Abstract- The increase of data traffic and access to the Internet has resulted in more consumer-driven markets. There is increasing pressure for Mobile Network Operators (MNOs) to quickly and innovatively provide personalised multimedia services that combine voice, video, presence and location. Many traditional SIP applications have always included presence capabilities in their implementations. Location, on the other hand has been less popular, possibly due to the lack of relevant software and hardware as well as the cost-effectiveness of delivering such services. Recent trends have indicated that Location Based Services (LBS) adoption and development are on the increase. The main focus of this paper is to showcase the functionality of a SIP Location Based Services (LBS) toolkit and location-aware Mobicents Presence Service we put together. This is done through the development of a basic friend-finder service. More cross-referencing LBS applications can be developed quickly using our toolkit.

Index Terms—LBS, J2ME, Mobicents, Geolocation-policy

I. INTRODUCTION

The increase of data traffic and access to the Internet has resulted in consumer-driven markets where Telecoms and Mobile Network Operators (MNOs) no longer have the benefits of the comfortable and monopolistic delivery of voice-centred services [1]. There is increasing pressure for MNOs to quickly and innovatively provide personalised consumer services and to open up their networks to third-party service developers for multimedia services that may combine voice, video, presence and location.

Many traditional SIP applications have always included presence capabilities in their implementations. Location, on the other hand, has not been as popular. This was most likely due to the lack of efficient hardware (such as GPS) and software to deliver such services at a cost-effective price [2]. Recent trends have indicated that LBS adoption and development are on the increase. Global market revenues for such services are expected to exceed $12 billion by 2014 [2]. It is therefore vital that service developers have facilities to create, deploy and control LBS easily.

In our earlier publication, we discussed some of the issues involved in the addition of LBS support to the Mobicents SDP [3]. We modified the Mobicents Presence Service (MPS), originally designed for the transmission of presence information, for private location transmission. On the client-side, we designed a toolkit based on the Java Micro Edition (JME) Mobile Information Device Profile (MIDP) for the quick development of LBS client applications. LBS components that produce location events are known as Targets and those that consume it as Location Clients (LCs). Although there are many categorizations of LBS, our focus was mainly on enabling cross-referencing LBS. An LBS is considered as cross-referencing if the Target for a particular location information is different from the LC [4]. This includes applications such as friend-finders or locators, trackers and proximity services.

Both the location-aware MPS enabler and the JME client toolkit were created on top of existing Internet Engineering Task Force (IETF) standards. For information transport, the LBS information is transmitted either civilly, as human readable addresses, or geodetically using geospatial coordinates. The Presence Information Document Format—Location Object (PIDF-LO) specified in RFC4119 is utilized for this purpose. Of paramount importance to cross-referencing LBS information transmission is the protection of the Target’s privacy. Many proposed privacy mechanisms such as anonymisation and pseudonymisation [5] were included in previous research and implementations. However, we resolved on using privacy policies because it was more pertinent to cross-referencing LBS. Here, the IETF geolocation-policy draft was used to guide our implementation both on the MPS and on the client toolkit.

The main focus of this paper is to showcase the functionality of a SIP-based LBS toolkit and location-aware MPS that we put together. We did this through the implementation of a simple friend-finder service. A friend-finder service is a social networking service that allows friends to locate each other in like fashion to Google Latitude. To put together such a service, we used the network facilities (the location-aware MPS enabler) and the JME client toolkit we built. Our objective is to showcase the simplicity of the toolkit and to provide a simple guide to potential SIP LBS developers who may want to improve or utilize the toolkit.

II. BACKGROUND

A. LBS Components and Document Formats

LBS can be realised through several architectures and produced and consumed through a plethora of devices. However, most LBS are comprised of the same key components. According to [6] these components are:

- mobile devices
- a communication network, positioning components
- and a data and content provider.
For different LBS, mobile devices can take on the roles of either a Target, LC or both. The secure transport and communication of location data over the Internet is essentially the same as that of presence. Therefore presence documents (PIDF) are reused for the carriage of location objects (PIDF-LO) [RFC4119].

