View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Universiti Teknikal Malaysia Melaka (UTeM) Repository

Building Location-based Service with Java Technologies

Anusuriya Devaraju
Kolej Universiti Teknikal Kebangsaan Malaysia, Malaysia.
anusuriya@kutkm.edu.my

Simon Beddus
British Telecommunications (BT) Research and Venturing, UK.
simon.beddus@bt.com

Abstract – *The growing use of Java in Location-based Service provides an opportunity to find solutions for problems and challenges in the rapidly changing telecommunications environment. This paper describes the development of location-based service components using Java technologies. The technologies include J2ME, Servlet, Java Server Pages (JSP) and XML Java Binding Tool. The developed components are the location server simulator, location service application and device client application. This study is crucial for support of BT's launch of User Location Service on prototype ERICA mobile application platform through supporting the testing and validation of the platform components.*

Keywords: Location Based Service, Simulator, Mobile Location Protocol, XML, J2ME

1 Introduction

Location-based Services or LBS are any services use spatial data that are available to anyone, anywhere, anytime on any mobile-based device. Common services for the consumers include Safety and Emergency, Tracking, Navigation, Information Guide, and so forth. This project deals with the design delivery and testing of location-based service components using Java technologies. This paper is structured into several parts. The first part opens discussion leads to the formulation of the problem that is to be analyzed. The second part contains theory; focusing on basic concepts and technologies to build LBS. The third part presents design and development of each component in the system. In the last part, the future and open work items are clarified and conclusions are presented.

1.1 Background

Before Windows, developers faced many problems to develop applications on computers. However, when Microsoft created basic software standard that took care of the system operation, the developers enable to build new applications with minimal fuss. Project Erica aims to do the same for mobile communications. This prototype mobile application platform (Figure 1) is purposely developed by BT to provide a middleware service access point to the network infrastructure enabling the mobile operators to build and deploy wide range of wireless API services with management capabilities [3]. One of mobile services offered by Erica is the User Location Service or ULS. The ULS is mainly designed to enable a single location Application Programming Interfaces (API) for network based location services. Application developers can easily integrate and deploy new and existing location based applications, utilizing real time positioning data from UK mobile operators such as *Three, O2, Vodafone, T-Mobile and Orange*. This helps to reduce the risk and cost to both parties. However, testing each application with real time data is charged per lookup. As the complex and expensive testing can inhibit cost-effective development of location-based applications, we need a different approach to testing and evaluating these applications without access to a real mobile network. In response to this need, this project introduces a location server simulator based on Location Interoperability Forum (LIF) API [11]. A location-based service application and a J2ME client called 'Map4U' are developed as part of the test and evaluation process for the simulator. Apart from this, there is a portal, which allows developers to remotely manage the location data.

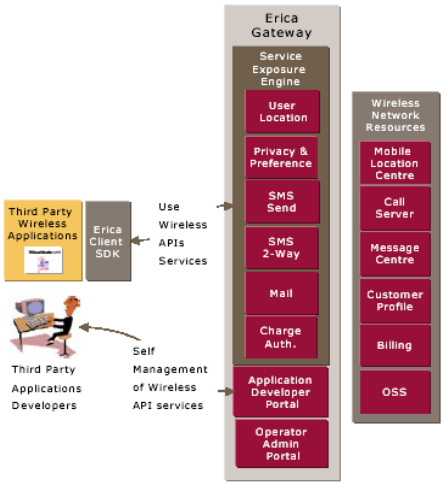

Figure 1. Erica mobile application platform

2 Basic Concepts and Technologies

In order to develop location-based service system, the location APIs, protocols, technologies and infrastructure of LBS must be understood and reconciled.

2.1 Positioning technology

There are two basic of positioning a mobile device, firstly by using satellite for instance, Global Positioning system (GPS); secondly by using mobile telephone network. This section will discuss the most prominent network based positioning, the Global System for Mobile Communications or GSM. GSM is the leading digital cellular radio network. While the current GSM system was originally designed with an emphasis on voice sessions, the General Packet Radio Service (GPRS)

system brings the packet switched bearer services to the existing GSM system. GPRS uses the bandwidth more efficiently because it does not require a dedicated line. The call charging is solely based on amount of transmitted data.

