International Journal of Engineering Research and Development
e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 9, Issue 12 (February 2014), PP. 05-12

Location Based Tracking

Sabyasachi Patra[1], Karishma Velisetty[2], Prathamesh Patel[3]
[1]*Mukesh Patel school of Technology and Management, Computer science, Mumbai, India*
[2]*Mukesh Patel school of Technology and Management, Computer science, Mumbai, India*
[3] *Mukesh Patel school of Technology and Management, Computer science, Mumbai, India*

Abstract: - This paper delves into a canopy of details as well as the intricate ones w.r.t Android Location Based services and route tracking. In this paper we present to you our 'Route Tracker' application. As the user travels with an Android device, our Route Tracker application monitors the user's location and bearing, visually displaying a route on a map. The user touches the 'Start Tracking' toggle button to begin tracking a route. The map shifts as the user moves, keeping the users current location centred on the screen. The route is basically a red line with black dots appearing every 10 data points received by the application. The view can be changed from 'Street Map' to 'Satellite map' as the user needs. As the user touches 'Stop Tracking' the app displays a dialog containing the total distance travelled (in KMs and Miles) and the average speed (in KPH and MPH) over the entire route.

Keywords:- Android, LBS, GPS, Google Maps, location based services, Route Tracking

I. INTRODUCTION

Nowadays, mobility is the most important criteria for a person/profession as such. Location based information help to find a find the accurate information at the right time and place. LBS in mobile phones help to visit places with complete information. With the help of wireless communication and location positioning technologies, location based services is emerging in the mobile domain.

LBS have two major actions, viz. obtaining the location of user and utilizing this information to provide a service

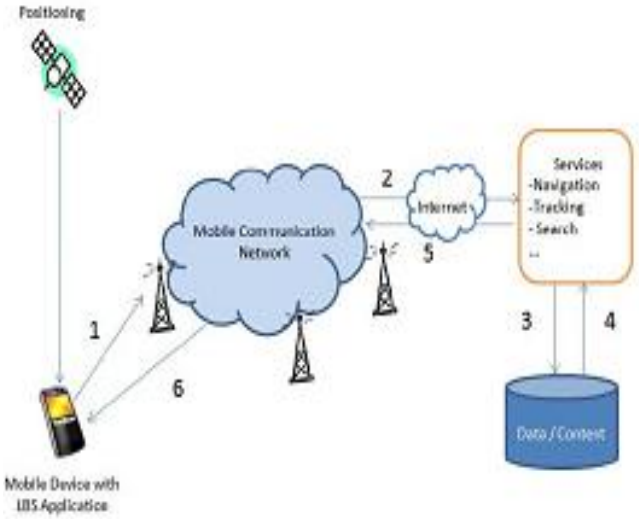

Fig. 1: LBS Architecture

LBS Categories:

- Triggered LBS services (push services)
- User-requested LBS services (pull services)

In a triggered (push) LBS service, the location of user's mobile device is retrieved when a condition set in advance and is fulfilled and in a user-requested (pull) LBS service. The user decides whether and when to retrieve the location of his/her mobile device and use it in the service [1][2][3].