B. LBS Information Transport over SIP Networks

(Jose Costa-Requena, Heng-Te Chu, ) both propose carrying the PIDF-LO documents in simple SIP requests such as INVITE and REGISTER without any event state system [7, 8]. However, this approach is mostly suitable for self-referencing and emergency LBS. In self-referencing LBS, the Target and the LC for location information are the same. An example of this is a navigation application. Other LBS implementations in SIP-based systems treat location as a special extension to presence information using the Presence event system. (Brian Avery, Piotr Boni) are some of the LBS implementations based on the SIP presence event system in environments such as the IP Multimedia Subsystem (IMS) [9, 10]. (Maja Matijasevic) implements a friend-locator prototype using a location-enhanced presence service [11]. The Erricsson IMSInnovation project at discussed at [12] also takes a similar approach, for an Android IMS client. LCs and the Service Providers readily have both types of data (presence and location) and do not need to maintain states of two separate event systems for them. The opportunities that arise from the combination of the two data types can lead to the development of context-aware and more informative services.

Some systems propose the implementation of presence-independent location transmission. Instead of reusing the presence event system, a whole new event system, exclusively for location-based event signaling, is used. (Renato Filjar Maja) implements a presence-independent SBB or enabler namely the IMS Location Service (ILS) in the IMS Service Layer [13]. Any service that wants to request location within the IMS Service Layer can use SIP SUBSCRIBE requests to the ILS. The ILS then notifies the subscribing applications with the information they have requested. This ILS enabler provides an opportunity for new value-added location based services to be composed within the IMS. The prototype services used to test and realise the ILS are the Push-based Location Aware Messaging (LAM) and the Location-aware Push-to-talk (LaPoC) Service. Separating presence and location event systems may be beneficial because location building blocks can be reused to compose other services without the overhead of a presence system and vice versa.

A. LBS Privacy

Pseudonymisation is an approach to LBS privacy that attempts to mask the identities of the LCs or Targets (mobile devices) by using false identity information (pseudonyms) [5]. An alternative approach is anonymisation which provides privacy by ensuring location queries cannot be linked to the LCs from which they originated. We observed that pseudonymisation and anonymisation are more focused on self-referencing and pull-based LBS: These approaches focus on ensuring the privacy of the LCs, which are also Targets, during LBS-information queries at the service provider.

Research work on users has indicated that users are more concerned with privacy involving cross-referencing and tracking services than they are with self-referencing ones [14]. In location information disclosure, the important factors from the end-user’s point of view are:

- their current location
- their current context
- the identity of the requester;
- and the time of the request.

Privacy policies are defined to allow users to specify their location-sharing preferences in line with these factors. We made an observation that privacy policy mechanisms such as the IETF Geolocation-policy are more suitable for cross-referencing LBS, because the Targets explicitly specify the granularity of the location information that LCs are permitted to see [15].

III. THE LOCATION-AWARE MOBICENTS PRESENCE SERVICE (MPS) AND XML DOCUMENT MANAGEMENT SERVER (XDMS)

We mentioned in the introduction that the objective of this paper is to showcase the functionality of the facilities we put together for the quick development of LBS on SIP networks. One of these facilities is the location-aware Mobicents Presence Service (MPS) and the other is the geolocation-policy application usage (appusage) added to the Mobicents XML Document Manager (XDMS).

