SIP-based Location Service Provision

By

YanHao Wu

A thesis submitted in fulfillment of the requirements
for the degree of Magister Science
in the Department of Computer Science,
University of the Western Cape

Supervisor: **Mr. Aleksandar Radovanovic**

Co-supervisor: **Mr. William D. Tucker**

December 2005

KEYWORDS

SIP

SIMPLE

Presence

Instant Messaging

Privacy

Location-based Service

Subscribe

Notify

Policy

Authorization

ABSTRACT

Location-Based Service (LBS) is a geographical location-related service that provides highly personalized services for users. It is a platform for network operators to provide new and innovative ways of increasing profits from new services. With the rapidly growing trend toward LBS, there is a need for standard LBS protocols. This thesis starts with introducing the Internet Engineering Task Force (IETF) GEOPRIV working group, which endeavors to provide standard LBS protocols capable of transferring geographic location information for diverse location-aware applications. Through careful observation, it is found that Session Initiation Protocol (SIP) is well suited to the GEOPRIV requirements. The aim of this research is therefore to explore the possibility of the integration of LBS and the SIP protocol and, to some extent, fulfill the GEOPRIV requirements. It is hoped that the result of this thesis will represent one of the few pioneering efforts providing general framework for the use of SIP-base LBS. In order to achieve this aim, the SIP Instant Messaging and Presence LBS developed by Yanhao (SIMPLY) system was developed as a Reference Implementation (RI) of SIP-based LBS. In this system two models were proposed as the potential solutions to SIP-based LBS. Both of them are implemented on the basis of the SIP Extension for Instant Messaging and Presence (SIMPLE) platform. They are implemented in this thesis not only for accommodating different users' needs, but also because they use different data transfer methods in order to find the unique advantages and disadvantages of each model, and therefore provide general recommendations for the use of SIP to provide LBS.

DECLARATION

I declare that *SIP-based location service provision* is my own work, that it has not been submitted before for any degree or examination in any other university, and that all the sources I have used or quoted have been indicated and acknowledged as complete references.

YanHao Wu Date December 2005

Signed

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

API	Application Programming Interface
AOL	America Online
CI	Common Interval
CGI	Cell Global Identity
CPP	Common Presence Profile
CTIA	Cellular Telecommunications Industry Association
FSM	Finite State Machine
GEOPRIV	Geographic Location/Privacy
GIS	Geographical Information Systems
GML	Geography Markup Language
GMLC	Gateway Mobile Location Centers
GSM	Global System for Mobile communications
HTTP	Hyper Text Transfer Protocol
IETF	Internet Engineering Task Force
IM	Instant Messaging
IMPP	Instant Messaging and Presence Protocol
IP	Internet Protocol
JDBC	Java Database Connectivity
LBS	Location-Based Service
LIS	Location Information Subscriber
LIN	Location Information Notifier

LPIDF	Location-enhanced Presence Information Data Format
MMS	Multimedia Message Service
MPC	Mobile Positioning Centers
MPP	Mobile Positioning Protocol
MRI	Minimum Reliable Interval
NGN	Next Generation Network
OGC	Open Geospatial Consortium
OMA	Open Mobile Alliance
PE	Presence Engine
PIDF	Presence Information Data Format
PoC	PTT over Cellular
PSAP	Public Safety Answering Point
PTT	Push-to-Talk
RFC	Request For Comments
RFID	Radio Frequency Identification
RI	Reference Implementation
RTP	Real-time Transport Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SIMPLE	SIP for Instant Messaging and Presence Leveraging Extensions
SLO	Spatial Location
SLoP	Spatial Location Protocol
SMS	Short Message Service

TCP	Transmission Control Protocol
UA	User Agent
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
VoIP	Voice over Internet Protocol
WGS84	World Geodetic System 1984
XML	eXtensible Markup Language
XMLNS	XML namespaces
3GPP	Third Generation Partnership Project

ACKNOWLEDGEMENTS

I wish to convey my greatest gratitude to GOD almighty who guides me all the way of my life and watches over me every step I make. To Him be glory for ever and ever.

I want to give special thanks to our research group leader William D. Tucker who gave me the greatest input that I could ever get. He is a group leader, a co-supervisor, more importantly, a great friend who occupies a special place in my heart.

I want to give special thanks to my supervisor Aleksandar Radovanovic who has guided me all the way in my Masters' study and has cared for me. He is not only a supervisor, but more importantly, a great friend who is always there for me.

Sandro Da Silva and Elroy Peter Julius deserve my gratitude for their great input to my thesis.

Special thanks are due to the NIST SIP Open Source group that provided the foundation for the SIMPLY system.

I would also like to thank the Telkom/Cisco/THRIP Center of Excellence (CoE) that provided me with necessary equipment and sponsoring my studies.

Finally, I would like to thank my parents, XuMing Wu and YingLan Cheng for providing me with moral and emotional support during the lows, when I had no more to give. Without them, I would never have completed this dissertation.

CHAPTER 1 Introduction

This chapter starts with a brief introduction to Location-Based Services (LBS). LBS is a geographical location-related service that provides highly personalized services for users. Four types of LBS are introduced to give an overview of its uses. They are location based-billing, advertising, tracking and emergency services. In general, LBS is a platform for network operators to provide new and innovative ways of increasing profits from new services. With the rapidly growing trend towards LBS, there is a need for standard LBS protocols. The Internet Engineering Task Force (IETF) GEOPRIV working group endeavors to select already standardized protocols and recommend for use. Through careful observation, it is found that Session Initiation Protocol (SIP) is well suited to the GEOPRIV requirements. The aim of this research is therefore to explore the possibility of the integration of LBS and the SIP protocol, and to some extent, to fulfill the GEOPRIV requirements.

1.1 Location-based Service

The first indoor location system, Active Badge was developed in 1992 (Want, Hopper, Falcao and Gibbons, 1992). Subsequently, projects like PARCTab (Want, Schilit, Adams, Gold, Petersen, Ellis, Goldberg and Weiser, 1995) and Cyberguide (Roach, 2002) show that location systems have been drawing more attention since the mid-1990s. The year 1997 witnessed the beginning of the adoption of location systems by the telecommunication industry for commercial use to provide LBS. There are three main components in LBS (see Figure 1.1). These include: a positioning system, an LBS application and a Geographical Information System (GIS).

The development of LBS can be divided into two phases. Phase 1 spanned 1997 to 2001. Telecommunication companies approach to LBS was to implement core nodes within the Signal System 7 (SS7) network that was capable of extracting the location of mobile devices from the network (Spinney, 2003). Mobile Positioning Centers (MPCs) and Gateway Mobile Location Centers (GMLCs) were developed as core

components at that stage. MPCs and GMLCs were used to provide cellular users with location information. Since then, LBS became part of the telecommunication services provided by telecommunication companies. In Phase 1, there was no specified standard interface to access MPC/GMLC. All LBS applications in the IP domain had to develop their own interfaces to access the MPC/GMLC. There is no standardized specification for commercial Internet GIS mapping servers either. Most GIS mapping servers had their own eXtensible Markup Language (XML) proprietary schemas.

Figure 1.1 Location-Based Services Phase 1

The Positioning System provides positioning technologies for a LBS application. GIS, which could be thought of as a map on a computer, is used for mapping and analyzing geographic information. LBS applications are used to provide services for end users based on GIS and the positioning system.

With the growth of LBS, application developers wanted to implement many LBS applications smoothly and quickly. It was expensive to develop and implement an LBS system because of non-interoperability among GIS, LBS applications and MPC/GMLC. Again, GIS and MPC/GMLC needed to be thoroughly analyzed in order to yield specific requirements. The LBS application could then be implemented specifically according to the requirements. Due to the integration problems encountered within Phase 1, a middle layer was introduced into LBS. As Figure 1.2 shows, application developers can use standard format and standard LBS protocol instead of developing new sets of Application Programming Interfaces (APIs) for GIS and MPC/GMLC.

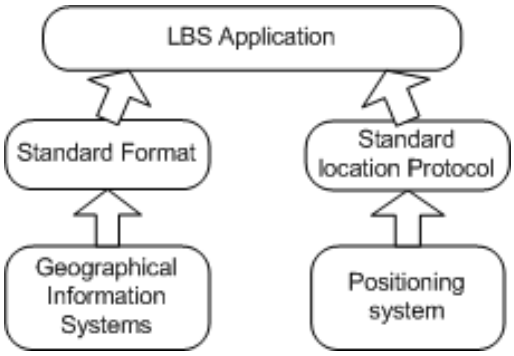


Figure 1.2 Location-Based Service Phase 2

Standard format and standard location protocols have been introduced into LBS as a middle layer in order to provide inter-operability. Different applications can use the same format and protocol to interact with a range of GIS and positioning system architectures.

This model ushered in phase 2 starting from about 2002. The Phase 2 model ensures that any application developer can develop an LBS application through standard interfaces without knowing the full details of the core network. The new model has dramatically reduced the implementation costs. The Open Geospatial Consortium (OGC) has contributed considerable effort towards LBS standardization. The OGC's APIs provide a finite invariant set of spatial processing function interfaces which include geocoding, reversing geocode, spatial querying, mapping and routing. The OGC's objective is to provide a standardized solution for the service operators (Spinney, 2003). The standardized solution allows a service provider to implement standard interfaces for their LBS architecture. Therefore, service developers have standard tools to use when they build LBS applications.