2.2 Positioning method

The common positioning methods found basis for providing location services in all networks is the Cell-based Positioning. This method provides a location coordinate based on the cell the subscriber is within. The location infrastructure contains information on the location coordinate of each cell centric. The current Cell ID can be used to identify the Base Transceiver Station (BTS) that the device is communicating with and the location of that BTS. The accuracy of this method depends on the size of the cell. Positioning is generally more accurate in urban areas with a dense network of smaller cells. Rural areas have a lower density of base stations.

2.3 Location API and protocol

Wireless services are not connected directly to mobile telephone system. They are connected to the Gateway Mobile Location Centre (GMLC) or Mobile Positioning Centre (MPC), which act as bridge between the IT world, and the telecom world to retrieve the location information from GSM/UMTS networks. One of the ways to acquire location data from network today is by using LIF API. LIF is a global industry initiative, which is jointly formed by Ericsson, Nokia and Motorola in September 2000. It has produced an application level protocol called Mobile Location Protocol (MLP). The purpose of the MLP specifications is to define a simple and secure access method that allows Internet applications to query location information from a wireless network, irrespective of its underlying air interface technologies and positioning method [11].

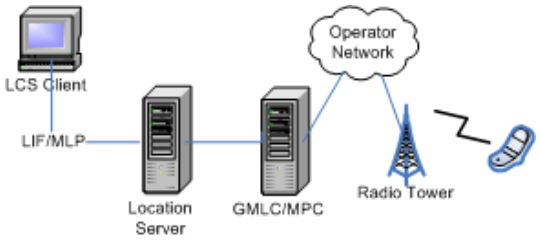

Figure 2. Location service architecture

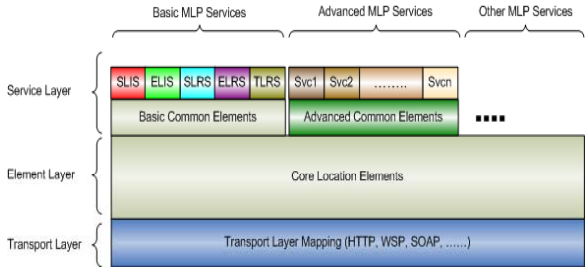

Figure 3. Layered view of Mobile Location Protocol

2.3 Why Java for location-based service?

The following topic describes briefly how Java technologies support growth of the LBS industry in the telecommunications environment.

Platform independence. Wireless applications developed using Java are optimized for portability. For example, an application written using the MIDP APIs from J2ME will be directly portable to any MIDP devices compared to native clients (applications written using proprietary programming languages such as C and C++).

Easy to develop. Java language retains most of the power of C++, but with far less complexity. The breadth and robustness of the core Java Foundation Classes allows developers to get their wireless applications developed more quickly with minimal fuss. In addition, the Java application server provides system infrastructure and management tools for deploying wireless applications.

Security for wireless applications[17]. Java provides several levels of security, from class loader and byte code verifier to Security Manager, which can protect systems from untrustworthy programs. Java also provides extended security APIs for securely transforming content over the wireless network. For instance, MIDP 2.0 include support for HTTPS, also there are also third-party Java-based security packages such as *Bouncy Castle*, *KSSL*, and *Phaos Micro Foundation toolkit* to build secure location-based applications.

The partnership between Java and XML. Java is portable language and XML is portable data. XML's extensible data types fit well Java's platform independent executables and combining these two provide interoperability and scalability between application components and application themselves [9]. Several Java based XML binding tools are freely available, such as *XMLBeans*, *Enhydra Zeus*, *Castor*, and etc.

Modularity. When developing a complete location-based service system, programs can get large and complex. LBS must be modular and reusable independently of the underlying systems so that they are easier to maintain and understand. The J2EE platform contains packages such as Servlets, JSP, Java Messaging Service (JMS), Remote Method Invocation (RMI) and Enterprise JavaBeans (EJB) that provide a way to modularize and divide the application down into different tiers and individual tasks [1].