A. Modifying the MPS for Location-awareness

Originally, the MPS was built for presence information signaling using the presence event system [16]. The server supported the Presence Data Model [RFC4479] and Rich Presence Information Document [RFC4480] extensions in addition to basic presence information. We modified the MPS such that it can be used for signaling geodetic and civic location and only basic presence information as part of the Presence event package. The process involved creating Classes and validation Objects for Location Objects which are part of the PIDF-LO documents. With the modification complete, the Targets could publish PIDF-LO documents to the MPS based on the SIP Event State Publication specification [2]. LCs could also subscribe to the PIDF-LO documents of Targets as long as their subscriptions are based on the SIP Event Package specification [2].
The MPS also had to be modified to make the geolocation-policy rules specified by the Targets affect the granularity of the PIDF-LO Location Object information sent to the LCs. This is depicted in Figure 1 for geodetic location information publication. The MPS modifies the location of a Target located at Point (lat, long). The LCs A, B and C get PIDF-LO documents with deliberate errors according to the specified policy. This process is well explained by [15].

IV. THE LOCATION-BASED SERVICES TOOLKIT ON JME

As we mentioned earlier, a mobile device could take on the role of either a Target or an LC or both. The JME LBS toolkit we put together provides an Application Programmer Interface (API) for quickly composing Target and LC client applications. We constructed the library along the lines of location transport and privacy protection. Developers may then build their Target, Rule Maker and LC applications by using different functions from the library.

A. LBS Transport Functions for the Toolkit

We mentioned that we based the server-side LBS infrastructure on the MPS and Mobicents XDMS. For the location transport, the required SIP tasks involved publishing the PIDF-LO document (for Targets) and subscribing to it for LCs. All client applications also needed to update their contact information on the Mobicents Proxy Server so registration functionality was required as well. SIP registrations, publications and subscriptions require that a client application sends refresh requests periodically to the SIP server. We put together the JME toolkit in such a way that the details of the SIP tasks such as refreshing were completely abstracted from the developer. The main class visible to the LBS developer for SIP LBS Information Transport is the SipConnectionManager class depicted in Figure 2.

![Figure2. The SipConnectionManager class, the programming interface for LBS transmission functionality](image)

With the SipConnectionManager class, registration dialogs with the server begin as soon as a static object of the class is created. The developer provides required SIP details such as their user name, proxy address, port and SIP address. If the application being developed requires Target functionality, then the beginPublicationTask method can be invoked. The method begins publishing the PIDF-LO document composed from the location information provided by the user or from the current geodetic location obtained using the JME location API. At any time, the content of the current PIDF-LO can be modified explicitly and the publication stopped by invoking the appropriate methods of the class.

The SipConnectionManager class can also be used to implement LC functionality by LBS developers. This can be achieved by invoking the addSubscriptionTask method. The method takes in the SIP address of the Target to be subscribed to so that the toolkit can initiate a SIP subscription dialog to their PIDF-LO documents. The getSubscriptionContent method can be used to get the objects of PIDF-LO documents belonging to all active subscription dialogs. At any time, subscriptions to particular Targets may be removed or added by invoking the appropriate methods of the class.

A. LBS Privacy Functions for the Toolkit

![Figure3. The GeolocationPolicy class, the programming interface for LBS privacy functionality](image)

LBS developers have access to the GeolocationPolicy class that allows them to easily create, get, modify and delete geolocation-policy documents at the Mobicents XML Document Server (XDMS). Client applications communicate with the XDMS using the XML Configuration Access Protocol (XCAP) specified in RFC 4825. Targets or Rule Makers have to create profiles with user names and passwords before they can invoke requests on their documents at the XDMS. The Mobicents XDMS provides a web-based management console for registering such profiles [16]. It also uses digest authentication to authorise XCAP requests to private Target or Rule-Maker directories. Our JME LBS toolkit abstracts the XCAP functionality, the document validation, creation and parsing, and request authorisation from the LBS developer. The GeolocationPolicy class provides all the methods required for geolocation-policy document processing and manipulation.

