Location Based Services in mobile Java applications A comparative study of Java Micro Edition and Android

Håvard Sataøen, Torbjørn Strøm
Department of Technology, Buskerud University College Kongsberg, Norway

Abstract

Location Based Services (LBS) are expected to become the next big thing within mobile applications as integrated GPS is becoming a standard feature of new phones. This paper, which is based on a master thesis, gives an introduction to LBS before comparing the support for LBS in the two Java platforms Android and Java Micro Edition. Location awareness, web services, integration of maps and peer-to-peer communication are the main areas which are covered.

To be able to compare the two platforms, an application which lets the user see the current location on a map, search for addresses, get directions and exchange location with other users has been implemented for each platform. Available web services and third party libraries have been used in addition to the platform libraries where this is appropriate and the platform does not provide the required functionality. The applications have been used in experiments to measure the generated internet traffic for different operations and solutions.

1 Introduction

The performance, memory size, bandwidth and screen of mobile phones are getting better and better for each passing year. The mobile technology is driven by fierce competition in a market where many customers replace their phone every 1-2 year. The mobile phones have seen more and more features being built into them. They can now function as digital cameras, mp3 players and recently also come with built in Global Positioning System (GPS). These new features and other mobile phone improvements opens up a whole new world of possibilities for the mobile application developer.

It is not only the hardware that changes in mobile phones. The operating systems and software platforms have also changed in order to take advantage of the new possibilities. There exist quite a few platforms for mobile development. Symbian from Nokia, Java Micro Edition (ME) from Sun, Android from Google and Qtopia from Trolltech to mention some. This paper will focus on and compare the well known Java ME and the brand new Android platform which is still under development. The focus will be on areas related to Location Based Services (LBS) which have been made possible due to location technologies such as GPS.

The paper is based on the master thesis [1]. It gives an introduction to what LBS is all about and to some of the issues that such services introduces. A brief introduction to the two mobile platforms, Java ME and Android, is followed by a comparison of the two. The comparison is based on practical use, experiments and available platform documentation.

2 Location Based Services

Location based services are services that are based on the physical location of the user. A LBS is typically used to answer questions such as "Where Am I?", "What is nearby?" and "How can I get from A to B?". In order to be able to use LBS, there are some basic components that must be available [2]:

- A mobile device must be available to the user in order to request the service. The device can be all from mobile phones to laptops.
- A mobile network must be present in order to send service requests to the provider and in order to receive the response.

This paper was presented at the NIK 2008 conference. For more information see http://www.nik.no/

- The location of the device must be determined by using some of the available location technologies (e.g. GPS).
- A service provider offers services to clients (e.g. address searches).
- Content providers are often used by the service providers in order to acquire maps and other data that might be collected (e.g. weather or traffic data).

Application categories

LBS applications are typically placed into two categories [2], push and pull services. According to [3], one can also add an additional category called tracking services.

Pull services

A pull service is requested by the user. By making a request, the user also gives the service provider permission to use his/her location in order to perform the service. A weather forecast service is an example of such a service. The user can send a request for the weather forecast the next hours for his/her location. [2] further divides pull services into functional services (e.g. order taxi) and information services (e.g. search for hotel).

Push services

A push service is initiated by the service provider as a result of an event (e.g. a given location or a specific time). The user must in this case give the service provider permission to send information to the phone by registering for the service. An advertising service is an example of such a service. The user can register with the service and be notified of bargains when passing stores.

Tracking services

Tracking services offers the location of the mobile phone to those who request it. The owner of the phone must give permission by registering the people who should be allowed to acquire the location. A friend finder is an example of such a service. The friends can all register with the service and thus be able to find each other's locations when needed.

