DEVELOPMENT OF LOCATION BASED SERVICES

White-paper 1.0 to give you a kick-start in your LBS project

Editor: Rikard Windh, Spring 2010
Introduction

Development of Mobile Services

The Location Based Idea

Positioning

Positioning Request Methods

Mobile Devices

Server-side

Content Providers

Summary

Acknowledgments

Appendix

More on Network-based positioning

More on Cell-Id based positioning
DEVELOPMENT OF LOCATION BASED SERVICES

A white-paper to give you a kick-start in your LBS project

Rikard Windh
Combain Mobile AB
Spring 2010

Introduction

Google Maps, Twitter, Layar, Gowalla, Brightkite and Qype. The number of location-based applications are now rapidly increasing. The purpose of this white-paper is to give you a kick-start in the development of your LBS project. But what is LBS? A location-based service (LBS) is an information and entertainment service, accessible with mobile devices through the mobile network and utilizing the ability to make use of the geographical position of the mobile device.\(^1\) This means that you have a mobile device that is aware of its location and utilize this when accessing an internet based service. LBS applications can be used for navigation, information sharing, tracking, location based games, emergency services, location based advertising and much more\(^2\). A popular category within LBS is Location-based Social Networks. These services builds up a community with users which based on the location interacts and get connected to each other. According to a report from Garner, the number of users of location-based services was estimated to 95.7 million during 2009 and which should result in over $2 billion in revenue.\(^3\) In a new report from Juniper Research\(^4\) suggests revenue from location-based services (LBS) are expected to surge to $12.7bn in 2012.\(^5\) This means a hugh estimated growth of the LBS market. This white-paper will help you to take part of this growth.

\(^1\) [http://en.wikipedia.org/wiki/Location-based_service](http://en.wikipedia.org/wiki/Location-based_service)
Development of Mobile Services

Developing location-based services builds mostly on the same principals as for other mobile services. You need a server-side solution as the backbone of your service, and you need applications or clients for your users to interact with the service. The special feature with LBS is that you have an additional valuable property to consider, the geographical position of the end-user. The context, the scope of information or the services you expose to the end-user can be based on, or enhanced by the fact that the services knows where you are. If you search for a taxi in your mobile web browser, you most likely would like to have the number to a local taxi company, not the number to the most popular taxi company in the world.

The Location Based Idea

Before the time of mobile handsets, you always knew Where you called. You called a fixed line. You knew the address of where the phone in the other end was located. Today, very often, the first question over a mobile phone is, Where are you? The location based services will now after 15-20 years give us the location again. And this time it is not just a fixed location. The location is mobile and follow you and your friends along your way. Before you call the person you can see where he or she is. You might also call your friend because you see he is at a certain location. “Hi! I see you are at Starbucks. I will come down and share a cup of coffee with you.” The knowledge of the mobile location is given us a paradigm of new services. How can you as a developer leverage on that? When the car was invented - the first cars looked like
wagons. We took the car engine and placed it on a wagon. The first location based services are very map centric. We take the position of you and your friends and place it on a map. You can do more. To get inspiration, you can find at bdnooz.com\(^6\) a list of over hundreds of links to Location Based Social Networks. In Yahoo’s Fire Eagle Application Gallery\(^7\) you find more than seventy location based applications. Check out Layar.com and get inspired by augmented reality. Read through WebServices provided by Geonames.org and API’s and mashups listed at programmableweb.com to see if you get some new ideas. When you have specified your idea, it is time to consider the implementation.

**Positioning**

First, you need to obtain the location of the mobile device. This can be done by various positioning methods. These methods can be classified into two groups: network-based positioning and terminal-based positioning\(^8\). In network-based positioning the network infrastructure locates the mobile device. In terminal-based positioning the location is obtained by the mobile device itself. Examples of network-based positioning is when mobile operators locates a mobile phone based on which cell-tower it is hooked to. The mobile device is just turned on and are has no active role in the calculation of the position. In the opposite method, terminal-based positioning, the device has a more central role. GPS for example, is a common example of terminal-based positioning. The device calculates the position based on received signals from the GPS satellites. Other terminal-based positioning methods are Cell-id, Wi-Fi, Bluetooth, NFC\(^9\) and 2D Barcodes\(^10\). The mobile device detects the unique ids from Cell towers, Wi-Fi or Bluetooth networks and over an internet connection look-up these id’s and get positioning information in return. The NFC and 2D Barcode tags can include the positioning information in the tag, or have it looked up over an internet connection.