2.4 Technologies that support system development

Java 2 Micro Edition. The J2ME technology consists of a range of Java virtual machines and a library of APIs that target consumer devices with limited memory, display and processing power, such as cell phones, pagers, PDAs and set-top boxes. Configurations and profiles are the two main building blocks in J2ME. A

configuration defines the Java language and virtual machine features and minimum libraries that expect to be available on all devices of the same class. A *profile* is reside on top of a configuration, adding a set of APIs for user interface components, persistent storage, network connectivity, and whatever else is necessary to write applications for a particular type of device. For instance, the Mobile Information Device Profile (MIDP), when combined with the Connected Limited Device Configuration (CLDC), provides a portable and extensible platform for developing wireless applications for small mobile devices, mainly cellular phones and two-way pagers (Figure 4). The *Sun Java Wireless Toolkit* is a state-of-the-art toolbox for developing wireless applications that are based on J2ME's CLDC and Mobile Information Device Profile. The toolkit includes the emulation environments, performance optimization and tuning features, documentation, and examples that required by developers to built efficient and successful wireless applications.

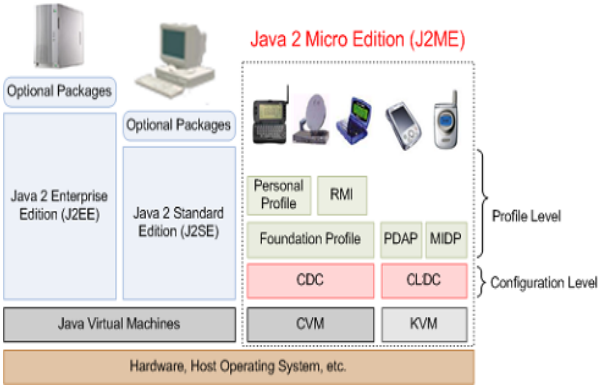

Figure 4. J2ME runtime environment

J2ME versus competing technologies [4]. *Thin client applications* are based on mark-up languages such as Wireless Markup Language (WML), HTML, and Compact HTML (cHTML) for content generation and interactions. These applications are relatively easy to create; however they require a constantly available network connection. In a thin client solution, the applications themselves reside on a server and the server is responsible for generating displays mark-up, this requires a round-trip every time the interface changes. In contrast, a J2ME client is contained within the device, so it can operate even when disconnected. On occasions when the devices do interact with the server, it incurs less network traffic because the J2ME client downloads only the application data. Besides that, the developer also can choose how to split their application and service logic between the device and server. Thin client solution such as WAP also has known security issues in the WAP gateway, where the protocol conversion occurs, while Java technology provides for a robust, well-tested security model, with many J2ME based devices supporting end-to-end secure HTTP (HTTPS).

Native applications are developed for a particular operating system such as PalmOS, Windows CE, SymbianOS are traditionally written in C, C++ or C# languages. These applications can work in an offline environment but are completely dependent on the operating system of the device. They require a higher level of expertise because of the complexity of the user interface, communication models, and the wide variations in OS capabilities. Apart from this, built-in, fixed-feature applications are difficult to upgrade and install new applications without getting the manufacturer involved. Conversely, native applications can take full advantage of inherent featured such as hardware management and interaction because they are tightly coupled to a particular operating system. Compared with the native platforms, the main strength of the J2ME is that it provides an access to dynamic content applications and allows us to write portable wireless applications, on any network and on any device easily.

Java Servlet and Java Server Pages (JSP). Servlets are instances of Java class that operates as a web component. They are invoked to service HTTP requests and generate responses to those requests. Servlets operate within a 'Servlet' container; application servers such as J2EE, Tomcat usually provides this container [8]. When the user issues a request for a specific Servlet, that server will simply use a different thread and then process the individual requests. This has a positive impact on performance, since multiple requests do not generate multiple processes. Threads need fewer resources, so their use already gives them an advantage.

JSP is server side programs to generate dynamic web contents. When the first time the request for a JSP comes in, the JSP engine on application server will compile the page to a Servlet, then the application server's Servlet engine will run the compiled page and then returns the resulting contents to client. If the JSP gets changed, the class loader of the application server detects it and recompiles before serving it again [8].