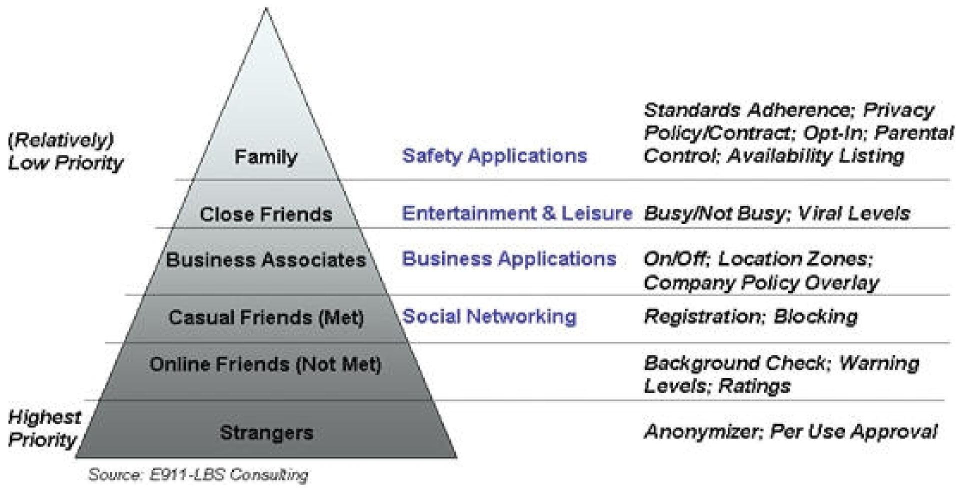
LBS issues

New technologies introduce both new possibilities and new risks. It is important that the developers of LBS applications are aware of the risks that the application might expose the user to.

Privacy and security

In order for LBS to become a success, it is very important that the privacy concerns of the users are taken seriously [4]. Social networking services for instance, poses a larger threat to the user when in a LBS setting. Sexual predators might pinpoint the exact location of their victims if these services are not taking privacy seriously enough. Companies that use LBS to track their employees must also be sensitive to the need for privacy outside working hours. Failure to meet the need for privacy and security will probably destroy the market for any LBS application as they rely on trust.

It is difficult to balance the amount of security built into an application with the usability of the application. The article states that the amount of security needed differ according to who have access to the user's location, see Figure 1. Applications that are used only within the family require less security than applications where strangers might get access to the location.


(Relatively) Low Priority
Highest Priority
Family
Close Friends
Business Associates
Casual Friends (Met)
Online Friends (Not Met)
Strangers
Safety Applications
Entertainment & Leisure
Business Applications
Social Networking
Standards Adherence; Privacy Policy/Contract; Opt-In; Parental Control; Availability Listing
Busy/Not Busy; Viral Levels
On/Off; Location Zones; Company Policy Overlay
Registration; Blocking
Background Check; Warning Levels; Ratings
Anonymizer; Per Use Approval
Source: E911-LBS Consulting
Figure 1: Privacy requirements pyramid [4]

Geoslavery

[5] discusses some of the problems related to tracking services which can be used to monitor people. According to the authors, the combination of Geographical Information Systems (GIS) and LBS can lead to what they call geoslavery. They define geoslavery as “A practice in which one entity, the master, coercively or surreptitiously monitors and exerts control over the physical location of another individual, the slave”. The article points out that although the benefits of LBS can be great in e.g. search and rescue operations, the same technology can be misused in the hands of the wrong people. Misuse can range from the more innocent, like overprotecting parents, to the more severe such as stalking and abusive husbands. Honour killings and modern slavery is also mentioned as rather extreme but possible scenarios in countries where these things occur today. The possible benefits of LBS are great but so are the risks of misuse.

Liability and laws

Due to possible misuse of LBS in e.g. surveillance, it is likely that one will see cases involving LBS technology in court in the years to come. It is therefore important that the laws of a country are updated to reflect the threats posed by this new technology. This is necessary in order to protect the privacy of individuals and prevent abuse from happening. It is not unlikely that a service provider that lacks the necessary security might be held liable in cases involving e.g. sexual abuse of children. In Norway, the location of a person is regarded as personal data and thereby covered by the Personal Data Act [6].

The market

According to [7], Tele Atlas has conducted a study which shows that 84 % of the asked U.S. consumers, identified as relatively advanced users, regarded built-in GPS as a valuable feature for their mobile phones. The participants ranked built-in GPS as a more important feature than e.g. internet access, MP3 player and video. It was only second to a digital camera. This might come as a surprise to some, but it shows that GPS is likely to become the next standard feature in mobile phones. Other interesting findings in this study were:

- 73 % wanted navigation and routing information.
- 68 % wanted to see their location in a detailed map.
- 72 % would search for point of interests and other information in the vicinity.