The getDocument method can be used to get whole geolocation-policy documents from the XDMS. parse them and return the data in the relevant objects. All the other get methods work the same way as the getDocument method. However, they are used to get specific elements of the document. The putDocument method can be used to put whole geolocation-policy documents to the XDMS. The developer can provide a document of their own to be put on the XDMS. Otherwise the
GeolocationPolicy class provides a default document of its own. The other put methods of the class allow for the developer to add or change different elements of the geolocation-policy document. For instance, the addUserToRule method allows for a new LC address to be added to the document; the addCivicLocationToRule method allows for a new civic address to be added to a Rule. The deleteDocument method allows for the application to easily delete geolocation-policy documents. Other delete methods such as the deleteUserFromRule method are used to delete elements of the document. By using the GeolocationPolicy class, developers can build applications that can fine-tune the policy documents at the XDMs.

V. THE FRIEND-FINDER SERVICE REQUIREMENTS

We have provided a discussion of the network infrastructure and the toolkit for building LBS client applications. In this section, we showcase the toolkit by composing a friend-finder service. We are mainly interested in the core functionality (transport and privacy protection) of LBS applications rather than on their presentation. For this reason, we ignore the Graphical User Interfaces (GUIs) completely. We start the discussion by looking at the requirements of the friend-finder service. In a friend-finder application, users expect to see the location of their friends. At the same time, they should share their location so that their friends can see it too. This implies that a user should have an application with capabilities of both a Target and a LC.

The Target needs to make registrations to the SIP proxy that provides it the location services, so that its current contact details are stored there. Secondly, the Target requires the means to publish their location information. Thirdly, the Target needs to be able to Create, Update and Delete (CRUD) their geolocation-policy documents. Finally, a Target needs the ability to view, add and remove friends from a friend list. It is important to note that when a friend is added or removed, a Target has to add or remove them from their privacy policy document as well to ensure integrity.

VI. IMPLEMENTING THE SERVICE BY USING THE TOOLKIT AND LOCATION-AWARE MPS

A. Setting up the FriendFinder Class

The FriendFinder class shown in Figure 5 uses the SipConnectionManager class to perform the required location transmission functions. Listing 1 shows the constructor of the FriendFinder class. A static SipConnectionManager object is created and it is provided with the relevant details (the username, proxy address and port) to start the registration process at line 3. This is an implementation of the Register to Proxy use case. Thereafter, a static object of the GeolocationPolicy class is also created and provided with the relevant XCAP client details (the username, password, xcaproot, proxy address and port) at line 4.

The geolocationPolicy object checks if the application already has a geolocation-policy document at the Mobicents XDMS. This done by calling the getDocument method of the class at line 7. If the user is unauthorised, they might have never logged on before, so they must create a profile on the Mobicents XDMS web interface in order to use the application as shown at line 9. If no document is found at the XDMS, then the default geolocation-policy document is put at the XDMS by the client at line 15. For simplicity, we limit the users to using only three geolocation-policy rules. These are identified in the default document as closeetie, moderateeties and loosesies.

B. Initiating Publications and Subscriptions

The initiatePublications method is invoked to implement the Publish Current Location use case; the basic presence information (open or closed) and the location information (especially civic addresses) may be provided as arguments to this method. The use of the Location interface as a parameter allows for different kinds of Location objects to be provided (Point, Polygon, Circle or CivicAddress). The method calls on the beginPublicationTask method of the SipConnectionManager class. This is shown in line 3 of Listing
At this point, we make an assumption that the user manually enters the addresses of the users they wish to subscribe to in the subscribeToFriend method. The addSubscriptionTask method of the SipConnectionManager class, adds a new subscription dialog for the SIP addresses provided. To see the actual content yielded from the active subscription dialogs, the getLocationsOfFriends method of the FriendFinder class may be invoked. This method gets a collection of the PIDF-LO Objects of all the active subscriptions from the getSubscriberContent method of the SipConnectionManager class.