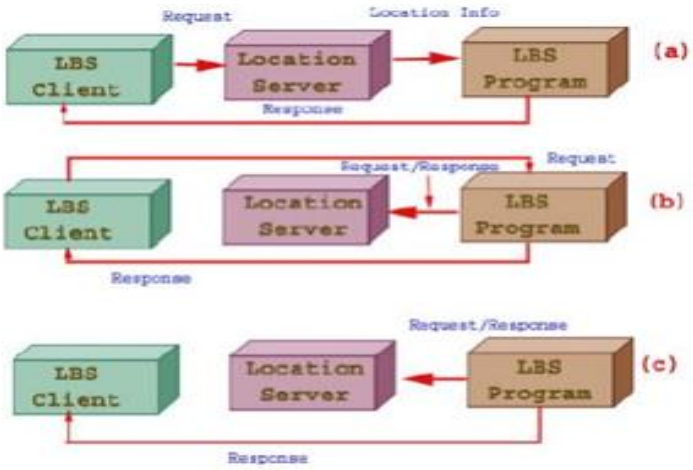

Fig. 2: LBS service models

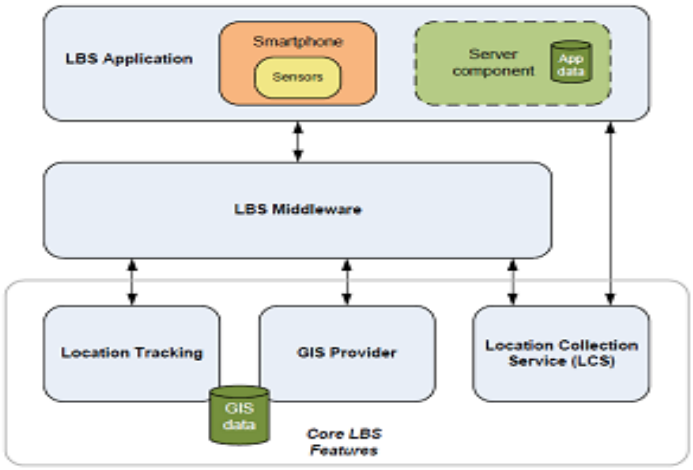

Fig. 3: Architecture model

Applications of LBS:

1. Public safety or emergency services
2. Consumer services
3. Enterprise services

LBS Components:

- LBS Middleware

This wraps access to Core LBS Features (Location Tracking, GIS Provider and Location Collection Services) to provide a consistent interface to LBS applications.

- Location Tracking

This component stores the location trace of individual users. This represents a fundamental
Component in next-generation LBS as it contains the data that allows a user's route to be determined and potentially predicted.

- GIS Provider

This component provides geospatial functionality for many LBSs including map information, map visualization and directory services. Google Maps with its API can be considered a GIS provider.

- Location Collection Service

This component performs location collection to get a latitude and longitude for a specific user.

- Location Manager

Location Manager Class of android is present to manage all other components needed to establish a LBS system.

- Location Manager

Location Manager Class of android is present to manage all other components needed to establish a LBS system [1][4][6].

II. DEPLOYMENT

Obtaining the Google Map API keys:

To create a full-fledged application using Google Maps API, the developer will have to obtain a unique API key from Google. Before this, Google requests a fingerprint that uniquely identifies your development computer. It is basically the MD5 fingerprint of the Signing certificate.
The value of the String resource named 'google_maps_api_key' in the string.xml file holds the unique key value.

Steps:

1. Install and configure the Google Play services SDK
2. Add the Google Play services version to your app's manifest
3. Get an Android certificate and the Google Maps API key
4. Display your app's certificate information
5. Create an API project in the Google APIs Console
6. Obtain a Google Maps API key
7. Add the API key to your application [7].

Fig. 4: Satellite and Map Views

Sending GPS data to an AVD with GPX files:

This enables testing the app without having an actual android device (through the AVD Emulator in Eclipse) to track. A file which contains GPS data/routes must be used in order to fulfil this. The data must be in GPS Exchange Format (.gpx extension).
The following steps need to be performed:

1. Once the app is up on the AVD, select Window -> Open Perspective -> DDMS to switch perspectives
2. Select your AVD in the devices
3. Click the GPX tab under the Emulator control tab (on the top panel)
4. Click the LOAD GPX button and select your file
5. Select your file and press the PLAY button to
6. Send the GPS data to the AVD and START

III. TECHNOLOGIES USED

This section presents the technologies that we used in our Route Tracker App

AndroidManifest.xml features:

1) Google Maps API is a non-standard API, and hence the library's name must be mentioned in the app's manifest with a uses-library element nested in the application-element.
2) Since most of our display is maps, we need to hide the title bar by using a standard Android theme. The attribute android: theme needs to be tweaked in the activity element
3) The complete list of permissions required when the user installs the app on the mobile device. If these permissions are not granted the app will not be installed.

Class Toggle Button:

This basically (package android. Widget) maintains an on-off state. Initially displays the text START TRACKING (On state). After the user selects it and presses it again, it toggles back to the off state – STOP TRACKING.