XMLBeans [16]. XMLBeans is an XML binding tool, which enable user to access and manipulate XML documents by Java programs easily. It is developed by BEA Systems, and was donated to the Apache Foundation in 2003. XMLBeans compiles an XML Schema to generate a set of Java interfaces that mirror the schema. An XML schema describes the structure or a set of rules of an XML document, to which other XML documents must conform. The resulting XMLBeans classes are able to parse any XML instance document that conforms to the schema. Also, an instance documents can be created by manipulating the XMLBeans classes.

3 System Architecture

The architecture of the system is shown in Figure 5. The simulator and location-based service application are built with Servlet and JSP and require a Servlet/JSP enabled web server to run on. In this case, Apache Tomcat is used to support both components. The device client application is based on MIDP 2.0 from J2ME. A user having a J2ME enabled mobile device will be able to download the device client application from a network and run them on his/her mobile device. The simulator does not have connection to obtain real-time positioning

data from network. Thus, a database is created on MySQL to store the simulated location data. The interface highlighted between the Location Service Application and Simulator uses the MLP 3.0.0. Among the five services offered by MLP specification, the simulator supports the *Standard Location Immediate Service (SLIS)*. SLIS is the standard service that allows subscribers to request a single location response from the Location Server. The request may be served by several asynchronous location responses (until a predefined timeout limit is reached). This service used request-response messages known as *Standard Location Immediate Request (SLIR)* and *Standard Location Immediate Answer (SLIA).*

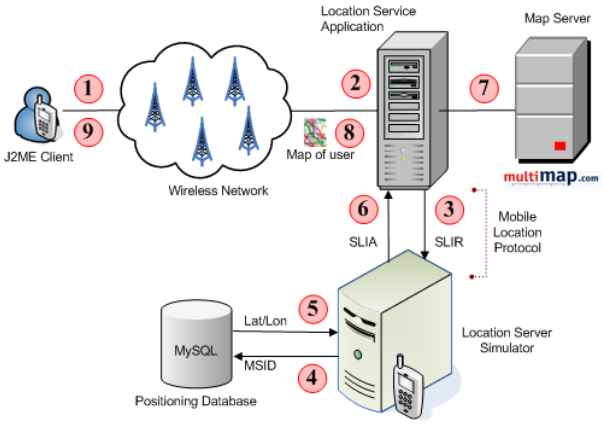

Figure 5. System Architecture

The following steps describe the interaction between components in the system.

1. MIDlet is a J2ME client application installed on mobile phone. The midlet supports map display, selective zooming, standard menus, and progress indicator to let the mobile users know that the application is working. The mobile id, username and password are sent to the location service application via HTTP to request the map of the subscriber.
2. When the location service application receives the request, it identifies the subscriber and the service requested by the subscriber.
3. Then, it constructs a SLIR, and sends it to the simulator.
4. The simulator parses the received SLIR. Parsing SLIR means retrieving data from an XML document based on its meaning and structure.
5. Format or method of request will be validated against DTD during XML parsing. If all checks pass, the simulator connects to the positioning database and retrieves a random location data of the mobile id. The positioning data are expressed in latitude and longitude decimal degrees. However, if there was an error in the request, or the request could not be fulfilled, an error response is returned instead.
6. Finally, the simulator constructs a SLIA, consists of location data and other elements such as radius, local time, GMT and so forth. It classifies the MIME type as 'text/xml' and writes the SLIA contents into opened HTTP connection from the application service provider.
7. After parsing the SLIA response message, the location service application opens an URL connection to the Multimap server with the location data. The map will be read from URL.
8. The location service application customizes the format and size of the map to fit the device's characteristics, and then send it to the mobile user. If an error message is returned from simulator, the service will not initiate a map request; instead it will forward the error message directly to the mobile subscriber.

3.1 Initiating a location request

XML over HTTP. In LIF/MLP specification, the protocols that can to be used for communication between the location application and location server are the Simple Object Access Protocol (SOAP) and XML. This project implements communication based on XML over HTTP. The XML request documents can be posted directly from developer application (using Java, or any other development environment) to the actual MLP 3.0.0 interface. The location service application issues an HTTP POST request towards the Location Server. The request includes the entity-header *Content-length* field as part of the request. The SLIS can be invoked by posting the SLIR to the following URL:
`http://mdp.opentel.bt.co.uk:8080/SLIS/`