1.2 Types of LBS

LBS is a geographic location-related service to help people to obtain highly personalized information. For example, Bob is in his car, at the mall, or somewhere outside and wants to know whether there is a traffic jam where his spouse is, or how to find the closest bakery. In the age of telecommunication competition, network operators continuously seek new and innovative ways to create differentiation and increase profits. LBS is a growing area for service providers looking to fulfill new customer needs, and for network operators looking to augment their revenue stream

(Rao and Minakakis, 2004). LBS provides highly personalized services for customers. Different types of location-based services have emerged with respect to different types of consumer demands. Four categories of LBS are briefly introduced below.

1.2.1 Location-based billing

Using location-based billing, wireless service providers can charge a different rate for each zone. A wireless carrier could charge subscribers a low flat rate for home and office use and another rate for outside those areas. It brings the convenience of having just one phone and one phone bill for telecommunication customers. In the same vein, it stirs existing subscribers to make telephone calls, which generates additional revenue for service providers. Openwave location-based billing (http:// developer.openwave.com) allows service providers to customize calling rate zones to fit the needs of an individual subscriber. LSB is mostly used by wireless service providers, enabling them to effectively compete with wire-line service providers.

1.2.2 Location-based advertising

LBS offers a way for service providers to generate advertising revenue. LBS can also be used for commercial adverts to provide information according to user's predetermined desires, such as notification of a sale of men's suits at a store close to the user's current proximity. A shopping mall can use LBS to advertise products for all shops within the mall by using the Short Message Service (SMS), for example, to inform a wireless LBS subscriber visiting the mall about a new flavour coffee if he is passing by a coffee shop. This in turn could mean additional sales for the entire shopping mall, or an overload of SMS adverts.

1.2.3 Location-based tracking

Tracking is a relatively large category that is useful for locating friends, vehicles and valuable assets. Nowadays, communication has become part of people's daily life. People sometimes would like to know where their friends are. Companies would like to know where their mobile assets (trucks, cars etc.) are. The Webraska Buddy Finder (http://www.webraska.com) allows users to automatically locate friends, family and

colleagues, and receive notifications of their proximity via Wireless Application Protocol (WAP), World Wide Web (WWW) and SMS. This service provides a pre-built application framework to display the locations of individuals or groups of people over multiple bearer channels. It translates geo-coding into a readable text-based location as well as displaying the location and its proximity on a map.

Webraska's Asset Tracking service (http://www.webraska.com) has tremendous implications for companies and individuals with a vested interest in tracking the exact location of mobile assets. Asset Tracking can be used to aid a trucking company in keeping track of trucks and deliveries, or a rental company in locating and recovering lost or stolen vehicles. Trucking companies are good examples of how this service can help businesses manage their supply chains and their labour resources through a better understanding of shipment timing. For example, a half empty truck sent to deliver certain goods to a location on the same route of another customer's location who has urgently requested an emergency delivery, can be redirected to pick up the additional load, therefore saving the cost of sending out another vehicle to pick up the delivery. In addition, this type of service can also be used to track the location of wireless GPS-enabled merchandises such as telephones, laptop computers, and PDAs.

1.2.4 Location-based emergency services

Location-based emergency services are used for pinpointing a user location and relaying it to the appropriate authorities, e.g. reporting a fire alarm to the fire department or a car accident to the traffic office. IP Telephony can support emergency services such as 911 call services in the USA, 110 in Germany, and 112 in the rest of Europe. The Hewlett Packard (HP) standards-based emergency services solution enables service providers to pinpoint the location of a subscriber (http://government.hp.com). Whenever the subscriber dials 911/112, the location information can be provided to the appropriate local Public Safety Answering Point (PSAP). The HP solution has been shown to be reliable in mixed infrastructure

networks and it currently provides emergency services to more than 100 million subscribers.

1.3 GEOPRIV Working Group

Since LBS is a platform that can provide new and innovative ways of adding valuable services, there is a need for standard LBS protocols. The GEOPRIV Working Group was formed at the 51st Internet Engineering Task Force (IETF) meeting in August 2001. The primary goal was to select and recommend already standardized formats and protocols for the secure distribution of location information. A key task was then to enhance selected formats and protocols to provide a service capable of transferring geographic location information for diverse location-aware applications. Two important concepts of the GEOPRIV working group are "Location Object" (LO) and "Using Protocol". SIP and GEOPRIV "Using Protocol" were found to share many characteristics. This thesis suggests that SIP is well suited to the category of GEOPRIV's "Using Protocol". Moreover, the SIP message can be easily extended with new features for adopting the attributes of the GEOPRIV LO.

1.3.1 The GEOPRIV Location Object and the SIP message

The GEOPRIV LO is a defined format used to convey both basic privacy-protecting instructions and location information (Cuellar, Morris, Mulligan, Peterson, and Polk, 2004). The LO contains the location information of the target, and other fields including an identity of the target, time information, core privacy rules, authenticators, etc. However, most of the fields are optional, including the location information itself.

The SIP message is used to convey both message bodies and header fields. All SIP messages may include a body (Rosenberg, Schulzrinne, Camarillo, Johnston, Sparks, Handley and Schooler, 2002). Location information can therefore be inserted into the message body. A valid SIP Request must contain the following mediatory header fields: "To", "From", "CSeq", "Call-ID", "Max-Forwards", and "Via". These mandatory header fields specify the identity of the initiator and the recipient of the

request in order to route the SIP message. There are some optional header fields that might be included in a SIP message for the purposes of authorization, authentication and time information (such as: "Authorization", "Authentication-Info", and "Timestamp"). More importantly, new header fields can be defined which provide the possibility to accommodate the GEOPRIV LO.

1.3.2 GEOPRIV "Using Protocol" and SIP

The GEOPRIV "Using Protocol" is the protocol that carries and uses the Location Object. A protocol that just transports the LO as a string of bits, without looking at them (like an IP transport protocol would do), is not a "Using Protocol", but only a transport protocol (Cuellar et. al, 2004). The "Using Protocol" has to obey the privacy and security instructions coded in the LO. When transmitting the LO, the sender and the receiver must agree on the data type of the location information. The "Using Protocol" may specify that the data type information is part of the LO or that the sender and receiver have agreed on it before the actual data transfer.

SIP is a text-based application-level protocol. Like any other protocol, SIP carries messages to fulfill certain tasks. It is not just a transport protocol that passes on the SIP messages as a string of bits without looking at them. When a SIP message arrives, SIP protocol needs to inspect the headers of the message. SIP checks the instructions coded in them and reacts accordingly. SIP can also be used to negotiate a common format for describing sessions because different session participants might have different multimedia capabilities. SIP is well known for its flexibility and scalability. SIP extensions such as SIP Presence and IM equip the SIP with more capabilities (Day, Rosenberg and Sugano, 2000), thus provide the possibility of integration with LBS.

1.4 Motivation

SIP has been chosen as the key technology to shift from the old fashioned intelligent networks to the integration of an all-IP network in terms of service architectures, service development approaches and programmability (Canal and Cuda, 2001). **The**

motivation behind this thesis is to explore the possibility of the integration of LBS and SIP, and therefore design for the use of SIP in line with LBS. Regarding to the IETF GEOPRIV working group, the approach this group uses is to select and recommend already standardized formats and protocols for the secure distribution of location information instead of defining new location information formats and protocols. SIP plays an important role in the Next Generation Network (NGN). It is a successful protocol that is widely accepted by industry. Many designers have accepted the SIP protocol and are using it. Therefore it is applicable to extend SIP rather than develop a new protocol.

1.5 Research questions

The research question is "**How can SIP be used as a "Using Protocol" for LBS to fulfill GEOPRIV requirements?"** This thesis aims at enhancing SIP as a GEOPRIV "Using Protocol". The outcome of this work is hoped to partially fulfill GEOPRIV requirements as a starting Reference Implementation (RI) of SIP-based LBS. This research question is divided into two sub-questions.

LBS can provide location services for location-tracking, billing, advertising, and emergencies. It is a useful and value added service to end users as well as service operators. However, LBS is used to transfer highly personalized location information that also presents the potential to reveal someone's sensitive information. Thus, the first sub-question of this thesis is: "**How can SIP be used to transfer location information in a private manner?"**

SIP is well known for its flexibility. It is easy to extend and adopt new features for new uses but the robustness of new features still needs to be evaluated. In order to provide Quality of Service (QoS), proposed solution should remain reliable and stable when dealing with situations like fast location data transfer, large number of data transfers, and long-term data transfer. Thus, the second sub-question is: "**How can**

the SIP-based solution proposed by this thesis be used to provide stable services for LBS?"

1.6 Research aim and significance

The aim of this research is to partially fulfill the requirements of GEOPRIV with respect to the following four aspects:

- A basic XML schema should be defined to contain minimum location information. This schema should be flexible and easy to extend in order to adopt new features for further development.
- The disclosure of location information should be directly and dynamically controlled by the end user in order to provide privacy.
- Multimodal location-based services should be implemented in order to accommodate different users' needs.
- Different data transfer methods should be investigated before recommending for use.