- 84 % were interested in a child locator. 74 % would accept to pay for the service.
- 47 % were interested in a friend locator. 40 % would accept to pay for the service.

According to [8], a report by ABI Research forecasts the annual global LBS revenue to become 13.3 billion dollars within 2013. This is up from 515 million dollars in 2007. The highest revenue is expected to come from personal navigation ($4.3 billion) and enterprise applications ($6.5 billion). Enterprise applications includes services such as fleet management and tracking of the work force. The article claims that a wider availability of all-inclusive data tariffs will increase the use of such services. The reason for this is that the user no longer has to worry about the amount of data traffic generated by the service. The forecast clearly shows that LBS is about to become big business in the years to come.

3 Locating the mobile phone

An accurate location of the mobile phone is essential in most LBS applications. A lot of effort and research has therefore gone into improving the accuracy and availability of location data. The different approaches and technologies have their strengths and weaknesses. Some have an advantage in urban areas while others performs better in open landscape. Certain types of location based services also need to know on which floor a person or object is located, not only their latitude and longitude. Different types of applications have different requirements when it comes to accuracy, availability, time it takes get a position fix and the cost of the receiver.

Enhanced 911 - distress calls

In 1996 the Federal Communications Commission created a set of rules in order to ensure that the receivers of 911 calls would be able to get the location of wireless 911 callers [9]. In the first phase (compliance within 1998), the wireless carriers had to provide a callback number to the device making the call and the location of the base station that received the call. In the second phase (compliance within 2003), the wireless carriers had to provide even more accurate location information about the caller. The requirements were different for network based and handset based solutions, see Table 1. Carriers which did not comply within the deadline faced fines if not given an extension.

Category	**67 % of calls**	**95 % of calls**
Handset based	50 meters	150 meters
Network based	100 meters	300 meters

Table 1: E911 requirements [10]

Categories

The commission categorized the available location technologies into terminal based, network based and hybrids of the two. The network based solutions require modifications to the base stations but work with any phone on the market. The terminal based solutions require changes to the phone circuitry, but are usually more accurate than the network solutions. Hybrid solutions require changes to both, but are the most reliable and accurate ones. It was up to the carriers to choose their solution. Within each category, there are several available technologies. Some of which can be seen in Table 2. These will not be described any further in this paper.

Category	Technology	Accuracy
Handset based	Global Positioning System (GPS)	10-30 m
Network based	Angle Of Arrival (AOA)	100-200 m
	Time Difference Of Arrival (TDOA)	100-200m
	Wireless location signatures	?
	Enhanced Cell Identity (E-CID)	100m-3 km
Hybrid	Assisted GPS (A-GPS)	5-10 m
	Advanced Forward Link Trilateration (A-FLT)	50-200 m
	Enhanced Observed Time Difference (E-OTD)	50-200 m

Table 2: Location technologies

4 The mobile platforms

Java Micro Edition

Java is probably best known as a programming language for desktop and server applications. According to [11], the first version of Java was developed in the early 1990s under the name Oak. Oak was made for consumer electronic devices and Java ME can therefore be seen as a return to the original intention of the Java platform. Java ME is a subset of the Java Standard Edition (SE) and takes into consideration the limited resources available to embedded devices such as mobile phones. In order to adapt Java SE to embedded devices, some of the interfaces have been omitted, overridden or even replaced by Java ME specific interfaces.

The Mobile Service Architecture (MSA) specification (JSR 248) [12] is the last Java ME expansion, and the one used during the development of the case application. MSA is an umbrella specification which adds increased functionality to the Java mobile platform in order to meet the market demand for new technologies and services. A MSA compatible device must either implement a given subset of the MSA or the full MSA specification. The MSA specification contains several interesting APIs for the LBS developer. The Location API can be used to retrieve the location and orientation of the phone, the Bluetooth API can be used to connect to an external GPS by those devices that do not support the Location API and the Web Services API can be used to communicate with servers offering LBS functionality. To send messages to other devices, the application can use either the Messaging API or the Session Initiation Protocol (SIP) API.