Request format. The request consists of two parts, the *header* and the *body*. Header element contains client credentials and other information pertaining to the client. The body contains the mobile identification, which is the XML formatted request. The simulator supports the GSM Mobile Subscriber ISDN (MSISDN) as the Mobile Subscriber Identifier (MSID). A subscriber's MSISDN is simply his or her mobile phone number.

```
<?xml version = "1.0" ?>
 <!DOCTYPE svc_init SYSTEM "MLP_SVC_INIT_300.DTD">
<svc_init ver="3.0.0">
     <hdr ver="3.0.0">
           <client>
           <id>asd</id>
           <pwd>asd38</pwd>
           </client>
     </hdr>
     <slir ver="3.0.0">
           <msids>
           <msid type="MSISDN">447979374734</msid>
           </msids>
           <eqop>
           <resp_req type="LOW_DELAY" />
           </eqop>
           <loc_type type="CURRENT_OR_LAST"/>
     </slir>
</svc_init>
```

Figure 6. A successful SLIR

Manage location data with portal. Request parameters such as username, password and mobile id will be provided when a user registered with the portal (Figure 6). Using these parameters, the service application providers able to test and demonstrate location-based applications using the simulator MLP interface. The data contained in this site is simulated data

and is not linked to any person or business. Using the web-based Graphical User Interface (GUI) of portal with provided username and password, the developers able to add and modify sets of location data and its response code for given mobile id that shall be possible to position.

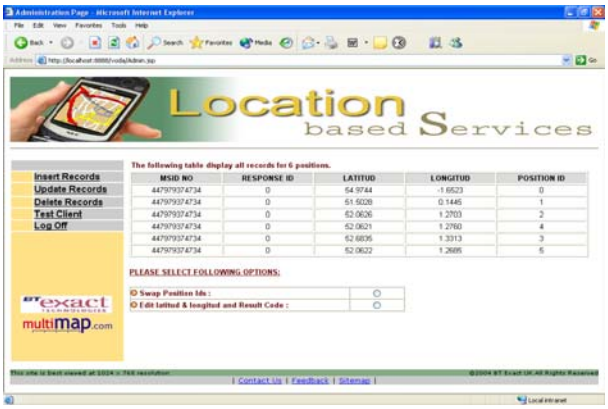

Figure 7. Portal to update location data

3.2 Generating a location response

XML binding. The method that is used to parse the request-response messages is *Simple API for XML* or SAX. SAX is an event-driven interface in which the parser invokes one of several methods supplied by the caller when a 'parsing event' occurs. 'Events' include recognizing an XML tag like start and end of the document, start element, end element, finding an error, encountering a reference to an external entity, or processing a DTD specification. The parser reads the SLIA contents line by line and fires events that contain information about the line that was just read. User defined Java program that listens to a particular events will extract the location request values from the XML document. The XML parser used in this project is *Apache Xerces for Java* [2]. In this project, XMLBeans is used to construct request/response parameters into XML message because SAX (even though faster) can be more complicated for writing XML from scratch. XMLBeans classes that mirrored the schemas of SLIR/SLIA XML documents are manipulated to create the instances of the request/response messages.

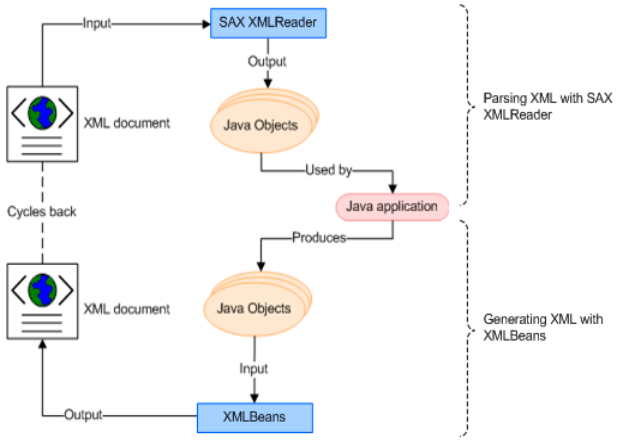

Figure 8. XML data binding application flow

Figure 9 shows a position for the indicated MSID, in the shape of a circle centered at the specified coordinate with the given radius. The position was acquired at the time 12:57:16 on 03-08-2005 in time zone +0700.

```
<?xml version = "1.0"?>
<!DOCTYPE slia SYSTEM "MLP_SLIA_300.DTD">
<slia ver="3.0.0">
      <pos>
      <msid>447979374734</msid>
      <pd>
      <time utc_off='+0700'> 20050308125716 </time>
      <shape>
      <CircularArea>
            <coord>
            <X>52.0622</X>
            <Y>1.2685</Y>
            </coord>
            <radius>20</radius>
      </CircularArea>
      </shape>
      <alt>5280.0</alt>
      <alt_acc>20</alt_acc>
      </pd>
      </pos>
</slia>
```