**Positioning Request Methods**

The position request can be trigged from an external source of from the mobile itself. If it is trigged from an external source it is called Mobile Terminated Location Request (MT-LR). This method can also be referred to as a pull-method or network-based positioning as mentioned above. The external source has none, or limited information and must continuously perform location requests to track the mobile device. It also be secured that the external source has the legal right to locate the mobile device. When the mobile itself performs the requests, as in terminal-based positioning, it is called Mobile Originated Location Request.

---


\(^7\) [http://fireeagle.yahoo.net/gallery/all?page=1](http://fireeagle.yahoo.net/gallery/all?page=1)


The legal issue is easier here since the mobile itself is in control. The mobile user is most likely running an application in the phone which is doing the requests. A third very interesting method is called Spatial Triggers. Triggers can be stored in the mobile operators location based systems. When certain conditions is being met, the triggers are fired. Example of triggers are: 1 - when a mobile phone leaves a certain area, 2 - when a mobile phone enters a certain area, 3 - when two defined mobile phones comes into the same area. By defining such triggers, you do not need an external services pulling, or requesting positions continuously. Which one of these methods shall you as a developer choose? You have to consider the methods different performance in accuracy and their difference level of deployment and scalability.

Network-positioning via mobile operators, does not require any application in the mobile device, but has limited accuracy and requires an agreement directly with the mobile network operator or via an aggregator that provides access to several operators. One such aggregator is for example Ericsson IPX. Ericsson Labs is providing a non-commerical network-positioning api they call Web Location.

Spatial triggers sounds promising but the usage of spatial triggers is still limited. It is a technical feature in some positioning systems the mobile operators are using, but it seems that the operators are not yet offering this as a service available for third party developers.

GPS positioning has good accuracy outdoor, but limited accuracy indoor and does require a GPS enabled handset. Continuously use of GPS positioning drains the battery in a mobile phone pretty fast.

Cell-Id has good indoor coverage and does not drain the battery, but limited accuracy compared to GPS and requires access to a cell-id database. Wikipedia provides a list of public cell-id databases. The largest database according to the list holds more than 5.6M cells.

Wi-Fi has better accuracy than Cell-Id, but the penetration of wifi enabled handsets are limited. Requires access to a wifi database. Skyhook Wireless is claiming they have a database of 100 million Wi-Fi spots. WiGLE claims to have more than 20 million wifi cells in their database.

The NFC (near-filed-communication) tags can give a very exact positioning information. By storing in a database, or in the tag itself, where the tag is placed, you will know where the mo-

---

12 http://www.ericsson.com/solutions/ipx/
13 https://labs.ericsson.com/apis/web-location/
15 http://www.skyhookwireless.com
16 http://wigle.net/
bile user are located when he or she is reading the tag. It requires that the user holds the mobile device close to the tag. Right now is the number of NFC enabled handsets limited but it can be a mass-market technology within a couple of years.

2D Barcodes is from a positioning perspective same as NFC, but instead of having an NFC enabled device, you use the camera in your mobile device to scan the tag. The penetration of camera phones is high but you need an application in your phone which can read the tags.

A hybrid positioning solution combines the methods above to get the best performance based on the current situation. A turn-by-turn navigation application will require the accuracy of a GPS, but the cell-id accuracy is good enough for a location based social network application as Gowalla. Wifi for automatic indoor positioning and NFC or 2D Barcodes can be used when more exact positioning is required and the use case accept the active handling of tagging.

Source: LBS Lecture Notes Steiniger 2006
Your users are mobile and they expect to be able to interact with your service when they are on the move. You need to support one or several mobile platforms. The more platforms you support, the higher number of end-users you can reach. BUT, it will require more development, support and maintenance resources from you. The latest mobile metrics report from AdMob shows that smartphones has passed feature phones in traffic share. For smartphones iPhone has 50% share, Android 24% and Symbian 18%.

Source: AdMob.com, Feb 2010.

When you develop solutions for the various platforms, you can choose to develop application designed for the operating-system on the platform, or use multi-platform compatible standards, such as SMS, MMS and mobile web applications. As an example, native applications written for iPhone, can not execute on an Android platform. But a mobile website hosted on your server-side platform can be viewed and executed both by the browser on an iPhone and on an Android phone. SMS messages sent from the server-side can be received by all mobile platform independent on platform.

As a general thumb-rule, always consider a basic mobile website so you have coverage on all platforms. Develop native applications to gain a richer user-experience on selected platforms. The number of platforms you support, is dependent on your target group and on your budget.