Android

Google acquired the 22-months old software company Android Inc. in July 2005 [13]. Little was known about what Android Inc. was doing other than making software for mobile phones. Some sources indicated that the company had been working on an operating system for mobile phones though. Shortly before the takeover, one of the founders told Business Week that there was a great potential in developing smarter mobile phones that were aware of the owner's location. This acquisition and others indicated that Google had a greater interest in the mobile market than just mobile search which led to some speculations. A lot of rumours circulated prior to the announcement that came on 5 November 2007. Some of them said that Google would announce the release of a new mobile device. This was not the case. Google announced the founding of the Open Handset Alliance and the open source mobile platform Android.

The Android platform [14] is a software stack consisting of an operating system, middleware and key applications. It is based on the Java programming language and applications run on a custom virtual machine called Dalvik which runs on top of a Linux kernel. Android is currently released in an early-look version and the first phones that support it is said to be delivered in the second half of 2008. Android is, not surprisingly, tightly integrated with Google map. As for Java ME, there exists a Location API which is responsible for acquiring the location of the phone. In addition, Android contains libraries for map control, map overlays, driving directions and address searches.

5 Comparing Java Micro Edition and Android

To support the comparison of the two platforms, an application containing the elements to be compared was developed for each platform. Due to the complexity and variety of solutions available, every possible solution has not been implemented. When this is the case, available literature, documentation and examples have been used to support the conclusions drawn.

There were some doubts regarding how the unfinished state of the Android platform would affect the implementation of the application and the following comparison. Android proved to be more complete than expected and thus the implementation of the application was little affected. Nevertheless, the unfinished state of some of the APIs and the test environment have not been to Android's advantage in the comparison.

The case application

The main purpose of the application was to demonstrate different aspects and to be used in experiments. A common analysis formed the basis for both platform versions, but individual detail designs and implementations differed quite a bit due to available libraries and differences in the two platforms. The functionality of the application that was developed is briefly described in this chapter through a use case, see Figure 2.

Exchange locations

To exchange locations with another person and indicate the location on a map.

Manage landmarks

To manage the landmarks by adding new ones, deleting existing ones and viewing the details of individual landmarks.

Display current address

To display the address closest to the current location to the user.

Subscribe to proximity alert

To alert the user when he arrives at a specified landmark.

Display compass

To display a compass to the user that indicates where north is and in which direction the user has to travel in order to reach the selected destination.

Display and control map

To display a map to the user with the current location and let the user pan and zoom.

Display driving directions

To display the route between the current location and a given landmark.