Figure 9. Successful SLIA

Error handling. The simulator performs error handling concerning following aspects to allow localizing.

a. *Unsupported request*
Client request a service other than SLIS, which is not supported by the simulator.

b. *Unknown and incorrect request*
A protocol element in the request has invalid format. The XML parser cannot resolve a XML request because of syntactical errors in the incoming request like missing brackets, end tags.

c. *Localization*
The request cannot be handled because connection to the positioning database is failed. The location data is not available for requested mobile id.

d. *Authentication and Authorization*
The requesting location-based application is not allowed to access the location server or a wrong password has been supplied. The application provider is unknown or no such subscription exists.

Further information regarding the error will be indicated in *add_info* element on SLIA message (Figure 10).

```
<?xml version = "1.0"?>
<!DOCTYPE slia SYSTEM "MLP_SLIA_300.DTD">
<slia ver="3.0.0" >
<result resid="6">POSITION METHOD FAILURE </result>
<add_info>Position method failure. The location
service failed to obtain the user's position.
</add_info>
</slia>
```

Figure 10. Failed SLIR (Position Method Failure)

Location response time. The simulator cannot provide a guaranteed response time for sending location information back to the service application. The simulator and positioning database reside on different server; response times vary and could be delayed by network conditions. For this reason, the simulator maintains a

cache of old subscriber locations. This cache can be made use of by specifying the *'resp_req'* element in Quality of Position (QoP) parameter in the MLP request. This element allows for slightly older information to be returned in exchange for a faster response [15]. Table 1 defines how long the application is willing to wait for a response before a response is returned.

Table 1. Controlling response time with '`resp_req`'

Value	Action
`NO_DELAY`	Requesting a location with no delay. If the cache location is not available, the simulator responds with error.
`LOW_DELAY`	Requesting a location with low delay (30 seconds). If the current location is not obtainable, the simulator checks for cache value. If the cache location value is not available, the simulator responds with error.
`DELAY_TOL`	Requesting a location with maximum delay (60 seconds). The behavior of this option value is identical to LOW_DELAY except for the difference in time allowance.

3.2 Generating map to fit device's characteristics

The location service application converts and resizes the image instead of the device client application because it makes the client smaller and likely to be much faster. Moreover, complex business logics and heavy computation are left to the server-side due to limited resources available on wireless.

Format Conversion. The map server returns the requested map in *Graphics Interchange Format* (GIF) format. Here, image format conversion is necessary, as the MIDP specification only requires support for a single format, the *Portable Network Graphics* (PNG) format. PNG images are always compressed with a loss-less algorithm. The algorithm is identical to the algorithm used for JAR files, so the MIDP implementation can save space by using the same algorithm for both purposes [10]. Images already in PNG format can benefit from being funneled through a service, because the service can adapt the image to fit the device's characteristics.

Map Resizing. In J2ME, there is no specific method to get the model number from the device at server. Thus, the mobile model name is set manually in the midlet as 'user-agent' to send through the request header. When the location service application receives the map, it uses the 'user-agent' parameter to obtain the screen size of a mobile phone from the *'mobilephone'* database. The database contains current list of MIDP devices with their screen size. More comprehensive list of MIDP devices can be found at Sun's J2ME MIDP website [7].

3.3 Deploying J2ME client application

The device client application and associated components are packaged into a JAD/JAR file pair using J2ME Wireless Toolkit. There are 2 possibilities to install midlet on a mobile device; either connects the device to the desktop with a data cable, or download midlet from the Internet to a device over the wireless network. The process of deploying midlet suites over a network is known as *Over-The-Air (OTA)* provisioning. OTA provisioning works as follows. First a mobile phone sends a WAP request for a JAD file to a Web server through a WAP gateway. The Web server sends JAD package to the mobile phone. Then the phone fetches the JAR package defined in the JAD file from the Web server. The phone's Java Application Manager (JAM) installs the package.