Note: This thesis is not attempting to completely fulfill the requirements of GEOPRIV. Embedding privacy-protecting instructions inside the SIP message is out of the scope of this thesis. The Presence community has defined a policy protocol and schema set called XML Configuration Access Protocol (XCAP), through which privacy rules can travel along with location information, and thus guide the Presence entities dealing with the location information (Rosenberg, 2004). This aspect is not addressed in this thesis, but is suggested for future work.

Since February 1st 2005, VoIP became legal in South Africa. Two months after deregulation of the telecoms industry came into force, South Africa witnessed a number of VoIP companies entering the market as telecommunication service providers capitalizing on the legalisation of this voice service. SIP applications should gain a wide range of support in South Africa. Much work has been done to demonstrate that SIP will pave the way for NGN by bringing significant advantages: SIP guarantees inter-operability between network systems and applications and SIP-

based architectures are open, scalable and flexible (Canal and Cuda, 2001). It is therefore natural to use SIP for this thesis not only because SIP applications can gain wide support, but also because SIP has the potential to provide a superior model to transfer data in a well-structured format. Technology like GPS can provide global outdoor location information. Researchers also use GSM/UMTS, Bluetooth, WaveLAN, RFID IrDA and Wireless LAN to provide local-area and indoor positioning. More importantly, all of these technologies could be and have been integrated into IP telecommunication services. Hence, it is practical in this thesis to put SIP into real use for LBS. It is hoped that this thesis will represent one of the few pioneering efforts in the use of SIP for LBS.

1.7 Thesis outline

The remainder of this thesis is organized as follows:

Chapter 2 – Literature review This chapter presents work related to LBS and SIP, as well as the relationship between them. The significance of privacy in LBS is discussed with five privacy requirements: dynamic response to circumstance, built-in plausible deniability, coarse-grained control, feedback and special exceptions for emergencies.

Chapter 3 – Research methodology The related work leads to a technical solution and approach to evaluating it. The purpose of this Chapter is to provide the research methodology to answer research questions. An empirical evaluation methodology is offered in which comprises three research methods: laboratory simulation, comparison between two models (PUSH and PULL), and software agent data collection.

Chapter 4 – System design This chapter presents the implementation details of two models, PUSH and PULL. The chapter discusses the fundamental architecture they are based on in order to address the research questions. The design of the proposed system illustrates how to enhance the SIP Presence function to provide a PUSH model, and the IM function for a PULL model LBS.

Chapter 5 – Experiment design This chapter presents the experimental design for the comparison of PUSH and PULL models. It shows how the empirical evaluation methodology helps to answer the research questions. It is used to evaluate the quality of the proposed solutions. The experimental design involves the following: preparation of the testing environment, execution of the experiments, and collection of the experimental data.

Chapter 6 – Data visualization and analysis This chapter displays the data from the four experiments described in Chapter 5: system functionality test, reliability test, stability test, and redundancy test. The purpose of the first experiment is to test if the implementation is functioning as required. The other three experiments are designed to compare the performance of PUSH and PULL models. The obtained data reveals the advantages and disadvantages of each model in order to provide general recommendations for the use of SIP to provide LBS.

Chapter 7 – Conclusion and future work This chapter answers the research questions and gives the conclusion. The recommendations for the use of SIP to provide LBS are presented. The thesis has a clear scope, which is to partially fulfill the requirements of GEOPRIV as a starting point for SIP-based LBS. Hence, the direction for its advanced development is presented for the further research.

CHAPTER 2 Literature review

LBS is a growing area for service providers seeking to satisfy new customers' needs and assist the network operators aiming at increasing their revenues (Rao and Minakakis, 2004). With the rapidly growing trend of LBS, application developers would want to implement many LBS applications smoothly and quickly. It has proved slow and relatively expensive to develop and implement an LBS system because of the lack of standard APIs and non-interoperability among private LBS protocols. Therefore, two standard LBS protocols were developed for requesting and transferring location information i.e. Mobile Positioning Protocol (MPP) developed by Ericsson and Spatial Location Protocol (SLoP) developed by IETF. However, these two protocols have their own limitations. MPP is built on the foundation of the traditional telephony service and it cannot be used to utilize other positioning technologies. The SLoP has not been widely accepted because some other existing protocols can replace its functionality. Hence, the IETF GEOPRIV working group was founded to select and enhance already standardized protocols. SIP plays an important role in the NGN. SIP has been chosen as the key technology for changing from the old fashioned intelligent network to an all-IP network due to its flexibility and extensibility. This chapter presents the related work with respect to LBS and SIP, as well as the relationship between them. In this chapter the significance of LBS privacy is also discussed. LBS is used for transferring highly-personalized information. Hence, it also involves potential risk of revealing sensitive information, which will certainly reduce the customer's appreciation for LBS. Privacy is an important factor which deserves significant attention. On the basis of previous LBS privacy research, five privacy requirements have been summarized and presented: dynamic response to circumstance, built-in plausible deniability, coarse-grained control, feedback and special exceptions for emergencies.

2.1 Standard Protocols for LBS

The LBS standard protocols make the development of LBS applications easier and quicker because it deals with all the necessary data parsing as well as providing standard APIs for developers. The LBS application developers do not need to study and fully understand the underlying physical positioning technology. Moreover, with standard LBS protocols, developers can develop LBS applications independently, and all the LBS applications based on the same LBS protocol can inter-operate with each other.

2.1.1 Mobile Positioning Protocol

Mobile Positioning Protocol (MPP) is an application level protocol particularly developed for the Global System for Mobile communications (GSM)-based positioning service. As LBS becomes part of the telecommunication service, the telecommunication operator provides LBS for GSM network users. MPP has been proposed by Ericsson as a standard protocol for requesting and transferring location information for GSM users. The Mobile Positioning Center (MPC) is the gateway between the mobile network and LBS applications. The MPC calculates the position of a mobile terminal through positioning technology like Cell Global Identity (CGI). It uses GSM base stations to locate a user. The coverage area of one base station can be seen as a cell. The accuracy is dependent on the cell size and varies from 10m (a micro cell in a building) to 500m (in a large outdoors macro cell). As shown in figure 2.1, the CGI retrieves location information from the GSM network and delivers it to the LBS application. Privacy is one of the major concerns of MPP. A request to retrieve a user's location information must be authorized and permission can be restricted to certain mobile phones or granted only to the emergency services. Ultimately, mobile subscribers can choose not to be located.

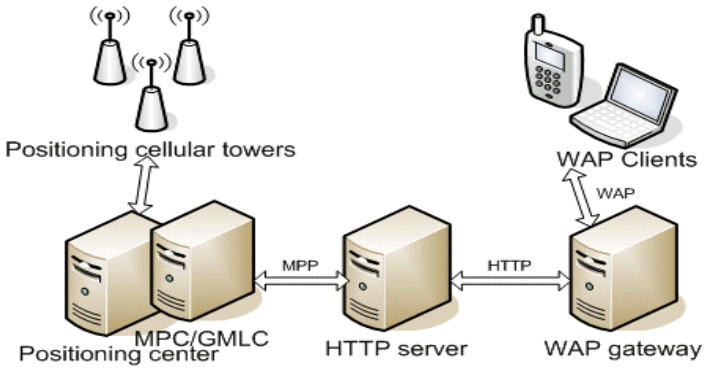


Figure 2.1 LBS architecture with MPP

The MPP uses Hyper Text Transfer Protocol (HTTP) to transfer location information, thus making the MPC available also for platforms with TCP/IP capability. It calculates the position of a mobile terminal through cellular towers and delivers it to HTTP server. The MPP defines a Universal Resource Locator (URL) that location-aware applications can use to request the position of a mobile device. As a response to the position request, the positioning center informs the application about the mobile device's location through WAP gateway.

Standard APIs of MPP have been provided by Ericsson, which also includes HTTP interfaces for location information transmission. The APIs offer a carefully designed generic interface towards the MPC, thus making the applications independent of the underlying physical positioning technology. As a result, location-dependent applications will directly take advantage of it by focusing only on the system function development. The application developers do not need to know how MPC is used when retrieving the position of a mobile terminal. However, MPP has its own limitation; it is positioning platform-related, built on the GSM platform and cannot be utilized by other positioning systems.

2.1.2 Spatial Location Protocol

The IETF has proposed a standard LBS protocol called Spatial Location Protocol (SLoP). The purpose of SLoP is to provide a standard way for LBS applications to acquire the spatial location of an identifiable resource over the internet in a reliable, secured, and scalable manner (http://www-nrc.nokia.com/ietf-spatial/). For inter-operability reasons, the SLoP uses an absolute location format. This location format is required to be supported by all the SLoP supporters. The WGS84 geodetic datum is

used as a default reference system (http://www.wgs84.com). It specifies the location in longitude, latitude and altitude parameters. The location format is composed of the following data items: User location type, Framework, Format, Geocentric, Position, Accuracy, Time stamp and Time-to-live (Rosen, Costa, Korkea, Ylianttila, Mahy, Takahashi and Farrell, 2000). Other than this, the SLoP is capable of supporting other location representations. It lists all the other absolute location formats and supports them on an optional basis. Privacy is also one of the major concerns of this Protocol. The SLoP defines a set of privacy policies to protect user-sensitive location information. A specific target's (end user's) policy is stored on a policy server, where the policy indicates how a server shall process the spatial location of the target. Server is able to adjust a location disclosure based on authorization policy, as well as the client being able to request certain elements of the whole location information.