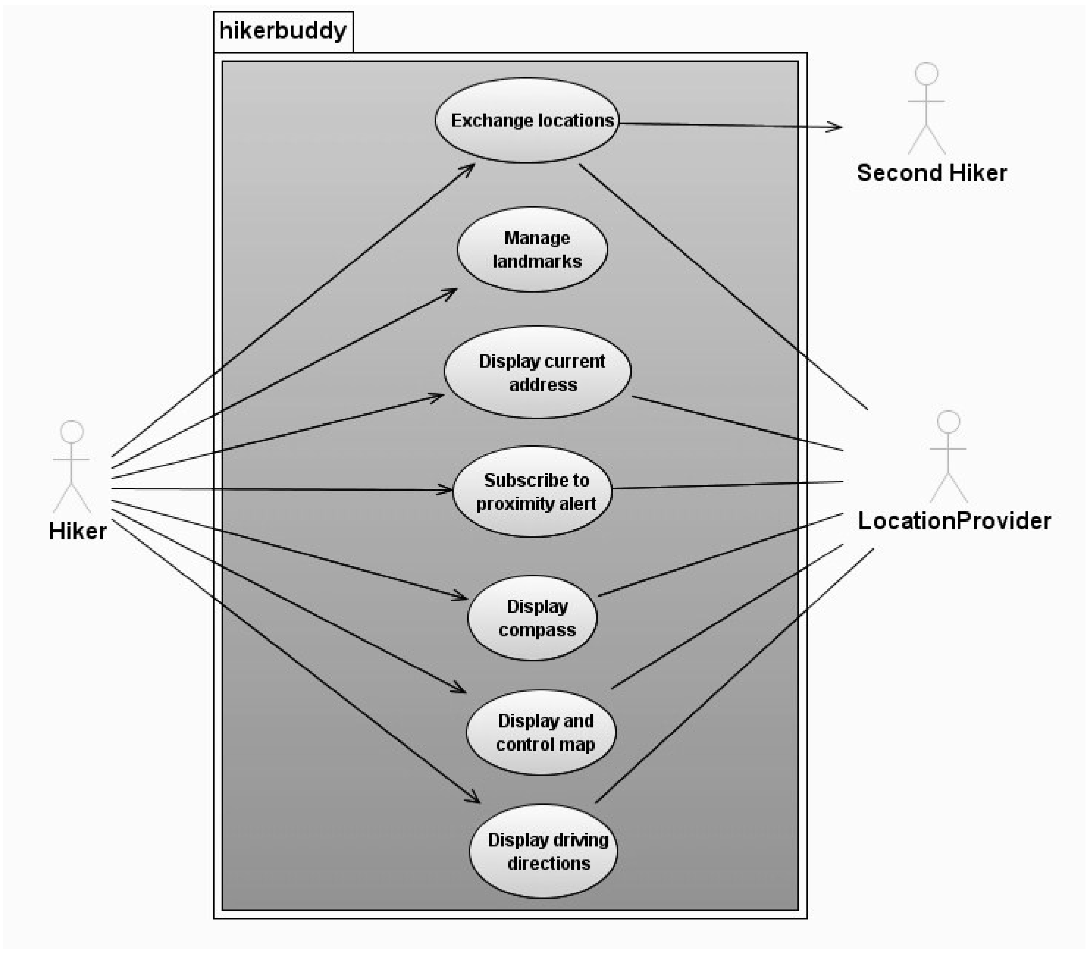


Figure 2: Use case

Location awareness.

The cornerstone of any LBS application is the current location. Both platforms comes with a Location API that hides the actual retrieval of the location from the application. The APIs are generic in the sense that they can be used with any of the available location technologies. The APIs are well designed and provide the developer with more or less the same functionality. In case the Location API is not supported by the device, the platforms contain a Bluetooth API which can be used to connect to an external GPS. Bluetooth was not implemented for the case applications as most new phones will come with built in GPS. Both platforms are well suited for retrieving a phone's location.

Geographical computations

Once a location has been retrieved, the application might need to perform some operations on the location. Both platforms have functions for calculating bearing and distance between two points and for formatting the textual presentation of a location. Only Android has functions for converting a location to screen pixel coordinates for drawing objects in a map. When it comes to the standard mathematical libraries, the Android libraries are similar to those of Java SE while Java ME only comes with a subset. The Java ME libraries are usually sufficient for LBS applications, but Android contains a more complete set of functions.

Phone sensors

Some LBS applications can make use of a phone's sensors in order to improve the user experience. A map can be automatically rotated according to a compass and an accelerometer can be used as user input. A Sensors API that can be used to retrieve data from any sensor is available for both platforms. Of these two, the Java ME version is the better one as it comes with a couple of useful listener interfaces. Java ME also comes with a simple interface for acquiring the phone's orientation as part of the Location API. Both platforms let the developer easily make use of a phone's sensor, but Java ME has a richer interface than Android.

Test environment

The use of location in an application can easily be tested on both platforms using mock location providers. The Sun Wireless toolkit, used for testing the Java ME application, lets the developer specify a scenario in a file or manually enter location data such as location, course and speed. Android supports several methods, and e.g. lets the user use real NMEA sentences or a Keyhole Markup Language (KML) file. A KML file was used during the testing of the Android application. Android provides more testing options, but both platforms have a good environment for testing location awareness.

Web services

Web services are often used in LBS applications in order to take advantage of available third party services and due to the limited resources of mobile phones. The master thesis focuses on how two of the best known web service technologies, Representational State Transfer (REST) and Simple Object Access Protocol (SOAP), can be used on the two platforms.

XML

XML, which is often used in web service messages, are well supported by both platforms. The main difference is that Java ME only comes with a push parser, while Android includes all the main categories (pull, push and model). There are also several available third party parsers in addition to those included on the platforms.

REST

Both platforms includes an API that makes it easy to set up a HTTP connection towards a REST service. REST based web services can therefore easily be consumed by applications on both platforms.