The following diagrams show how the mobile user uses Map4U midlet to send a request and receive the map from to location service application. The components have been tested via O2 GPRS service.

Figure 11. Screenshots of device client application

4 Future Works and Conclusions

Besides SLIS, there are different possible types of location services from LIF framework that can be implemented in the simulator, for example Emergency Location Immediate Service, Emergency Location Reporting Service and Triggered Location Reporting Service. The simulator should support more error handling concerning internal errors and other possible errors such as too many position items have been specified in the request, a protocol element or attribute in the request has an invalid value, account suspended, this is normally when an application tries to connect more than three times with a wrong password and so forth. To control the age of location and response time, the *resp_req* parameter should be mapped with other parameters *loc_type*, *max_loc_age*, *resp_timer* into timer values for the location response. Apart from this, the simulator should allow location lookups for multiple subscribers in one single request. One way to ensure secure communication between simulator and location service application is by implementing the Java Secure Socket Extension (JSSE) packages. JSSE implements a Java technology version of Secure Sockets Layer (SSL)

and Transport Layer Security (TLS) protocols. It includes functionality for data encryption, server authentication, message integrity, and optional client authentication.

4.1 Conclusions

The main contribution of this project is a location simulator based on LIF/MLP. The simulator permits third parties to validate their application with location data in a test environment prior connecting to live and charged services from the mobile network. The simulator doesn't support all the services defined in MLP specification, but it supports basic service for developing and test most location-based application. The implementation of MLP has been applied to a sample location-based application and J2ME client to show its structure and use.

This project has focused on mobile location service solution developed using Java. A wide range of Java technology has been introduced to build components in the system. The deployment of Java in building location-based applications provides benefits, which include ease of use, cross-platform architecture, language simplification, and access to the Internet's established Java API development.

References

[1] Allamaraju, et al, *"Professional Java Server Programming J2EE"*, Wrox Press, 2001.

[2] Apache Xerces Parser ; http://xml.apache.org/xerces2-j/

[3] BT Exact Technologies, "Erica Wireless API service Gateway", British Telecommunications plc, UK, 2002.

[4] C.Enrique Ortiz , "Elements of a Typical J2ME MIDP Business Application", Micordevnet Technical Article, 2001. http://www.microjava.com/articles/techtalk/midp

[5] Hjelm J, *"Creating Location Services For The Wireless Web"*, John Wiley & Sons, Inc, 2002. http://developers.sun.com/techtopics/mobility/midp/articles/https/

[6] Jagoe A, *"Mobile Location Services- The Definitive Guide"*, Prentice Hall PTR, 2002.

[7] Java 2 Platform, Micro Edition (J2ME); http://java.sun.com/j2me/index.jsp

[8] Jepsen T, et al, *"Java in Telecommunications: Solutions for Next Generation Networks"*, John Wiley & Sons, 2001.

[9] Knudsen J, *"Wireless Java: Developing with J2ME"*, 2nd Edition, Apress, 2003.

[10] Kroll M and Haustein S, *"Java 2 Micro Edition (J2ME) Application Development"*, Pearson Education, 2002.

[11] Location Interoperability Forum, "Mobile Location Protocol Specification", 2002. http://www.openmobilealliance.org/tech/affiliates/lif/lifindex.html

[12] Mahmoud Q, *"Learning Wireless Java"*, 2nd Edition, O'Reilly, 2002.

[13] Mahmoud Q, "Secure Java MIDP Programming Using HTTPS with MIDP", Sun developer's Technical Article. http://developers.sun.com/techtopics/mobility/midp/articles/https/

[14] Muchow J, *"Core J2ME Technology"*, Prentice Hall, 2001.

[15] Openwave, "Location Studio 2.1 MLP 3.0.0 Developer's Guide",Openwave Systems Inc, 2004.

[16] XMLBeans; http://xmlbeans.apache.org

[17] Yu Feng, Jun Zhu, *"Wireless Java Programming with J2ME"*, Sams. 2001.