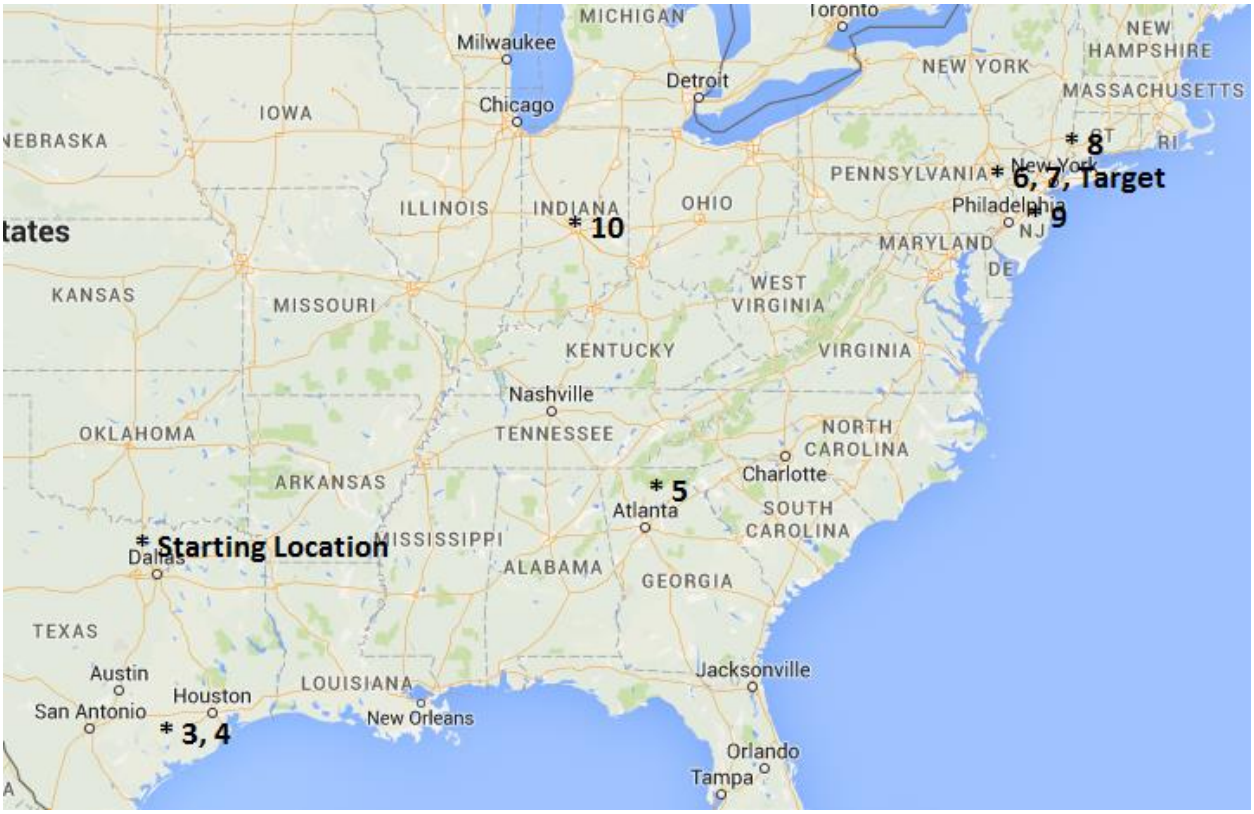
- Chase Bank
- CVS Pharmacy
- Dollar Tree
- IHOP
- Kroger *
- Startbucks *
- Subway
- Walgreens
- Walmart
- Wells Fargo Bank

The ones marked with * were obtained win WiFi enabled, and others were obtained with cellular data.

Traceroutes for all the businesses can be found in Traceroutes folder. For the purpose of illustration, we will pick on Traceroute data (Chase Bank) and analyze it.

TargetIP	TargetLat	TargetLon	StartingLat	StartingLon	HopNo	HopIP	HopLat	HopLon	HopRTTStart			
159.53.42.11	40.7087	-74.0104	33.23003	-97.1454	1	172.20.128.2	NaN	NaN	67.147			
159.53.42.11	40.7087	-74.0104	33.23003	-97.1454	2	72.20.128.131	NaN	NaN	166.251	166.06	165.821	169.691
159.53.42.11	40.7087	-74.0104	33.23003	-97.1454	3	107.77.168.13	28.8053	-97.0036	160.621	165.18	156.901	184.776
159.53.42.11	40.7087	-74.0104	33.23003	-97.1454	4	107.77.168.2	28.8053	-97.0036	184.604	184.287	187.514	183.661
159.53.42.11	40.7087	-74.0104	33.23003	-97.1454	5	07.77.170.116	33.8503	-84.347	136.985	104.391	110.2	102.299
159.53.42.11	40.7087	-74.0104	33.23003	-97.1454	6	12.83.186.101	40.7706	-74.5036	105.547	101.789	101.677	101.416
159.53.42.11	40.7087	-74.0104	33.23003	-97.1454	7	12.83.186.85	40.7706	-74.5036	107.482	107.278	103.717	106.94
159.53.42.11	40.7087	-74.0104	33.23003	-97.1454	8	12.122.2.121	40.7144	-74.006	106.774	106.646	74.535	68.057
159.53.42.11	40.7087	-74.0104	33.23003	-97.1454	9	12.122.100.89	40.3974	-74.1357	79.863	83.415	79.575	83.133
159.53.42.11	40.7087	-74.0104	33.23003	-97.1454	10	192.205.37.126	39.7709	-86.1585	95.363	79.991	72.597	65.072

As we can see, the target IP is 159.53.42.11. The target was reached in 10 hops. The first two IP addresses do not have location information because they are Internal IPs. RTT for all the hops seem consistent between tries. We did not encounter any routers along the way which did not respond to the traceroute command. The map below shows the starting location, target location and locations of all the hops along the way:



5. Design and Implementation Challenges

- Since the GSM network was UMTS based, I had to do some research to understand UTRAN CID format that is returned by Android API
- Getting the code for csv file save to work correctly was a bit tricky, mainly because of the permissions that we are supposed to setup, and getting access the required location
- The Wireless app was running continuously in the background and draining the battery, so had to fix a bug to stop measurements when application was in the background
- During outdoor data collection, it was difficult to see handoff happen at 20 mph

6. Conclusion

In this project, I extended the Wireless application on Android OS to measure UTRAN CID, MNC, MCC and PSC. I added the ability to take screenshots of the measurements, along with user label. I also modified the file save feature to save the data in csv format, for better readability. Using the Wireless application, I measured telephony and location data for various indoor and outdoor locations. I learnt about concepts like internal and external IP addresses, bearing, accuracy, and altitude using WGS 84 reference ellipsoid. Using Traceroute application, I obtained traceroutes for many target IP addresses, and learnt about how to obtain location information based on IP addresses.