In communication terms, SLoP supports UDP transport with retry timers for reliability and TCP as an optional transport. Hence, SLoP servers are accessible from any platforms with TCP/IP capabilities. However, SLoP has not been widely accepted and well-developed because some other existing protocols can replace its functionality.

2.2 Privacy for LBS

LBS greatly increased the value of IP telecommunication services both for customers and service providers. Nevertheless, LBS is used for transferring highly-personalized information. It involves the potential risk of revealing sensitive information that will certainly reduce the customer's appreciation of using LBS. Privacy is therefore an important factor that deserves considerable attention.

2.2.1 Privacy is the key

At the beginning, location systems like the active badge system (Want et. al, 1992) did not take privacy into account. Active badge detects the location of each user and broadcasts the information to everyone in the building. The system originally

deployed assumes everyone in the building is trustworthy; it therefore provides no mechanism to limit the dissemination of individuals' location information (Beresford and Stajano, 2003). Once a person puts on the badge, his location information will be broadcast to everyone in the building. People feel uncomfortable and insecure and even tired of using it. As the designer himself points out that "there will always be some days when for whatever reason somebody does not wish to be located, he can remove the badge and leave it on the desk when he does not want to be located"(Want et. al, 1992). Since the LBS privacy is acknowledged, designers have started to put more effort into privacy issues. Subsequently, privacy has been considered a part of the original system design. The Cricket Indoor Location System was developed at MIT (Priyantha, Chakraborty and Balakrishnan, 2000). As a location support system, Cricket allows clients to control their own location. There is no central entity to keep track of each individual component in the system, and location data is delivered to a personal digital assistant under the sole control of the user.

Actually, privacy is closely related to the human being itself. Westin once defined information privacy as the claim of individuals, groups or institutions to determine for themselves when, how, and to what extent information about them is communicated to others (Westin, 1967). Obviously, location systems are designed to serve customers instead of embarrassing customers. As mentioned earlier, it is a known fact that privacy is a potential danger which is threatening and reducing the customer's appreciation of LBS. Privacy of LBS is therefore particularly surveyed in the research that follows.

2.2.2 Privacy requirements for LBS

Privacy concern, in general, is closely related to human psychology, social custom and a wide range of conceptions. It is difficult to portray privacy as a whole. Hence, based on related work done by other researchers, five specific facets are pointed out that this research could focus on and deal with.

Dynamic response to circumstance

The Notifier's willingness to disclose his/her geographic information primarily depends on who is requesting and why. Depending on the social relationships between the notifier and the subscriber, the response might be quite different from one subscriber to another. While traditional approaches understand privacy as a state of social withdrawal, Altman instead sees it as a dialectic and dynamic boundary regulation process (Altman, 1975). Privacy management is not about setting rules and enforcing them; rather, it is the continual management of boundaries between different spheres of action and degrees of disclosure within those spheres.

Built-in plausible deniability

Hindus suggested a need to avoid potentially embarrassing situations, undesired intrusions, and unwanted social obligations (Hindus, Mainwaring, Leduc, Hagström, Bayley and Casablanca, 2001). A good example of this is with mobile telephones. If a person does not answer a mobile phone call, it could be for technical reasons such as being outside of the service range, not having the phone on him/her or that the telephone is off. It could also be for social reasons such as being busy or not wanting to talk to the caller at the time the call came in. It is a simple model for protecting the user's privacy, while the caller cannot tell why that person is not answering. By default, it does the right thing without the end-user having to take any special action.

Coarse-grained control

LBS needs coarse-grained control. A LBS user should have a way to stop or adjust the information disclosure to the level that the user wants to reveal to others. Lederer and Hong suggest that ubiquitous computer systems that convey location could incorporate both a precision dial (ordinal) and a hide button (binary) such as the volume and mute controls of audio devices, so that users can either adjust the precision at which their context is disclosed or decidedly halt disclosure (Lederer and Hong, 2004). This helps users to accommodate the controls and even co-opt them in ways the designer may not have intended.

Feedback

It is important for a user to know his/her actual information disclosure. Users will have difficulty in appropriating a system into their privacy practice if the scope of its privacy is unclear. With feedback mechanisms, it could provide social visibility to prevent abuses (Hong and Landay, 2004). For example, Alice is less likely to repeatedly query Bob's location if she knows that Bob can see each of her requests. A user feels comfortable when he/she has the capability to control the disclosure of his/her sensitive information. Simple feedback and indicator mechanism can provide social visibility to prevent abuses.

Special exceptions for emergencies

In emergency situations, safety far outweighs privacy needs. An emergency should be given the privilege of being treated specially. IP Telephony supports emergency purposes such as E911 call services in USA, 110 in Germany, and 112 in the rest of Europe (Costa and Tang, 2002). Hospitals, for example, may require up-to-date information about the location of patients, particularly when medical emergencies arise. Trusted proxies are sometimes used to handle these kinds of situations. And people are willing to pay for this service. For example, MedicAlert (http://www.medicalert.org) is a paid service that stores personal medical records and forwards it to emergency responders in the case of medical emergencies.

2.3 SIP and LBS

In meeting the above five privacy needs, SIP is seen as a promising protocol in this research to handle both the standardizing of the transmission and preventing unwanted disclosure of location information. SIP is a signaling protocol used for establishing sessions in an IP network. SIP extensions equip SIP with more capabilities. SIP carries a payload to fulfill a certain set of tasks. The payload contains the main headers and might contain a message body with the details of how to establish a multimedia transaction. It can be easily extended with new features for adopting other services due to its open and scalable architecture.

2.3.1 SIP Introduction

The original draft of what was to become SIP standard started back in February 1996. The first IETF draft, which was titled "draft-ietf-mmusic-sip-00", includes only one request type, which was a call setup request. Three years later, the IETF published the RFC 2543 document after twelve revisions, which was called "Session Initiation Protocol" (Handley, Schulzrinne, Rosenberg, 1999). It contained the six requests which people are now familiar with: INVITE, ACK, OPTIONS, BYE, CANCEL and REGISTER. SIP was designed on the basis of an open and scalable architecture by the IETF. SIP was originally conceived to improve the set-up and handling of IP telephone service. As more SIP extensions emerge afterwards, they provided more possibilities for developing personalized telecommunication services as well as simplifying all forms of real-time communications.

2.3.2 SIMPLE

SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) is a contrived acronym that describes a body of work going on in the IETF. SIMPLE builds upon the SIP and uses it for multimedia communication signaling over the IP network by adding Presence and Instant Messaging (IM) functionality. SIMPLE is used for multimedia communications such as voice, video, application sharing, messaging etc. It is an important step in bringing standardization to Presence service and Instant Messaging. A key area of usage for SIMPLE is in Push-to-Talk (PTT), which is also known as PTT over Cellular (PoC). This is a hot cake in new wireless service market. It provides a walkie-talkie experience over cellular network (Woodruff and Aoki, 2003).

The primary value proposition of Presence service is that all types of communication are linked by it. Presence connotes a user's willingness, ability and desire to communicate across all different kinds of media types. Before a user makes any kind of communication attempt with SIP, whether it's to setup a VoIP call, a video-conference, or an IM chat, the Presence indicates the willingness of the recipients to

participate in that session. Two SIP extension methods are proposed to handle Presence services: SUBSCRIBE and NOTIFY. Presence service is used to accept, store, and distribute Presence information. Historically, Presence information has been limited to “on-line” and “off-line” indicators. The notion of Presence becomes broader nowadays. Presence information can include personal information (name, age, ID number), activity information (in a meeting, playing basketball) and location information, on which this research is focusing.

The IM refers to the transfer of messages between users in near real-time. These messages are usually, but not required to be, short. IM is often used in a conversational mode, that is, the transfer of messages back and forth is fast enough for participants to maintain an interactive conversation. RFC3428 proposes the MESSAGE method which is an extension to the SIP that allows the transfer of Instant Messages (Campbell, Rosenberg, Schulzrinne, Huitema and Gurle, 2002). Since the MESSAGE request is an extension to the SIP, it inherits all the request routing and security features of that protocol. MESSAGE requests carry the content in the form of Multipurpose Internet Mail Extensions (MIME) body parts (http://xml.coverpages.org/ni2004-02-17-a.html). MESSAGE requests do not themselves initiate a SIP dialogue; under normal usage each Instant Message stands alone, much like pager messages, which might provide another way of transferring location information.