SOAP

At the time of writing, only Java ME comes with a platform API (JAX-RPC) for accessing SOAP web services. KSOAP is an alternative third party library that can be used from both platforms, but it needs some modifications in order to work with Android. It is also possible to manually build SOAP messages using the XML and HTTP support of the platforms. Java ME is the platform that best supports SOAP messages at the moment, but this might change in the future as Android is finished.

Maps

Maps play a central role in many LBS applications as they give a clear picture of the location of the user and the surroundings.

Android comes with Google map integrated in the platform which makes it very easy to implement maps in an application. Java ME has no API for map integration and must therefore rely on third party libraries or web services. The Java ME case application implements two map solutions, one based on the J2MEMap API [15] and another on the REST based web service Yahoo Map Image. These solutions can unfortunately not be used commercially. The J2MEMap API is the one which is compared with Android as it is an API and not just a service returning a map image.

The main impression after using these APIs is that the map API of Android is better designed and easier to use than J2MEMap. This does not come as a surprise since the J2MEMap is a non commercial API developed by a single person. Android on the other hand, is developed by Google which has a lot of experience when it comes to maps.

Map functionality

Android and J2MEMap comes with the same basic map styles, street and satellite. In addition, Android can show traffic information in the map. Both map APIs contain standard map functionality such as pan, zoom and toggling of map style. But there are some differences.

J2MEMap offers the possibility to change the provider of the map (e.g. Google or Yahoo) and has a track object which can be used to e.g. record the history of the user movements. It is also possible to set the tile size used by the map. This function should be used with care, as changing the tile size to 16 or 32 pixels made the application fail after a certain amount of user activity.

Android lets the application nicely animate the map movement to another location or slave the centre of the map to the current location. For drawing information in layers above the map, Android comes with a well designed Overlay class that is easy to use. The overlay in J2MEMap is based on an URL to the overlay being specified by the application. This is not very useful from a programmatic point of view.

Generated data traffic during map operations

The experiments that have been performed shows that the Android API downloads less data than J2MEMap during map operations. This is mainly due to the difference in tile size (16x16 for Android and default 128x128 for J2MEMap) and overhead introduced by the interface when downloading tiles. Larger tiles means that a larger area than is visible to the user is downloaded. It is worth noticing that the expected positive effect of using smaller tiles in J2MEMap does not occur after a certain point. This means that the overhead of downloading tiles in J2MEMap is much larger than for Android.

The results of two of the experiments can be seen in Figure 3 and Figure 4. The first one is a scenario where the map is zoomed from the innermost level to the outermost level and back. In the second one, the map centre is slaved to the location while moving along a predefined track. A tile size of 16 pixels could not be used for this scenario as it makes the application fail before the predefined route can be completed.