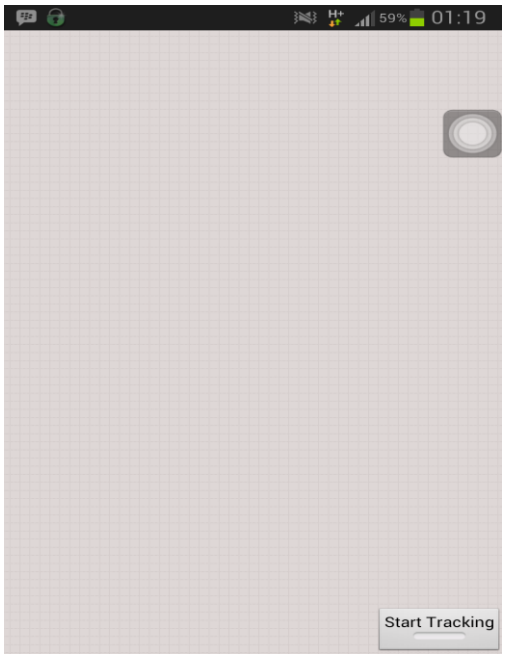

Fig. 5(a): App Home Page (Start Tracking)

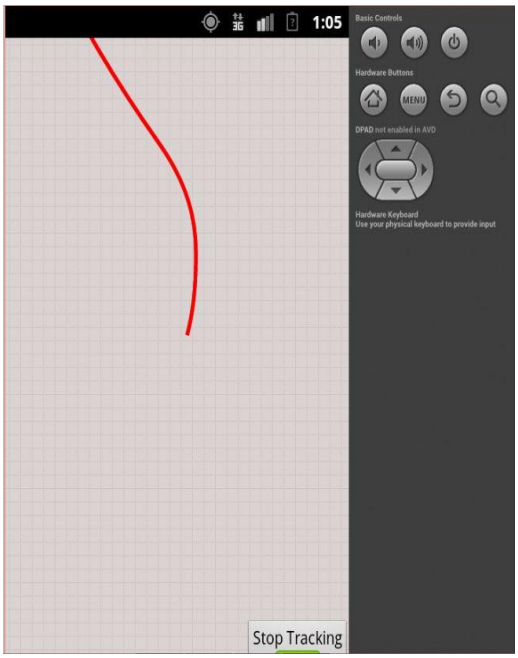

Fig. 5(b) The Route tracking starts

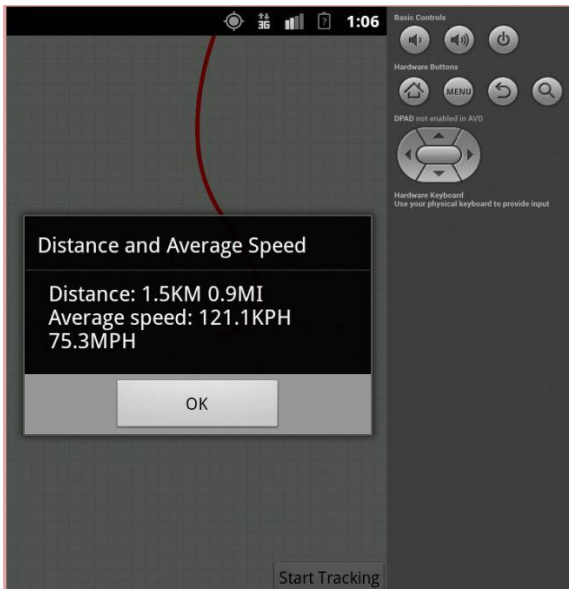

Fig. 5(c): Display of Speed and distance covered

Classes Map Activity, Map View and Overlay:

1) The package com.google.android.maps contains the classes that we need to interact with the Google Map API. Our main class is a subclass of MapActivity – an activity that manages a MapView for displaying the maps obtained.
2) To display data on a map view Overlay class is used.
3) GeoPoints class is used to translate GPS data into points that can be used to re-centre the map based on the user's location and to draw the route

Location Data:

1) Package android. Location contains the classes and interfaces for acquiring and using location data.

2) Location Manager: Access to device's location and services. Depending on the device various location providers may be supported.
3) A Location Provider can deliver requested updates to a Location Listener.
4) For receiving GPS data for tracking, we implement GpsStatus.Listener interface.

Programmatically determining the Device's Display Size:
Class Display (package android. View) provides access to the device's screen dimensions. We use these dimensions to help scale the maps as we rotate them to match the user's current bearing [4][7][8].

IV. ALGORITHM

1. START
2. RouteTracker subclass of MapActivity (RouteTracker.java -> Main class)
a. Override the method onCreate by FrameLayout, MapView and mapController classes.

