Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 1, January 2015, pg.205 – 210

RESEARCH ARTICLE

Personal Service Areas for Location-Based Wireless Web Applications on Android Platform

Bhawana D.Sarode[1], Prof. P. P.Karde[2]

[1]Department of Computer Science & Information Technology, HVPM COET, M.E First year, Amravati, India

[2]Department of Computer Science, HVPM COET, Amravati, India

Abstract: Location-based wireless services enable mobile users to access Web-based information about resources in their immediate vicinity. The primary focus is to get familiar to Android to provide a featured GUI which contains light weight software for buyer's to search for property listings specific to their needs. The application also provides a drag and drop and zoom control to save a list of selected property listings while browsing other options on the property. The main emphasis lies in providing a user-friendly G-map for effectively showing the desired results on the GUI. We review mobile location area concepts and describe a context-sensitive algorithm for delineating a mobile user's personal service area. We have vast number of applications and usage where a person sitting in a roadside café needs to get relevant data and information. Such needs can only be catered with the help of LBS. By using this system we can easily manage property for buyers and sellers.

Keywords: Android, GPS, LBS, Google Maps, location-based services, Web services, mobile Web

1. Introduction

With the advent of mobile users' location tracking capabilities, location-based wireless services are positioned as a major motive for the development of future wireless systems.

Now a day, communication technology plays an important role in person's day today life. With mobiles becoming fashionable, more popular and more trending, they are increasingly prone to the theft. Location based service (LBS) is emerging as a killer application in mobile data services thanks to the rapid development in wireless communication and location positioning technologies. While location is a key element of the user context, other applicable elements that affect a location-based service. Users can flexibly access, control and process the free Google map and implement location based mobile service in his mobile systems at low cost. Android platform will not only promote the technology (including the platform itself) of innovation, but also help to reduce development costs, and enable developers to form their mobile systems with unique characteristics.

2. Preliminary

An important aspect of context -awareness is the system's ability to deliver information that is personalized.

A. APPLICATIONS

A set of core applications are on the top level in the framework, including an email client, a SMS app, a calendar, a maps-application, web browser, contacts-app, and many more. All apps are written using the Java programming language.

B. *LIBRARIES*

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework.

C. ANDROID RUNTIME

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Every Android application runs in its own process given by the OS, and owns its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM is executing files in the .dex (Dalvik Executable) format which was optimized for minimal cpu-and-memory-usage. The Virtual Machine is register-based, and runs classes compiled by a Java language compiler that have been transformed at compile-time into the .dex format using the "dx" tool, that are shipped with the SDK. The Linux Kernel can run multiple instances of the Dalvik VM, also providing underlying functionality such as threads and lowest-level memory management.

3. Proposed System

A. ACTIVITY

Activity is the most common one of the four Android building blocks. An activity is usually a single screen in your application. Each activity is implemented as a single class that extends the Activity base class. Your class will display a user interface composed of Views and respond to events. Most applications consist of multiple screens. For example, a text messaging application might have one screen that shows a list of contacts to send messages to, a second screen to write the message to the chosen contact, and other screens to review old messages or change settings. Each of these screens would be implemented as an activity. Moving to another screen is accomplished by a starting a new activity. In some cases an Activity may return a value to the previous activity – for example an activity that lets the user pick a photo would return the chosen photo to the caller.

When a new screen opens, the previous screen is paused and put onto a history stack. The user can navigate backward through previously opened screens in the history. Screens can also choose to be removed from the history stack when it would be inappropriate for them to remain. Android retains history stacks for each application launched from the home screen. Android uses a special class called Intent to move from screen to screen. Intent describes what an application wants done. The two most important parts of the intent data structure are the action and the data to act upon. Typical values for action are MAIN (the front door of the application), VIEW, PICK, EDIT, etc. The data is expressed as a Uniform Resource Indicator (URI). For example, to view a website in the browser, you would create an Intent with the VIEW action and the data set to a Website-URI. There is a related class called an IntentFilter. While an intent is effectively a request to do something, an intent filter is a description of what intents an activity (or intent receiver, see below) is capable of handling. An activity that is able to display contact information for a person would publish an IntentFilter that said that it knows how to handle the action VIEW when applied to data representing a person. Activities publish their IntentFilters in the AndroidManifest.xml file.

Navigating from screen to screen is accomplished by resolving intents. To navigate forward, an activity calls startActivity (myIntent). The system then looks at the intent filters for all installed applications and picks the activity whose intent filters best matches myIntent. The new activity is informed of the Intent, which causes it to be launched. The process of resolving intents happens at run time when start Activity is called, which offers two key benefits: Activities can reuse functionality from other components simply by making a request in the form of an Intent. Activities can be replaced at any time by a new Activity with an equivalent.

B. INTENT RECEIVER

You can use an Intent Receiver when you want code in your application to execute in reaction to an external event, for example, when the phone rings, or when the data network is available, or when it's midnight. Intent receivers do not display a UI, although they may display Notifications to alert the user if something interesting has happened. Intent receivers are also registered in AndroidManifest.xml, but you can also register them using Context.registerReceiver method. Your application does not have to be running for its intent receivers to be called;

the system will start your application, if necessary, when an intent receiver is triggered. Applications can also send their own intent broadcasts to others with Context .broadcast Intent method.