Authorization and privacy play a key role in SIMPLE systems. A user needs to be able to block other users from seeing their Presence information, or allow them to see only specific pieces of information. As part of the unification between Presence and geolocation, the IETF has developed a common framework for representing authorization and privacy statements. This framework defines a basic XML document format for representing authorization and privacy information. Presence-specific and geolocation-specific permissions are then defined within that framework. Additional extensions can be defined that provide permissions for instant messaging, telephone

calls, video conferences, and so on. In this way, SIMPLE provides a common privacy system for multimedia communications, rather than separate ones for each individual application (http://xml.coverpages.org/ni2004-02-17-a.html).

2.3.3 PIDF

Presence Information Data Format (PIDF) is standard Presence format for SIP and widespread in the Instant Messaging community (Sugano, Fujimoto, Klyne, Bateman, Carr and Peterson, 2004). Its Internet draft was first published through the IETF Instant Messaging and Presence Protocol (IMPP) working group in 2003. PIDF was proposed as a common data format for Common Presence Profile (CPP) compliant protocols, which allows Presence information to be transferred across CPP-compliant protocol boundaries without modification.

PIDF is encoded in XML language and has minimal set of Presence status values (Day, Aggarwal, Mohr and Vincent, 2000). The root element of a PIDF document is <tuple>, which consists of a mandatory <status> element. The <status> element contains one optional <basic> element, followed by any number of optional extension elements. The <basic> element has two values "open" and "closed". These values indicate availability to receive and send Instant Messages. A user can chat with other people while his status is "open", but he cannot when the status is "closed". However, as mentioned earlier, Presence indicates a user's willingness, ability and desire to communicate with others. If someone does not want to be disturbed by unwanted visitors, he can close his status while remaining online which refers to "hiding" or "appearing offline". The status automatically becomes "closed" when the user disconnects from the notification server.

PIDF is considered as a well-structured and extensible format. Except for a minimal set of Presence elements and values, a Presence application is able to define new elements and status values. The Presence information extensibility framework is based on XML Namespaces (XMLNS). All elements and some attributes are

associated with a XMLNS, which in turn is associated with a globally unique Uniform Resource Identifier (URI). Figure 2.2 shows how developers can introduce their own element names as well as avoiding conflict by choosing an appropriate URI (Sugano H. et. al, 2004). For the above features, PIDF is considered as the most desirable format over other candidates to accommodate location information in this research.

```
<?xml version="1.0" encoding="UTF-8"?>
  <presence xmlns="urn:ietf:params:xml:ns:pidf"
     entity="pres:someone@example.com">
    <tuple id="sg89ae">
      <status>
        <basic>open</basic>
      </status>
      <contact priority="0.8">tel:+09012345678</contact>
    </tuple>
  </presence>
```

```
<?xml version="1.0" encoding="UTF-8"?>
  <presence xmlns="urn:ietf:params:xml:ns:pidf"
     xmlns:local="urn:example-com:pidf-status-type"
     entity="pres:someone@example.com">
    <tuple id="ub93s3">
      <status>
        <basic>open</basic>
        <local:location>home</local:location>
      </status>
      <contact>im:someone@example.com</contact>
    </tuple>
  </presence>
```

Figure 2.2 Using namespace to extend PIDF

Developers can use the standard namespace-based extensibility rules to define their own status values. The document on the left side is a standard PIDF document while the PIDF document on the right side uses XML namespace to extend a new element "local" and its attribute "location" through pointing another defined XML schema by using namespace.

2.3.4 Using SIP payload for transferring geographic information

Costa and Tang worked on enhancing SIP with the required spatial location capability for supporting the mandatory emergency call services (Costa and Tang, 2002). These services are required to be universally available regardless of the network type on which the user is registered and the device he/she is using. In their research, the Spatial Location (SLO) information is considered as session data to be carried in the SIP payload instead of the usual Session Description Protocol (SDP) packets (see Figure 2.3). The "Content-Type" header indicates the different payload that is embedded in the message. The "Content-Encoding" header indicates the coding type the body. The body length is given in the "Content-Length" header field. The SIP is used as a call control protocol in IP Telephony. It is a remarkable fact that adding location information as part of the signaling flow facilitates the development of location-based services. This also enriches the telecommunication services.

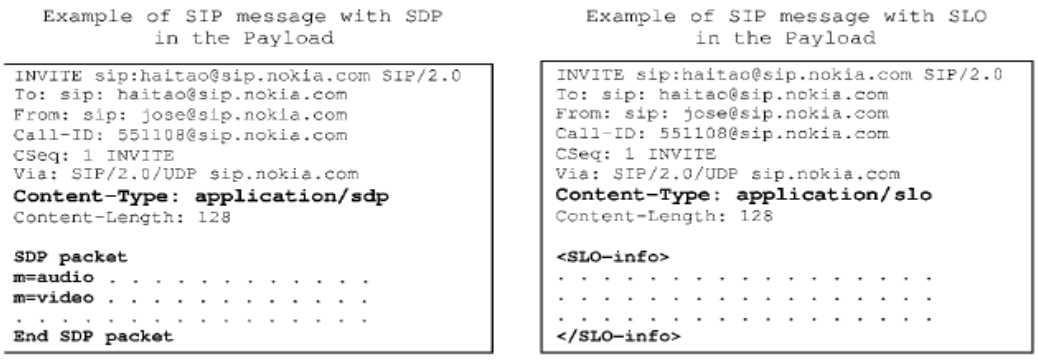


Figure 2.3 Comparison of SDP and SLO in the SIP payload

The SDP information is normally carried via the SIP payload from one SIP entity to another in order to reach an agreement to establish multimedia sessions. However, SIP accepts complementary information inserted as SIP payload for other applications. In Costa and Tang's research, the SLO information is considered as session data to be carried in the initial call set up and placed in the SIP payload for emergency call services purposes.

SLO information is an XML structure defined in the IETF to represent the user location information. The basis for SLO is to provide a common and extensible container where the user identifiers, security factors, location representation, and other parameters can manage the users' location according to their specifications (Costa and Tang, 2002). The objective of the SLO is to be secured and self-contained to carry the user's information. Furthermore, the SLO has to fulfill user's needs and meet the privacy requirements proposed by the Cellular Telecommunications Industry Association (CTIA) (Altschuk and Coleman, 1994).

2.3.5 SIP for event-trigger LBS

Event-trigger LBS means that the positioning system detects a user's movements and sees the movements as event triggers to inform the user of some useful information he/she might need. For example, to inform potentially affected users about a traffic jam or present special offers in their vicinity. Event-trigger LBS also includes information distribution such as stock information, and advertising. The mobile network is one of the early adopters of the event-trigger feature. It started with the Short Message Service (SMS) and Multimedia Message Service (MMS) by using Wireless Application Protocol (WAP). With the coming of NGN, the convergence of

the old fashioned intelligent network towards an all-IP network makes unified service architecture possible. SIP User Agents (UA) will be available in all Universal Mobile Telecommunications System (UMTS) terminals, supporting bearer establishment directly (Pospischil, Stadler and Miladinovic, 2001).

Three SIP extensional methods are proposed for Presence and Instant Messaging i.e. "SUBSCRIBE", "NOTIFY" and "MESSAGE" (Rosenberg, 2004) (Rosenberg et. al, 2002). Pospischil, Stadler and Miladinovic worked on the use of these SIP extensions to provide event-trigger LBS in 2001. A user can use the SUBSCRIBE method to subscribe to the traffic alerting channel. When a traffic jam occurs, it is pushed to all interested users with the NOTIFY method. The user can also unsubscribe a channel if he/she is not interested anymore. The basics for this service in mobile network are currently being standardized in 3GPP. Packet switched UMTS sessions will be handled via SIP. With the extensions defined in RFC3265, SIP can also be used for messaging and asynchronous notifications. A disadvantage of using SIP is that this protocol is not originally designed for LBS which might affect the Quality of Service (QoS). But the following benefits of using SIP for LBS are evident:

1. Available in all future mobile terminals and in the Internet
2. Flexible content
3. Open Internet standard
4. Decentralized and scalable architecture

2.3.6 SIP LBS in the ubiquitous network

Ubiquity of computing resources does not only facilitate selection of the multiple access network, but also enables users to discover and connect to necessary computing resources while they move from one place to another (Iso, Kurakake and Sugimura, 2003). Recently, SIP has been actively involved in providing communicative service for mobile devices in the ubiquitous network where Internet access is available anytime and anywhere.

Japan NTT lab developed a functional element based on the SIP Presence platform called Presence Engine (PE), which generates Presence information and shares it among mobile devices (Kanamaru and Yoshitsugu, 2004). The Presence information is based on the user's context (age, name, activity, interests, location etc.). This information is collected from each mobile device and converted into Presence information according to standardized PIDF schema (Sugano, Fujimoto, Klyne, Bateman, Carr, and Peterson, 2004). Technologies like GSM/UMTS, Bluetooth and Radio Frequency Identification (RFID) have been integrated into mobile devices to provide local-area and indoor positioning. Network devices can be defined as sensors that detect the location information through currently connected access networks. PE collects location information via sensor interface and modifies the mobile device's Presence information (Kanamaru, and Yoshitsugu 2004). Hence, a user's movement is automatically reflected by the content of Presence information.