A. Modifying and Fine-tuning Privacy Policies

At some point, the user might want to customise the three rules (closeties, moderateties and looseties) to fit their personal requirements. They will also need to add their friends to them. The Specify or Modify Location Privacy Policy use case is implemented by using the different methods provided by the GeolocationPolicy class. The addGeodeticLocationToRule method of the FriendFinder class can be used to add new geodetic locations to the location conditions of any of the three default rules (closeties, moderateties and looseties). As it can be seen in Listing 3, the method takes arguments identifying the rule name and a unique name (identifier) describing the location being added. We have limited ourselves, in our research, to expressing geodetic location conditions using Geoshape Circles only. Therefore, an Object of the Circle class is also provided as a parameter to the method. The method invokes the addGeodeticLocationToRule method of the geolocationPolicy class (line 5). For civic location, the addCivicLocationToRule method, which calls on the addCivicLocationToRule method of the GeolocationPolicy class, may be invoked.

The transformations of the three rules may be changed by calling the changeCivicPrivacy and changeGeodeticPrivacy methods of the FriendFinder Class. These methods invoke the relevant methods of the GeolocationPolicy class in similar fashion to the addGeodeticLocationToRule method. In the next section, we discuss the View, Add or Remove Friend and Send Friend Request use cases of the friend finder service.

B. Adding Friends to Policy Documents

The user may add friends to one of the three rules by using the addUserToRule method of the GeolocationPolicy class. However, the best time to add or remove a friend to a rule may be when they are being accepted or removed as friends.

Figure 6 illustrates the process of FriendFinder A making a friend request to FriendFinder B. A first adds B’s SIP address to their buddy list (resource-list) and to one of their default geolocation-policy rules (steps 1 and 2). At step 3, A subscribes to B’s location information. Because A is not on B’s geolocation-policy rules, the subscription status of the dialog is set to pending. B later subscribes to their watcher information (watcher-info) event package and discovers that A has a pending subscription status and treats this as a friend request (steps 3 and 4). If B accepts the friend request, they add A to their buddy list and to their geolocation-policy list as well (step 7 and 8). Contrarily, if B opts to not grant the privilege of friendship to the request,
then \( A \)'s location information will be visible there until they decide to reverse the request by removing \( B \) from their buddy list and geolocation-policy documents.


To implement this functionality on JME, we perform most of the operations involved in requesting and accepting a friend as part of the `processFriendRequest` method shown in Listing 4. Line 5 implements the process shown in steps 1 and 6 of Figure 6. The variable `buddyList` is an object of the ResourceLists class that allows resource-lists to be added to the XDMS. Step 2 and 8 of Figure 7 are implemented by line 6 which uses the geolocationPolicy object used for earlier operations.

Line 7 initiates a subscription to the potential friend, and therefore takes care of steps 3 and 4 of Figure 6. The remaining steps in the Figure, 5 and 6, involve making a subscription to the Target's watcher-info package and accessing it from the notifications. We modified the constructor of the FriendFinder class so that it always starts subscribing to watcher-info package just after setting up registrations. We also created a `getFriendRequests` method that returns watcher-info objects from the watcher-info subscription dialog. We have not shown any of the error handling in the code snippet for clarity.

**VII. CONCLUSION**

In this paper we discussed the network facilities and the JME toolkit for the easy and quick development of LBS. These facilities abstract the transport and privacy protection to SIP-based LBS. The main aim of writing this Paper is to demonstrate the functionality of these facilities in their private transmission of basic LBS. To do this, we put together an application sufficient to support the general LBS requirements of a friend-finder service. In short, a friend-finder service is a social networking service that allows friends to locate each other in live fashion to Google Latitude. We are mainly interested in the core functionality (transport and privacy protection) of LBSs rather than in their presentation. For this reason, we ignore the Graphical User Interfaces (GUIs) of the applications completely.

Ndakunda Shange-Ishiwa received her undergraduate degree in 2007 from Rhodes University and is presently studying towards his Master of Science degree at the same institution. His research interests include location-based services and privacy, presence-based systems, network convergence and mobile phone services.

Madeleine Wright is currently a Java lecturer, Computer Science Department at Rhodes University. Her research interests include web technologies and mobile phone applications development amongst others.

### REFERENCES