C. SERVICE

A Service is code that is long-lived and runs without a UI. A good example of this is a media player playing songs from a play list. In a media player application, there would probably be one or more activities that allow the user to choose songs and start playing them. However, the music playback itself should not be handled by an activity because the user will expect the music to keep playing even after navigating to a new screen. In this case, the media player activity could start a service using the Context.startService method to run in the background to keep the music going. The system will then keep the music playback service running until it has finished. Note that you can connect to a service with the Context.bindService method. When connected to a service, you can communicate with it through an interface exposed by the service. For the music service, this might allow you to pause, rewind, etc.

D. CONTENT PROVIDER

Applications can store their data in files, a SQLite Database , preferences or any other mechanism that makes sense. A content provider, however, is useful if you want your application's data to be shared with other applications. A content provider is a class that implements a standard set of methods to let other applications store and retrieve the type of data that is handled by that content provider.

E.*GOOGLE MAP IN ANDROID:*

Android provides a number of objects to handle maps in LBS system like Map View which displays the map. To handle this a Map Activity class is there. To annotate map it provides the overlays class. Even it provides canvas by which one can easily create and display multiple layers over the map. Moreover, sufficient provisions are there to zoom the map, localize the map by means of MapController. Following code-line shows the Map Handling in Android:

```
<com.google.android.maps.MapView
android:id=”@+id/map_view”
//specify different attributes/>// map controller
MapController mapController = myMapView.getController();
mapController.setCenter(point);
```

4. Mobile Location Areas

Flexible map display and control functions and location support are provided in Android for mobile system design. This section analyses Map View, Map Activity and Location- Based API on Android, and presents a simple design example to show the location-based mobile service design on Android. Map View is used to display a view of the map. It can accept the keyboard events such as on Key Down and on Key Up to support the map movement and the zoom feature. It also supports multi-layer Overlay and user can draw coordinates, pictures and strings on the map. Map View is set up only by Map Activity. Because Map View uses the file system and network in the background, all of these threads are in the control of the Activity life cycle. Android defines UI by the layout. Map View is required to be added into the layout, as follows:

```
<com.google.android.maps.MapView
android:”@+id/myMap”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”
android:anabled=”true”
android:clickable=”true”
android:apiKey=”application apikey”/>
```

Other necessary views can then be added, such as an

Edit Text to enter an address and a button to carry out queries as shown in Figure 1.

Figure 1. Address search interface.

We then use the Android Location-Based API to collect user current position and display that location on the screen, and use Google Maps to display the current user location on the cell phone.
Now create a Location Manager from which we can get the coordinate values
LocationManager lm=(LocationManager)
getSystemService(Context.LOCATION_SERVICE);

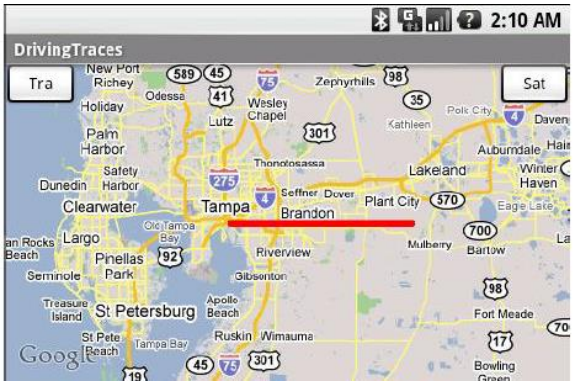

Figure 2. Driving trace result

We can also use setZoom (int) to zoom out the map to the level needed, such as zooming out one more level as follows:
myMapController.setZoom (myMap.getZoomLevel ( )-1);
Map view types can be switched, e.g. switching to the satellite view and hiding the current traffic view, such as:
myMap.setSatellite (true);
myMap.setTraffic (false);[2]

5. Personal Service Area

For a traveling tourist, a key issue is how to find out her relevant surrounding region, as this will affect the choice of local resources displayed on her terminal. This local service area is determined by several factors that are either obtained directly or calculated from the user's context. Besides location, additional mobility elements of context that affect the displayed information include direction and speed of travel since, for a given travel time, fast moving users can reach places that are further away. CATIS, tracks a user's previous locations and from these the system can compute the user's direction of travel as well as average speed. We devised a slidinggrids window methodology in which location, direction and speed of travel are taken into account to deliver a list of recommended points of interest (POIs). For example, POIs in the opposite course of travel, and beyond a certain distance threshold, will not be listed. Our sliding-grids window algorithm uses a two-step approach whereby first the tourist's pertinent service area is determined and then a list is generated based on a preferred maximum distance of travel to a POI. This approach relies on detecting the user's direction of travel and leveraging a pre-partition of the metropolitan area into a grid of square-shaped cells where the user is traveling as shown in Figure 2. The size of the cells is pre-defined so that the time to traverse a cell is about the time the user is willing to travel to a listed POI at a given speed (e.g., half an hour). The cell's size will determine the number of cells that need to be retrieved from a Web service following a

user query. In Figure 2, two successive queries are issued by the user, first at *Time 1* and then at *Time 2*, and the shaded cells are the corresponding retrieved cells. The *personal service area* is derived by the intersection of the retrieved cells with a circle that represents the maximum distance that the user is willing to travel to a listed POI. This algorithm proves to be more efficient compared to an exhaustive computation of distance over all POIs, and also adapts the list of relevant POIs to the user's direction of travel.