**Mobile Devices and Location**

iPhone and Android has built-in support for a hybrid positioning method combining GPS, Cell-Id and WiFi. The API’s are available when building apps for these phones. When build-
ing web applications, there is an interesting API coming up from W3C. Mobile browsers supporting this standard will make the location of the handset available for mobile web applications. I.e. no need to run a native app to get access to the GPS.

**Android**

In Android you have excellent support to work with location. In Android there is specific location package called android.location which are proving API to be used. You also have the possibility to incorporate the external Google Maps library into your solution. The location class has easy to use functions to read out current latitude and longitude coordinates. The Android platform is using a hybrid positioning method to determine the position. If within GPS coverage the GPS is used, otherwise the platform can use found GSM/3G or wifi cell to get the position. Certain criterias can be defined for selecting positioning method. You as a developer can select if you need the high accuracy or instead want low power consumption and can accept lower accuracy.

The usage of cell-id is normally handled within the location class. You as a developer do not have care about that on Android. But if you still want, Ericsson Labs has written a tutorial for this called: Create a Simple Cell-id Look-up Application for Android.

**iPhone**

For iPhone and development of iPhone apps, you have to register to Apples developer program to get access to the Location documentation. But the general principal is similar to how it works on Android. You as a developer gets a geographical position from the API's. Under the hood, the iPhone can have used GPS or cell-id or wifi to determine the position.

For iPhone and development of location aware mobile web applications, you need to be aware of the Geolocation API. With this API you are able to read out the position directly from the mobile web site. A javascript is executed and retrieves the location of the phone. A simple test is to browse to [http://bemoko.com/blog/iphonegeo](http://bemoko.com/blog/iphonegeo) from your iPhone and you can see how it works. Your position on a map without a single native app. Just a Geolocation API script on a mobile web page and a Geolocation API enabled Safari browser. A common misunderstanding is that you need a mobile application to read our data from the GPS in the phone.

---

17 [http://dev.w3.org/geo/api/spec-source.html](http://dev.w3.org/geo/api/spec-source.html)
Symbian

At Nokia forum you can read how to make Google Maps using Location API in Symbian\(^\text{22}\).

JavaME

For Java developers the Location API is defined by the JSR179 specification\(^\text{23}\).

From the description of the JSR: *This specification defines a J2ME Optional Package that enables mobile location-based applications for resource limited devices. The API is designed to be a compact and generic API that produces information about the present geographic location of the terminal to Java applications. This API covers obtaining information about the present geographic location and orientation of the terminal and accessing a database of known landmarks stored in the terminal.*

\[
\begin{verbatim}
// Set criteria for selecting a location provider:
// accurate to 500 meters horizontally
Criteria cr = new Criteria();
cr.setHorizontalAccuracy(500);

// Get an instance of the provider
LocationProvider lp = LocationProvider.getInstance(cr);

// Request the location, setting a one-minute timeout
Location l = lp.getLocation(60);
Coordinates c = l.getQualifiedCoordinates();

if (c != null) {
    // Use coordinate information
    double lat = c.getLatitude();
    double lon = c.getLongitude();
}
\end{verbatim}
\]

Picture: Usage of the Location API, JSR179, in JavaME\(^\text{24}\).

\[^{22}\text{http://wiki.forum.nokia.com/index.php/Google_Maps_using_Location_Api_in_Symbian}\]
\[^{23}\text{http://jcp.org/aboutJava/communityprocess/final/jsr179/index.html}\]
\[^{24}\text{http://developers.sun.com/mobility/apis/articles/location/}\]
At Location-API.com you can read how to obtain locations by using Cell-ID in JavaME.25

Windows Mobile

On Windows Mobile platforms you will use GPS Intermediate Driver26 to determine the position. The GPS Intermediate Driver is useful to developers writing applications that use GPS (Global Positioning System) devices as well as to GPS hardware manufacturers. It is useful because it provides an intermediate layer that abstracts the actual GPS device from developers and manufacturers. Ideally, this enables application developers to write code that works with any GPS hardware, and GPS device manufacturers to produce hardware that works with any application.

SERVER-SIDE

You need a server-side solution for your location based service. The server-side solution will be the back-bone, the central database holding information about your users and their activities. You need a server to host your solution. Here is a list of hosting companies.27 Choose one which is inline with your preferences. You are probably or Microsoft, a Java or a Linux fan. You can choose from building from scratch or from building on top of an existing Content Management System (CMS). Several hosting companies is providing Wordpress, Joomla or Drupal. Here is a list Drupal hosting companies28. You can also build your service on top of an existing platform for location based solutions, such as SVCNGR29 or Yahoo Fire Eagle30.