2.4 Summary

LBS is a service of great value for both customers and service providers. Nevertheless, LBS privacy is an important factor which involves the potential risk of revealing sensitive information. Location systems could somehow embarrass customers if privacy has not been correctly handled. People could feel uncomfortable and insecure, even tired of using it and this could cause a reduction in customer's appreciation of LBS. Hence, how to put location data under the sole control of users in order to provide privacy is the question which also deserves significant attention. With the coming of NGN, the convergence of the old fashioned intelligent network towards all-IP network makes unified service architecture possible. SIP has been chosen as a key protocol because its flexibility and extensibility. Much research has been done to demonstrate that the SIP could possibly provide a standard way to transfer the spatial location over the Internet in a private and reliable manner for the unified service, which is the goal that this thesis strives to achieve.

CHAPTER 3 Methodology

The purpose of this chapter is to present the research methodology that is used to answer the research questions of this thesis. This chapter starts with two models of the LBS (PUSH and PULL) that were presented and developed by this thesis. In order to answer the research questions, an empirical evaluation methodology is used that is composed of three research methods: laboratory simulation, comparison between two models, and software agent data collection. Laboratory simulation is used to set up an experimental environment, in which experiments can be flexibly performed with minimal running costs. A comparison is made between the PUSH and PULL models in order to provide a suitable evaluation for the proposed strategy. In order to collect and verify the data, a software agent is introduced into the system to observe the whole system while sitting above the operation level.

3.1 "PUSH" and "PULL" concepts

This thesis defines two distinct sets of clients. One set of clients, called the Location Information Notifier (LIN), provides its current location information to be stored and distributed. The other set of clients, called the Location Information Subscriber (LIS), subscribes and receives LIN's location information. In order to accommodate different users' needs (see research aim in Chapter 1), two types of LBS are proposed by this thesis: PUSH and PULL. These two models are based on "PUSH" and "PULL" concepts. The "PUSH" concept means that a LIN is active; the LIN pushes every LIN's location change to the LIS. The "PULL" concept means that a LIS is active; the LIS pulls every LIN's location change from the LIN.

From a user point of view, a PUSH model user (LIS) does not need to query every time when he wants to know where his partner (LIN) is, but only need to send one request to build a connection between them, and all his partner's location changes will be pushed to him through that connection. But, sometimes, it is not necessary to know every location change his partner made; the user might feel irritated by

unwanted location updates. To avoid this situation, the PULL model helps the user to obtain his partner's location information whenever he wants to or at a certain pace decided by himself. Moreover, in the PULL model a constant connection is not needed. For instance, if he decided to check his partner's location once an hour then there is no need for them to be connected for the whole hour.

3.2 The Software Development Method

The software development for this thesis is based on the prototyping method. This method is used to develop a SIP-base LBS prototype to implement the "PUSH" and "PULL" concepts of this thesis. An exploratory prototype process is used to perform rapid development of a system, where an initial prototype is produced and refined through a number of stages towards the final system (Sommerville, 2000). Generally, the steps include: requirements specification, requirements analysis, rapid software writing, software testing, and prototype refining (see Figure 3.1).

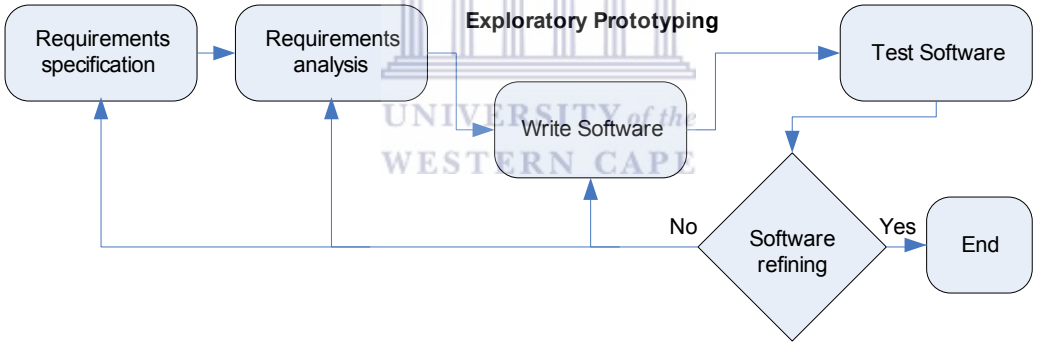


Figure 3.1 The proposed software development method

This exploratory prototyping method includes five main steps. Before the actual programming, Requirements specification and Requirements analysis are two steps needed to be taken in order to provide the whole system objective and divide it into individual function modules. After the software implementation, the software needs to be tested to see if all function modules are functioning as required. If not, the problem needs to be identified for another prototyping cycle, and depending on how severe the problem is, these steps could be taken: re-do the requirements specification, requirements analysis, or directly rewrite the software.

Requirements specification

The main task of this step is to collect the LBS requirements and its related privacy concerns from a designer's point of view. It involves investigation of previous LBS

systems as well as paying attention to the pitfalls of existing research and commercial systems in order to identify the privacy needs of the LBS. Finally, combine all the collected information and provide a requirement specification.

Requirement analysis

This step analyzes the requirements specification and matches them with the SIP architecture and its extensions in order to conceive possible solutions. Additionally, the LBS requirements need to be merged into individual SIP components.

Software writing

The results obtained during the requirements analysis and service definition are followed by an implementation of the SIP Presence Agent (PA), Presence User Agent (PUA) and Instant Messaging Client (IMC). All these entities are required to support LPIDF.

Software testing

After implementation, the software is tested under different scenarios with simulated GSM cell data generated by MPC Map Tool. MPC Map Tool is developed by Ericsson and used in this thesis to create a location database. The location data can be randomly extracted from the database to test the PUSH and PULL models to see if they are functioning as required.

Software refining

The main goal of this step is to refine the software in order to test the software with the required parameters such as location change rate and defined policies. According to the results received after testing the software, adjust original design and re-implement the software until the system is functioning as required.

3.3 The Research Methodology

In order to answer the research questions, an empirical evaluation (Weiderman, Habermann, Borger and Klein, 1987) is used to investigate the feasibility of the "PUSH" and "PULL" concepts by using the SIP. The empirical evaluation is composed of the following steps: preparation of the testing environment, system

design, experimental design, data collection, data visualization and data analysis (see Figure 3.2).

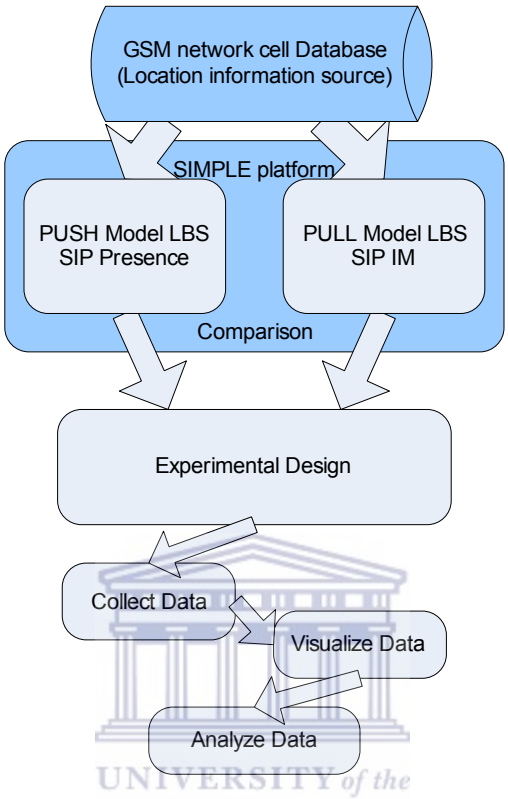


Figure 3.2 Empirical evaluation

The empirical evaluation is chosen to examine the proposed solutions (PUSH and PULL model) in order to answer the research questions. Laboratory simulation is used to create a fictive environment in the lab as a test-bed for the PUSH and PULL models. This thesis enhances the SIP Presence function to the PUSH model and the IM function to the PULL model. These two models use different forms of data transmission. The strategy used in this thesis is by comparing these two models to find a stable way of transferring location information. Hence, three experiments (fast data transfer, large number of data transfers and long term data transfer) are designed and carried out. A software agent is used to collect data on a potentially large and ongoing basis, followed by data visualization and, finally, data analysis in order to provide general recommendations for the use of SIP-based LBS.

The empirical evaluation combines three approaches: laboratory simulation, comparison of PUSH and PULL models, and the software agent observation. These approaches are discussed further below.