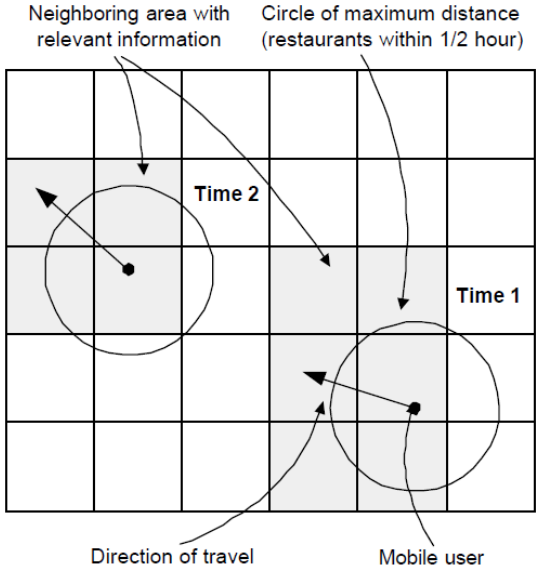

Figure 2 Personal service area based on direction, speed, and distance to resource

6. Application Of LBS

Location-based services or LBS refer to a set of applications that exploit the knowledge of the geographical position of a mobile device in order to provide services based on that information.
They can be classified in three categories:

A.PUBLIC SAFETY OR EMERGENCY SERVICES*:*
Since the location of the subscriber can be provided by the carrier, the mobile phone is a valuable access point in the times of emergency. In the US, Europe and Japan, it is mandatory by law for carriers to be able to provide such information.

B. CONSUMER SERVICES**:**
i. Navigation – users get route maps to a particular destination, real time traffic routing that takes into account actual congestion patterns etc.
ii . Location based advertising – advertisements of discounts or offers from a store as the user comes within the vicinity.
iii. Location based reminders – users can enter in to-do lists, whose location information is activated when the user passes by, for instance, pick up shopping or laundry etc.
iv. Family and friend finder – allows users to keep track of the location of their children, relatives or friends, with the informed consent of these subscribers.
v. Location based search – allows users to access local services, or find even more detailed information such as listings and ratings of movies playing in theaters nearby etc.
vi. Location based mobile gaming which began a decade ago has larger scope now as positioning and handset technology have improved considerably.
C. *ENTERPRISE SERVICES:* LBS enables firms in fleet and asset tracking, field service dispatching, route and delivery optimization, and mobile workforce management. This has proved to be extremely useful for small and medium businesses.[1]

7. Conclusion

The feature of location based service is emphasized on Android platform. One can integrate a fully zoom and drag enabled map by adding just few lines in the Java code and XML code of the Android-Default-Application. Through the research on Android architecture and application development, and from the design method and results of an application example in this paper, the availability and performance of the platform is verified and the design result also shows the easiness to implement self-location, to draw the driving trace, to perform queries and to flexibly control the real-time map on Android. The actual system also achieves high running performance. The future work is to design a more powerful mobile location-based system featured with more unique customized functions based on Android.

Acknowledgements

This paper has benefited from conversations with many different people – far more than can be acknowledged completely here. Still we would like to particularly thank, HOD of CS&IT department for his support guidance and support.

References

[1] Amit Kushwaha, Vineet Kushwaha,"Location Based Services using Android Mobile Operating System" 1Department of Electronics & Communication Engineering IIMT Engineering College, Meerut-250001, India 2Department of Information Technology Indian Institute of Information Technology, Allahabad- 211012, India

[2] G. Abowd et al., "Cyberguide: a Mobile Context -Aware Tour Guide", Wireless Networks 3, pp. 421-433, 1997.

[3] Research on Mobile Location Service Design Based on Android Xianhua Shu, Zhenjun Du, Rong Chen School of Information Science and Technology Dalian Maritime University Dalian, Chinxiansimba@163.com

[4] C. Haseman ,Android Essentials, PDF Electronic Book, 2008. Available from: http://androidos.cc/dev/index.php.

[5] N. Gramlich, Android Programming , PDF Electronic Book, 2008. Available from: http://androidos.cc/dev/index.php.

[6]Virrantaus, K., Markkula, J., Garmash, A., Terziyan, V., Veijalainen, J., Katanosov, A., and Tirri, H. Developing gis supported location-based services. In Web Information Systems Engineering (2001), IEEE

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