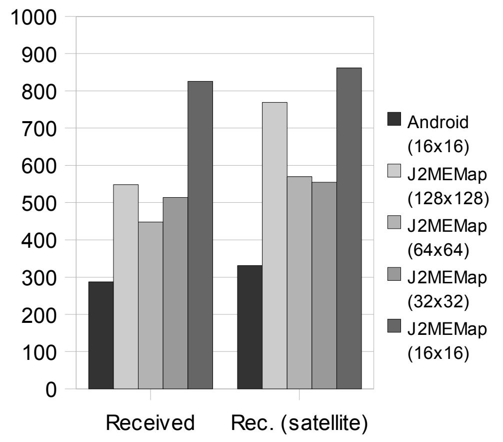


Figure 3: Received data (kB) during map zoom

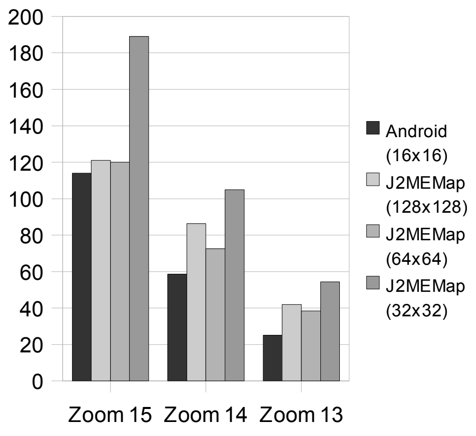


Figure 4: Received data (kB) during slaving

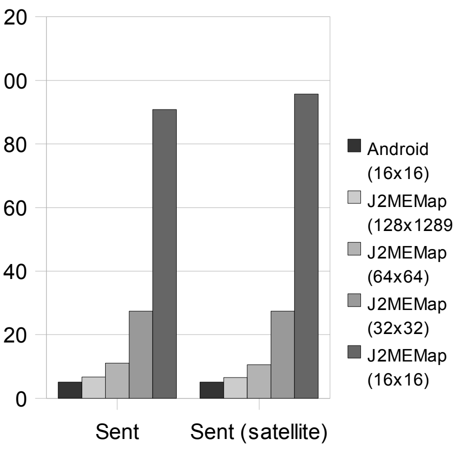


Figure 5: Sent data (kB) during map zoom

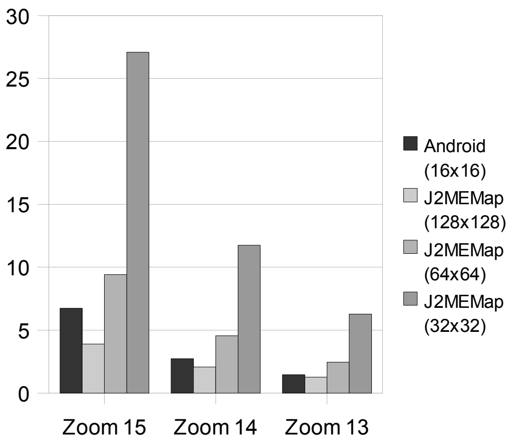


Figure 6: Sent data (kB) during slaving

The amount of data sent during these scenarios can be seen in Figure 5 and Figure 6. There is little difference in the amount of data sent when the default tile size is used for J2MEMap. As one reduces the tile size, the amount of data sent increases rapidly. This is again caused by a less efficient interface for J2MEMap than for Android. The slightly better result, that is seen for J2MEMap (128x128) during the slaving scenario, is a result of Android having to request new tiles more often due to the smaller tile size.

When looking at both the sent and received amount of data, it is clear that Android generates less data traffic and thus less costs for the user. It is also clear that there is little to gain by altering the tile size used by J2MEMap. A tile size of 64 pixels might be slightly better than the default one, but other than that, smaller is worse.

LBS functionality

In addition to the acquisition of the user location and the integration of maps, the thesis also looked at the support for some common LBS functionality.

Addresses

Addresses play an important role in many LBS applications. Both platforms have classes representing an address, but only Java ME comes with a class made for address storage. In order to store addresses in Android, one has to use a regular database.

Different types of address searches are common in LBS applications. Geocoding and reverse geocoding searches for the location of an address and vice versa. A search for Points Of Interest (POI) returns e.g. hotels in the vicinity. Android has a Geocoder API that can be used to perform these services. In order to implement the same services in Java ME, the application must use some of the available web services. This makes address searches much easier to implement in Android than in Java ME.

Proximity alerts

Both platforms lets the application register itself as a listener for notification when the user enters the proximity of a specific location. There are some small differences in the proximity alert functionality, but all in all the two platforms supports this feature equally well.

Driving directions

How to get from one point to another is one of the most popular LBS features. The central element here is the calculation of the route. Only Android has an API for acquiring routes. Java ME must rely on available web services. This makes the task of implementing routing much easier in Android than in Java ME.

Peer-To-Peer communication

Some LBS applications need to exchange locations with other devices. This can be done using SMS messages or by sending messages over an internet connection.

SMS

Both platforms contains APIs for sending and receiving SMS messages. Although the APIs are quite different, the functionality and ease of use are the same. The applications can gain access to the contacts of the phone on both platforms, but only Java ME has a third party GUI component made for this task. An application can also be activated in response to a message on both platforms. All in all, the handling of SMS messages is well covered both in Android and Java ME.

Internet connection

Both platforms provides APIs for exchanging messages over an internet connection. Java ME comes with a Session Initiation Protocol (SIP) API while Android comes with a GTalk API which is currently based on the eXtensible Messaging and Presence Protocol (XMPP). It is worth mentioning that XMPP is not fully supported and that there are some uncertainties regarding the choice of protocols and APIs in future Android releases. The two APIs seem fairly easy to use, but have not been used in the case applications due to the available time.