3.3.1 Laboratory simulation

Munoz et al. conducted a workplace research of LBS at IMSS general public-health hospital in Ensenada, Mexico (Munoz, Rodriguez, Favela, Martinez-Garcia and Gonzalez, 2003). Due to the complexity and urgency of the hospital environment, many researchers have acknowledged that hospital communication typically involves different locations, work hours, and communication paths (Bossen, 2002; Reddy, and Dourish, 2002). The authors need to deeply understand information management in hospital. Hence, doctors, nurses, laboratory personnel were required as real users to test the system. However, after carefully comparing the real user approach of Munoz with several simulated approaches, laboratory simulation has been used in this thesis for the follow reasons:

- Location information is the only type of information involved in this project and it is environment-independent information
- World wide positioning technology is required. Location information changes beyond a local area level (not only inside a hospital). A user's location might change from one country to another
- Rapid and frequent location information changes are needed for testing purposes that can hardly be achieved through using real users

Therefore, a laboratory simulation approach (see Figure 3.3) was selected to set up an experimental environment. For testing, PUSH and PULL models need nothing more than a user's location data. It is costly and unnecessary to let the real users move swiftly from place to place just to generate new location data. While a location information database can be created and extracted randomly from a simulated location database. As a result, experiments can be conducted which are flexible and without actual running costs. The GSM mobile network is used is this thesis as an underlying positioning system for testing purposes because it possesses one of the widest coverage over the world and cell phones have become the most common mobile devices. It is practical and easy to put into real use. A Cell is the basic unit of a mobile system and is defined as the geographical area where the radio coverage is

given by one base station. When a call is set up, the phone is always connected to the base station belonging to the Cell where the user is located (see chapter 5 for more details).

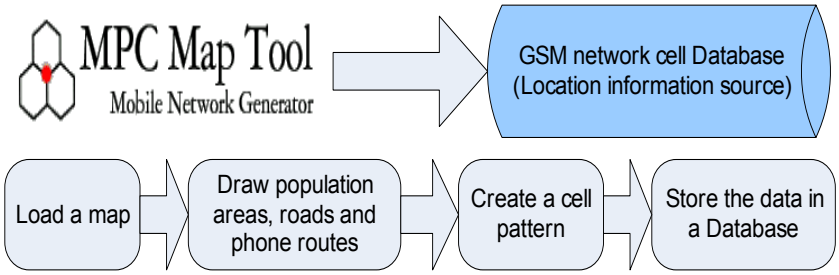


Figure 3.3 The process of creating a GSM network cell database

The GSM mobile network possesses one of the widest coverages over the world and cell phones have become the most common mobile devices. It is used is this thesis as an underlying positioning system for testing reasons. The Ericsson MPC map tool is used to create a GSM network cell database with four main steps: load a map, draw population areas, roads and phone routes, create a cell pattern, and store the data in a database.

3.3.2 Comparison of PUSH and PULL models

There are two types of LBS in this thesis: PUSH and PULL model SIP-based LBS. Two main components of the LBS are LIN and LIS. In the PUSH model a LIN pushes every location change to a LIS as soon as it is available. In the PULL model a LIS performs a periodic query to obtain LIN's location information at a pace decided by the LIS. From a technical point of view, two models are employed to implement an LBS application by using the SIMPLE platform. The PUSH model builds on the SIP Presence extension. The LIS is equivalent to the SIP Presence entity "Watcher" and the LIN is equivalent to the "Presentity". The PULL model builds on the SIP IM extension; both the LIS and LIN are seen as SIP IM clients because they are equal peers. A LIS sends a message to request a LIN's location information. A LIN replies with a message containing its location information to inform a LIS.

Push model

This model requires a constant connection between LIS and LIN. This is done by the LIS sending out first SIP SUBSCRIBE message to LIN, which initiates a Presence session. Once the Presence session is established, the LIN updates its location changes automatically to LIS through sending SIP requests (see Figure 3.4). This

model uses stateful transmission. A single SIP request and all responses to that request are seen as a transaction. The life cycle of each transaction is managed by a Finite State Machine (FSM) which consists of four states: Trying, Proceeding, Completed and Terminated (Rosenberg et. al, 2002). For each transaction, the FSM creates a state associated with it and keeps in the memory for the duration of the transaction.

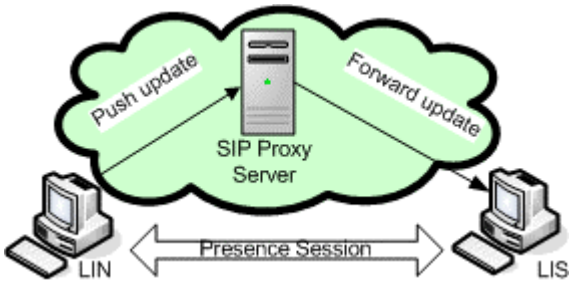


Figure 3.4 The PUSH model

In this model, the LIN is active and the LIS is passive. The SIP Presence is applied to build up a Presence session between a LIN and a LIS. A LIS first sends out a SUBSCRIBE request to a LIN to build up the session. Once the session is built, the LIN updates its location changes through NOTIFY requests. The advantage of this model is that the LIS will not miss any location change of LIN. The LIN delivers a location change to a LIS as soon as it is available without being queried by the LIS. But the disadvantage of this model is that the LIS is forced to receive every change which LIS might not want to receive.

Pull model

The SIP IM is applied to the PULL model where a constant connection between a LIN and a LIS is not required. The LIS performs periodic checks for location information from the LIN at a pace decided by the user. The connection is only needed when the LIS tries to pull information from the LIN. A LIS sends a message to request a LIN's location information. The LIN replies with a message containing its location information to inform the LIS (see Figure 3.5). This model uses stateless transmission which is simpler than stateful transmission. It does not have any notion of the transaction and, therefore, it does not need to maintain the state of a transaction. The SIP entities directly forward SIP messages to the transport layer without creating any transaction.

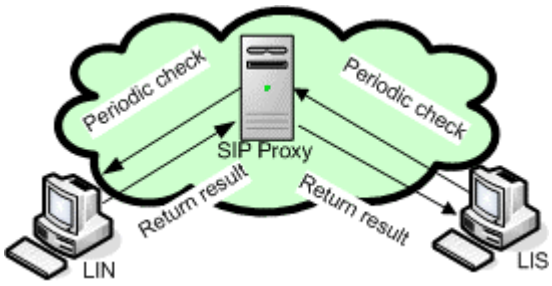


Figure 3.5 The PULL model

In this model the LIS is active and the LIN is passive. The SIP IM is used for this model where a constant connection is not required. A LIS performs periodic checks to obtain a LIN's location information at a certain pace decided by the LIS. The LIS sends a MESSAGE request to query the LIN's location information. The LIN replies to the MESSAGE request to inform the LIS of its location information. The advantage of this model is that the LIS can get LIN's location information at a certain pace decided by the LIS, and this model does not need to maintain a constant connection between the LIS and the LIN. The disadvantage of this model is that the LIS needs to query every time for the LIN's location information. In addition, the LIS might fetch the same location information or miss some location changes.

Comparison approach

Shekhar and Yoo address a technical operation of the LBS (nearest neighbour query) through four alternative methods. An experimental framework was used to compare them in order to examine the behaviour of the solutions in terms of three parameters affecting the performance (Shekhar and Yoo, 2003). The comparison approach shows that it is adequate to identify both the advantages and disadvantages of each solution through the experimental results. It also provides suitable evaluation for the proposed strategy.

With respect to the thesis, PUSH and PULL are two alternative solutions for the SIP-based LBS. They are proposed in this thesis not only for accommodating different users' needs, but also because they use different data transfer methods (stateful and stateless transmission) in order to identify the advantages and disadvantages of each model (see research aim in Chapter 1). A comparison is therefore made between the SIP PUSH and PULL models to examine the robustness of each model. The anticipated result of this method is an indication of the areas where the SIP PUSH

model is preferred to the SIP PULL model in line with transfer location information and vice-versa.

3.3.3 Software agents data collection

Hilbert and Redmiles described how software agents could be used to collect information regarding application usage, on a potentially large and ongoing basis (Hilbert and Redmiles, 1998). Repeated human work processes can be tiresome and sometimes inevitably prone to mistakes. Some experiments in this project were required to be repeated over a hundred times under the same conditions. With people as subjects, this could be difficult due to the time and labour involved in collecting data, lack of scalable tools for automatic data collection, and the lack of proper incentives for high-quality voluntary data collection on the part of users. Hence, an additional software agent is implemented to collect experimental results automatically.

This software agent is used to observe the whole system while sitting above the operation level. The purpose of this entity is to collect generated location information from the LIN side and received location information from the LIS side. Other data the software agent needs to collect is the policy set by the LIN (see Figure 3.6). The policy is a system parameter dynamically set by a user through real-time interaction. This parameter is used to control the location information disclosure for both PUSH and PULL models (see chapter 4 from more detail). The software agent captures all data that has been sent or received from both the LIN and LIS. Because this entity is not part of the system except to watch the system from a higher level, it has the nick name “bird’s eye”.

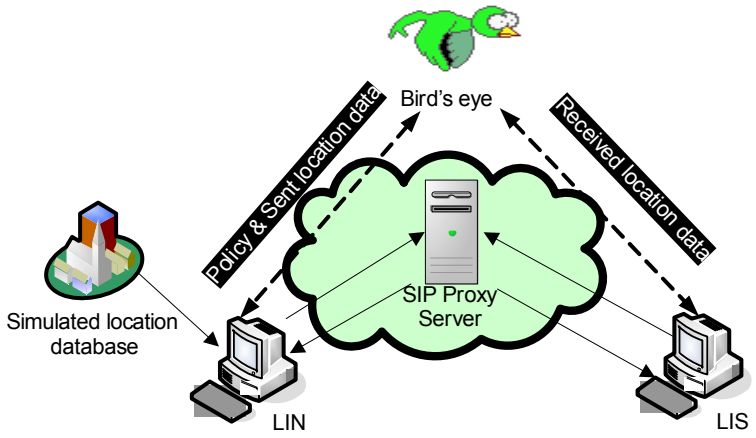


Figure 3.6 The software agent data collection

The bird's eye is a software agent used in this thesis to collect three types of data automatically: Sent location information from the LIN, received location information from the LIS and the policy set by the user from the LIN. It is called bird's eye because it is not part of system except to watch the system from a higher level.