Bearing Frame Layout = new Bearing Frame Layout(this, getResources().getString(R.string.google_maps_api_key));FrameLayout main Layout =(FrameLayout)findViewById(R.id.mainLayout);mainLayout.addView(bearingFrameLayout, 0);

b. Overriding Activity methods onStart and onStop
c. Location Listener call to Method updateLocation if (location != null && gpsFix) // location not null; have GPS fix

```
{
// add the given Location to the route
routeOverlay.addPoint(location);
// if there is a previous location
if (previousLocation != null)
{
// add to the total distanceTraveled
distanceTraveled   location.distanceTo(previousLocation);
} // end if
// get the latitude and longitude
Double latitude = location.getLatitude() * 1E6;
Double longitude = location.getLongitude() * 1E6;
// create GeoPoint representing the given Locations
```

d. Anonymous Inner Class LocationListener to respond to LocationManager Events
e. Anonymous Inner Class That implements GpsStatus.Listener to respond to GpsStatus Events
3. BearingFrameLayout subclass of FrameLayout
a. Method getChildLayoutparams to return LayoutParams object
b. Constructor class

```
mapView = new MapView(context, apiKey); // create new MapView
mapView.setClickable(true); // allow user interactions with the map
mapView.setEnabled(true); // enables the MapView to generate events
mapView.setSatellite(false); // display map image
// set MapView's layout mapView.setLayoutParams(getChildLayoutParams());
addView(mapView); // add MapView to this layout
```

c. Overriding View method DispatchDraw
4. RouteOverlay subclass of Overlay

a. Constructor

```
pathPaint = new Paint();
pathPaint.setAntiAlias(true);
pathPaint.setColor(Color.RED);
locations = new ArrayList<Location>(); // initialize points
positionPaint = new Paint();
positionPaint.setAntiAlias(true);
positionPaint.setStyle(Paint.Style.FILL);
```

b. Overriding Overlay Method Draw reset the Overlay for tracking a new route

Draw this Overlay on top of the given Map View for each Location convert Location to Geo Point if (previous != null) get Geo Point convert Geo Point add new point end if move to next location end else END

V. APPLICATION METRIC ANALYSIS

In this section we have basically done a complexity analysis of the algorithms/classes implemented for getting the application up and running. We have tested the application on various parameters and presented the results below, class wise.

Table I: Class Complexity Analysis (a)

Metric	Total	Mean	Std. Dev.
➢ Cyclomatic Complexity		1.2	0.4
➢ BearingFrameLayout.java		1.2	0.4
- Dispatch Draw	2		
- get Child Layout Params	1		
- Bearing Frame layout	1		
- Set Bearing	1		
- Get Mapview	1		
➢ Number of parameters		0.8	0.748
➢ Nested Block Depth		1.2	0.4
➢ Depth of Inheritance Tree		4	0
➢ Weighted methods per Class	6	6	0
➢ Lack of Cohesion methods		0.75	
➢ Number of attributes	3	3	0
➢ Number of methods	5	5	0
➢ Number of classes	1		

Table II: Class Complexity Analysis (b)

Metric	Total	Mean	Std. Dev.
➢ Cylcometric Complexity		1.615	0.923
➢ RouteTracker.java		1.714	1.161
- Update Location	4		
- On Option Item Selected	3		
- On Create	1		
- On Start	1		
- on Stop	1		
- is Route Displayed	1		
- on Create Option Menu	1		
- on Location Changed	2		
- on Provider Disabled	1		
- on Provider Enabled	1		
- On status Changed	1		
- On Gps Status Changed	2		
- On Checked Changed	2		
➢ Number of Parameters		1	0.784
➢ Nested Block Depth		13.08	0.606
➢ Depth of Inheritance		2.25	2.165
➢ Weighted Methods per Class	21	5.35	4.085
➢ Lack of Cohesion of Methods		0.228	0.394
➢ Number of attributes	14	3.5	6.062
➢ Number of Static Attributes	3	0.75	1.299
➢ Number of Methods	13	3.35	2.487
➢ Number of Classes	4		
➢ Total Lines of Code	209		

Table III: Class Complexity Analysis (c)