Testing P2P

The test environment for SMS messages provided by Java ME is much better than that of Android. Java ME lets the developer send messages between emulators and to/from a testing tool. The developer can also look at the content of any exchanged message. In Android, one can only test the reception of a SMS message by sending it from the testing tool and to the emulator. This means that the Java ME test environment is clearly the superior one when it comes to testing SMS. It is likely that the test environment of Android will improve in future releases.

6 Conclusion

The main purpose of the master project was to investigate and compare how Location Based Services (LBS) can be implemented using the two mobile Java platforms Android and Java ME. An application has been made for the two platforms in order to explore and demonstrate some common LBS functionality. The application was implemented successfully on both platforms.

Maps, different address searches and driving routes are all covered by platform APIs in Android, while Java ME must rely on third party APIs and web services. This affects both the application size and the time it takes to develop the LBS application. When it comes to handling locations, web services and sending of messages, there is no substantial difference between the two platforms. The main difference lies in the many LBS related APIs that can be found in Android but not in Java ME.

The overall conclusion is that the Android platform appears to be the one best suited for implementing LBS applications as the developer will rarely have to rely on third party APIs and web services.

7 References

[1] H. Sataøen, *Location Based Services in mobile Java applications. A comparative study of Java Micro Edition and Android*, Master thesis, Buskerud University College Kongsberg, 2008.

[2] S. Steiniger, M. Neun and A. Edwardes, *Foundations of Location Based Services*, retrieved 28 Mar 2008, <http://www.geo.unizh.ch/publications/cartouche/lbs_lecturenotes_steinigeretal2006.pdf>

[3] GSM MoU Association, *Location Based Services*, 2003, retrieved 28 March 2008, <http://www.gsmworld.com/documents/lbs/se23.pdf>

[4] D.H. Williams, *LBS Development - Determining Privacy Requirements*, Directions Magazine, 2006, retrieved 28 March 2008, <http://www.directionsmag.com/article.php?article_id=2323&trv=1>

[5] J.E. Dobson and P.F. Fisher, *Geoslavery*, IEEE Technology and Society Magazine, 2003, pp. 47-52, retrieved 29 March 2008, <http://dusk2.geo.orst.edu/virtual/2005/geoslavery.pdf>

[6] Datatilsynet, *Act of 14 April 2000 No. 31 relating to the processing of personal data (Personal Data Act)*, 2000, retrieved 25 May 2008, <http://www.datatilsynet.no/upload/Dokumenter/regelverk/lov_forskrift/lov-20000414-031-eng.pdf>

[7] Tele Atlas, *Tele Atlas Expands Commitment to Wireless Industry*, 2007, retrieved 4 May 2008, <http://www.teleatlas.com/WhyTeleAtlas/Pressroom/PressReleases/TA_CT015697>

[8] *Mobile Location Based Services Revenue to Reach $13.3 Billion World wide by 2013, says ABI Research*, Phone Content, 2008, retrieved 4 May 2008, <http://www.phonecontent.com/bm/news/2038.shtml>

[9] Federal Communications Commision, *FCC amended report to congress on the deployment of E-911 phase II services by tier III service providers*, 2005, retrieved 19 February 2008, <http://hraunfoss.fcc.gov/edocs_public/attachmatch/DOC-257964A1.pdf>

[10] Federal Communications Commision, *FCC wireless 911 requirements*, 2001, retrieved 19 February 2008, <http://www.fcc.gov/pshs/services/911-services/enhanced911/archives/factsheet_requirements_012001.pdf>

[11] E.C. Ortiz, *A Survey of Java ME Today (Update)*, Sun Microsystems, 2007, retrieved 19 February 2008,<http://developers.sun.com/mobility/getstart/articles/survey/>

[12] Sun Microsystems, *Mobile Service Architecture*, 2007, retrieved 22 February 2008, <http://java.sun.com/javame/reference/docs/msa_datasheet.pdf>

[13] B. Elgin, *Google Buys Android for Its Mobile Arsenal*, Business Week, 2005, retrieved 2 March 2008, <http://www.businessweek.com/technology/content/aug2005/ tc20050817_0949_tc024.htm>

[14] Android homepage, <http://code.google.com/android/documentation.html>

[15] J2MEMap homepage, <http://j2memap.8motions.com/>