CONTENT PROVIDERS

Now you are almost ready with your LBS project. You have your fantastic idea, you have developed an application which can determine the position of the mobile device. You have the latitude and longitude coordinates. But what now? You might need access to content related to the current position. One very common task is translate the coordinates to an address. In Android this functionality is provided by the platform, but for other platforms you need to look-up the address in an external database. One such database is hosted at Geonames.org. Below you some more examples of providers of location enabled content.

Geonames31

---

Geonames provides WebServices to look-up addresses, place-names and streets. You can also look-up nearby weather and nearby Wikipedia entries.

<table>
<thead>
<tr>
<th>Method</th>
<th>Format</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>findNearby</code></td>
<td>XML, JSON</td>
<td></td>
</tr>
<tr>
<td><code>findNearbyPlaceName</code></td>
<td>XML, JSON</td>
<td></td>
</tr>
<tr>
<td><code>findNearbyPostalCodes</code></td>
<td>XML, JSON</td>
<td></td>
</tr>
<tr>
<td><code>findNearbyStreets</code></td>
<td>US-only</td>
<td>XML, JSON</td>
</tr>
<tr>
<td><code>findNearbyStreetsOSM</code></td>
<td>XML, JSON</td>
<td></td>
</tr>
<tr>
<td><code>findNearbyWeather</code></td>
<td>Note: 1</td>
<td>XML, JSON</td>
</tr>
<tr>
<td><code>findNearbyWikipedia</code></td>
<td>XML, JSON</td>
<td></td>
</tr>
<tr>
<td><code>findNearestAddress</code></td>
<td>US-only</td>
<td>XML, JSON</td>
</tr>
</tbody>
</table>

Picture: Example of WebServices at Geonames.org

**Panoramio**

This Google picture service is proving a WebService delivering nearby photos. You send in the coordinates and Panoramio replies with links to nice images taken in the area.

**Twitter**

Twitter has several API functions related location. When you tweet, your current location can be stored and connected to the tweet. Other API's can then search and find tweets related to certain locations. You can read out twitter trends by location and you can find other Twitter users in your proximity.

**Yahoo**

- Maps and location enabled search for Weather, Traffic and much more.

**Flickr**

- Submit a position and get photos taken nearby that location.

**Google Maps API**

- API to use maps in your service. Free to use if your service is freely accessible for end-users. If not, Google is providing a Premier service.

**Open Street Map**

- maps and geodata shared under a Creative Commons Attribution-ShareAlike 2.0 license.

---

33 [http://apiwiki.twitter.com/Twitter-API-Documentation](http://apiwiki.twitter.com/Twitter-API-Documentation)
35 [http://www.flickr.com/services/api/flickr.photos.geo.photosForLocation.html](http://www.flickr.com/services/api/flickr.photos.geo.photosForLocation.html)
Summary
Mobile positioning is an enabler for a new paradigm of services. The mobile devices are getting mature and a wide range of phones is providing necessary functionality to develop nice applications and services. Out on the internet you find various resources of positioned enabled content you can use to enhance your service. Use a combination of positioning methods to get best overall performance. Do not hesitate to drop me a mail if you have any comments or questions. Have fun!

Rikard Windh
Combain Mobile AB
rikard@combain.com

Acknowledgments
This White Paper on LBS is the first edition of what we hope will be a yearly series with updated technical information on Location Based stuff. This paper is brought to you with compliments from Nordic Game Resources, eLBeS, Future Position X, Mobile Heights and Net-Port.

Thanks to Lars Andersen, Niklas Agevik and Karl-Petter Åkesson for the valuable feedback and inspiration.
Appendix

More on Network-based Positioning

As mentioned above, one advantage of network-based positioning is that the mobile device just need to be turned on. The location determination is performed by the network. Depending of which infrastructure, or mobile positioning system (MPS) the mobile network operator has, different Location Determination Techniques (LDT) will be used.

Location Determination Techniques

The less accurate network-positioning method is called Cell Global Identity (CGI). This method just gives which cell the mobile handset is connected to. The accuracy will depend on the size of the cell. If urban areas it can be 100 - 1100 meters. In rural areas the accuracy is lower since the cells are bigger.