Metric	Total	Mean	Std. Dev.
➢ Cyclometric Complexity		1.75	1.299
➢ RouteOverlay.java			
- Draw	4		
- Route Overlay	1		
- Add Point	1		
- Reset	1		
➢ Number of Parameters		1	1.225
➢ Nested Block Depth		1.5	0.866
➢ Depth Of Inheritance Tree		2	
➢ Weighted methods per Class	7	7	
➢ Lack of Cohesion of Methods		0.583	
➢ Number of Attributes	4	4	
➢ Number of Methods	4	4	
➢ Number of classes	1		
➢ Total lines of code	77		

VI. DEPLOYMENT CONSTRAINTS

1. Technology Constraints

For LBS to be operational on a large scale, mapping under the geographical information system (GIS) needs to be more comprehensive than it is today. This raises significant challenges for improving the breadth and the depth of the existing coverage of GIS. The most important factor in enabling the growth of LBS is wide availability of cheap GPS enabled handsets. GPS enabled handsets are being manufactured now days. The issue of cost remains to be tackled, since these phones are still all high-end units.

2. Infrastructure Constraints

One of the main problems is the lack of spread of the wireless network into the countryside. In developing country like India, the wireless technology is in very nascent stage. In metro cities and areas, the problem of network congestion is also an important issue. The percentage of service operators not meeting the congestion rate benchmarks has risen substantially.

3. Market failure

One of the main constraints to the provision of value added services, in general, and LBS in particular, is the market structure of the mobile industry and the failure to unleash the forces of competition. A key essential need for LBS provision needs cross-network connections to be seamless, and the current practices go against a cooperative attitude for LBS provision.

VII. CONCLUSION

In this paper we basically delved into the details of what Location based services actually are, their architecture and why it is important for the generation of today. We also developed a basic support application 'Route Tracker' implementing LBS, Google Maps and Route Tracking. This application basically deals with enabling the users to track their movements and get them displayed as a line. Developing the app required the use of many internal as well as external technologies. Various classes from packages such as 'com.google.android' and 'android. Location' were used. Location management classes and display classes were used for communication and display. The complexity of every implemented class is a good indication of the development efforts. All in all, implementation of LBS for mobile involves an interesting combination of basic native Android development as well as the working and integration of external Google API's.

VIII. FUTURE WORK

1. Photo Tagging:

LBS can be used to tag the photos clicked from camera and can be showed on the map on the location where they were clicked.

2. GPS device:

Extending the application with some distance algorithms like Dijikstra shortest path and including more parameters like speed, distance, it can used for as a GPS.

ACKNOWLEDGMENT

This research paper is made possible through the help and support from many people, and in essence, all sentient beings. Especially, please allow me to dedicate my acknowledgment of gratitude towards our college Librarian Mr. Pradip Das and his team. I would like to thank Mr. Anand Gawadekar of the NMIMS IEEE Committee due to which we could get all requested references seamlessly without any trouble.

Finally, a sincere thanks to our college and Computer Science department for allowing us to enter the field of research in our so important final year of B.Tech.

REFERENCES

[1]. Xianhua Shu, Zhenjun Du, Rong Chen," Research on Mobile Location Service Design Based on Android" Dalian, China

[2]. D'Roza, T., and Bilchev, G. An overview of location-based services. BT Technology Journal 21, 1 (2003),20_27

[3]. Schwinger, W., Grin, C., Prll1, B., and Retschitzegger, W. A light-weight framework for location-based services. In Lecture Notes in Computer Science (Berlin, 2005), Springer, pp. 206_210

[4]. Zeimpekis, V., Giaglis, G., and Lekakos, G. A taxonomy of indoor and outdoor positioning techniques for mobile location services. SIGecom Exch. 3, 4 (2003), 19_27

[5]. Wei Duan ; Jianzhang Ma ; Chenhui Wang,"The research of Android System architecture and application programming", Harbin, 24-26 Dec. 2011, Computer Science and Network Technology (ICCSNT), 2011 International Conference on (Volume:2)

[6]. John, N. ; Udayakumar, S. ; Gupta, R," Generic framework for mobile application development", Cisco Syst. India (P) Ltd., Bangalore, India, 4-6 Nov. 2011

[7]. Android.http://en.wikipedia.orglwiki/Andr oid[EB/OL]. 2009.1 O

[8]. Web 2.0 Summit 2009: Evan Williams and John Battelle "A Conversation with Evan Williams", O'Reilly Media, October 21, 2009