The Cell Global Identity method can be improved by also measure the power level of the signal received from the handset. This method is called Enhanced Cell Global Identity (E-CGI). The accuracy is still dependent on the size of the cell but you know have an accuracy of around 50-550 meters in urban areas.

Instead of working with just one cell, you combine three cells in a method called Time of Arrival (TOA). And in this method the handset calculate the difference in time for the signals to travel from the synchronized base-stations to the handset. This gives an accuracy of around 125-200 meters.

An enhanced version of TOA is called Enhanced Observed Time Difference (E-OTD) gives an accuracy of 50-150 m. But this method requires special hardware in the handsets and it has not been widely used. A more common method is Uplink Time Difference of Arrival (U-TDOA) which does not require any special hardware or software in the phone. U-TDOA has been widely embraced by major U.S. GSM carriers to meet the government’s E-911 requirements. The accuracy of U-TDOA is dependent on the network layout but 50 meter or better is mentioned.

Assisted Global Positioning System (A-GPS) requires GPS enabled handsets but the GPS is assisted by functionality in the network to enhance performance and accuracy. It is of-

---

37 http://arxiv.org/abs/0911.0785
40 http://www.trueposition.com/web/guest/u-tdoa
41 http://www.3g.co.uk/PR/May2003/5332.htm
ten combined into a hybrid together with U-TDOA to cover indoor and non-GPS enabled handsets.

**More on Cell-ID based Positioning**

As mentioned above, Cell-id positioning requires access to a cell-id database. The database is needed to look-up the cell-id and translate it to a geographical position. But it also requires an application in the mobile phone which can read out the cell-id information from the platform. Each GSM/3G handset which is powered on and within coverage, will know the unique cell-id of the cell-tower it is connected to.

**Cell-id structure**

A mobile cell-tower can cover and be responsible for one or several cells. Each cell has a unique cell-id. The cell-id is actually a common name for four different parameters: MCC, MNC, CID, LAC. It is the combination of these four parameters that is unique for the cell.

- **MCC** - Mobile Country Code.  
  Example: 240 for Sweden

- **MNC** - Mobile Network Code  
  Example: 01 for the operator Telia in Sweden

- **LAC** - Location Area Code  
  Example: 1397 for an area of Telia cells in Lund in Sweden

- **CID** - Cell-Id  
  Example: 3ab6 for this specific cell

**Cell-id in JavaME**

A common way is to use a JavaME applications to read out the cell-id information. The cell-id information is stored in system properties which can be read if you know the name of the properties. The name of these properties varies from platform to platform. Some manufacturers also requires that the Java application is signed by a trusted authority otherwise the properties calls will return zero.
Overview

This code snippet demonstrates how to get the mobile cell ID by using Nokia-proprietary system properties in Series 40 and S60 devices. Note that a different system property is used to get the cell ID in Series 40 and S60 devices:

Series 40 3rd Edition, FP1 (or newer):

```
System.getProperty("Cell-ID")
```

S60 3rd Edition, FP2 (or newer):

```
System.getProperty("com.nokia.mid.cellid")
```

Note: In Series 40 devices the MIDlet needs to be signed to a manufacturer or operator domain; otherwise the property will be null. Signing is not needed for S60 devices.

Picture: Read out cell-id in a JavaME application on Series 40 and S6042.

Location that is not GPS based

These properties usually need security access, that is the application needs to be signed by a trusted authority (3rd party, operator or manufacture).

I know LG needed that in some most of their devices or they'll return null value.

For LG:

- `com.lge.net.isonhomeplmn` - is on home plmn - checks for roaming data, returns true if on home network, false if no reception or not (so extra parameter needs to be checked before deciding you are roaming).
- `com.lge.net.cellid` - cell id parameter
- `com.lge.net.cmcc` - current mobile country code
- `com.lge.net.cmnc` - current mobile network code
- `com.lge.net.mcc` - mobile country code
- `com.lge.net.mnc` - mobile network code
- `com.lge.net.hmcc` - home mobile country code
- `com.lge.net.hmnc` - home mobile network code
- `com.lge.net.lac` - location area code

Picture: Cell-id JavaME properties for LG43.


---

Development of Location Based Services
Cell-id database lookup

Once you have read out the cell-id information from the handset, you need to translate these parameters to geographical coordinates before you can do something useful. As mentioned before, there are several initiative to build-up public databases for cell-id look-ups.

Picture: Cell-id look-up in the cell-id database provided by Location-API.com.