3.4 Summary

This chapter presented the research methodology of this thesis in order to evaluate the proposed solutions (PUSH and PULL). This research methodology includes six main steps. The first step is to prepare the testing environment. In order to obtain location data, the MPC map tool is used to create a required GSM network cell database. Location data is extracted from the database as input for both PUSH and PULL models. This step will be discussed further in Chapter 5. The second step is system design. According to the proposed strategy, the PUSH model is implemented on the basis of the SIP Presence extension and the PULL model on the basis of the SIP IM extension. This step will be discussed further in Chapter 4. The third step is experimental design. The PULL model and PUSH model is expected to provide QoS. The performance of each model is tested under the situations of fast location data transfer, large number of data transfers, and long-term data transfer. This step is handled in Chapter 5. The fourth step is data collection. Input location data will change over a hundred times under different values of the two parameters. A software agent is used to collect data which includes input location data, output location data and applied policy. This step will be discussed in Chapter 5. The fifth step is data

visualization. Collected data from the software agent is visualized in tables and graphs. The input and output location data under the same test condition is presented in the same graph for comparison purposes. This step will be demonstrated in Chapter 6. The last step is data analysis. After data visualization, data analysis is used to identify both the advantages and the disadvantages of each model. This is expected to provide general recommendations for the use of SIP-based LBS. This step will be presented in Chapter 6.

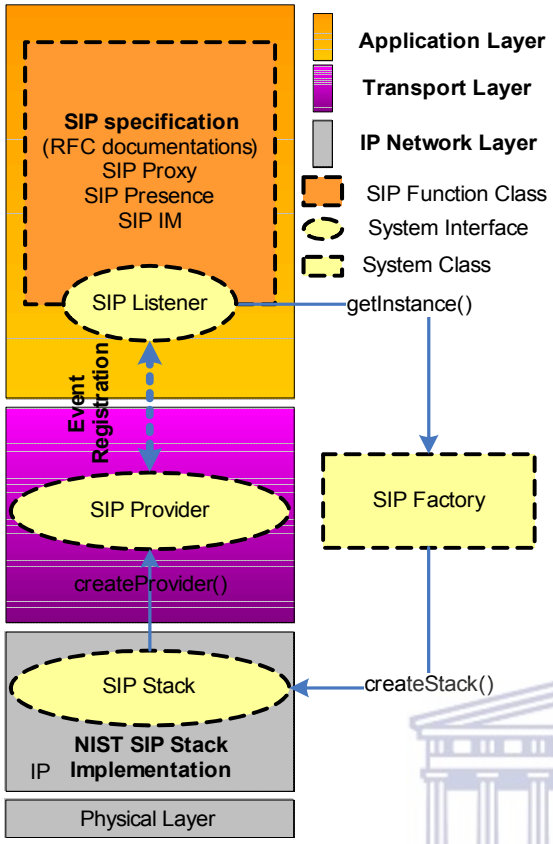
CHAPTER 4 System design

The purpose of this chapter is to present the implementation of the PUSH and PULL models and the fundamental architecture on which they are based. The name of the system implemented in this thesis is SIP Instant Messaging and Presence LBS developed by the author Yanhao (SIMPLY). The SIMPLY system is built on the open source project JAIN-SIP-PRESENCE-PROXY provided by NIST[1]. SIMPLY enhances the original Presence function to provide a PUSH model location-based service and modifies the IM function to provide a PULL model location-based service.

4.1 System architecture from a network point of view

Like any form of conversation in which two people speak and hear in order to communicate, network components must also be able to receive and send signals in order to communicate with one another. SIP deals with messages. SIP messages can be categorized into two basic types namely Request and Response. Core SIP Requests include INVITE, ACK, OPTIONS, BYE, CANCEL and REGISTER. Requests are sent out for certain communication purposes such as initiating a call or terminating a call. Responses comprise six classes: 1xx, 2xx, 3xx, 4xx, 5xx, and 6xx (1xx means from 100 to 199). A Response is used to inform the SIP agent about its Request such as “the Request is accepted successfully” or “the Request cannot be fulfilled”. In order to communicate, any SIP agent should be capable of sending Requests to its desired SIP partner as well as receiving Responses from its partner.

[1] The source code of JAIN-SIP-PRESENCE-PROXY is available at http://www-x.antd.nist.gov/proj/iptel/



This is a standard 4-layer IP network structure that every SIP entity in this system must follow. It is designed to use four fundamental interfaces/classes (SIPListener, SIPFactory, SIPStack and SIPProvider) to send and receive SIP messages to and from the network. It is a Listener/Provider event model. There is a direct relationship between the event-provider and event-consumer; event-consumer must register with the event-provider. The SIPProvider is the event-provider that receives messages from the SIPStack and passes them to the SIPListener as events. The SIPListener represents the event-consumer, which listens for incoming events that it may respond to accordingly. The SIPStack is created from a single class SIPFactory. The SIPStack is associated with a physical IP address which receives messages from the network as well as setting the Route-Path that determines how to route messages.

Figure 4.1 System object structure

Figure 4.1 shows the route that every SIP component in SIMPLY must follow in order to send and receive SIP messages. This four-layer architecture is inherited from the JAIN SIP object architecture (http://www-x.antd.nist.gov/proj/iptel/tutorial/JAIN-SIP-Tutorialv2.pdf), which is a Listener/Provider event model. The system is designed to implement this event model using four fundamental system interfaces/classes: SIPListener, SIPFactory, SIPStack and SIPProvider. There are 5 steps in implementing any SIP agent (using the SIP IM client as an example):

1. Implement a SIPListener interface to a class to fulfill a SIP IM client function
2. Obtain SIPFactory by using the getInstance() method inside the SIP IM client class
3. Create a SIPStack from the obtained SIPFactory by using the createStack() method
4. Create a SIPProvider from the created SIPStack by using the createProvider() method
5. Register the SIP IM client to the created SIPProvider by using the addSipListener() method

4.2 Implementation of SIMPLEUA

The core entity of SIMPLY is called SIMPLE User Agent (SIMPLEUA). It is a Back-to-Back User Agent (B2BUA) that processes received Requests as a user agent server (UAS) and generates Requests as a user agent client (UAC). SIMPLEUA is built on top of JAIN SIP stack which implemented a transaction layer under the application layer to deal with stateful transmission of SIP Requests and Responses for PUSH model. The transaction layer handles application-layer retransmissions, matching of responses to requests, and application-layer timeouts. Any task that a UAC/UAS accomplishes takes place using a series of transactions. Both Requests and Responses are related to transactions (see Figure 4.2). However, the transaction layer is not in the notion of the PULL model because the PULL model does not take care of transactions, which makes it simpler than the PUSH model. With respect to the PULL model, SIMPLEUA behaves as a stateless UAS/UAC.

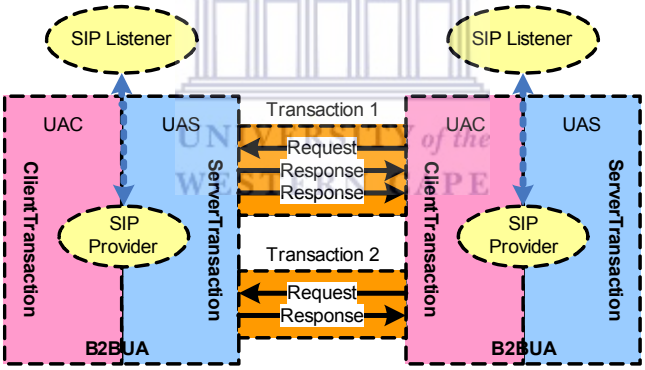


Figure 4.2 Transaction layer

The transaction layer deals with stateful transmission of SIP Requests and Responses. A single Request and any number of Responses to that request are seen as a transaction. There are two types of transactions: client-transactions and server-transactions. A client-transaction is used by a UAC application to send a Request message to a UAS application as well as to match Responses to that Request. A server-transaction is used by a UAS application to handle an incoming a Request message on a specific transaction and send back Response messages in response to that Request. The SIPProvider is used to fire Requests to the SipListener as well deliver Responses to the transport layer.

SIP is a text protocol that is composed of a set of rules and standard procedures. Developers implement the protocol into software entities by transferring the text rules

into programming classes, which can be done in any programming language. SIMPLEUA includes Presence and Instant Messaging function defined by the SIP specifications and is written in JAVA. It is the implementation of the SIPListener interface (see Figure 4.1), which sits on the top of the IP 4-layer structure and processes SIP messages according to the recommendation of SIP specifications: RFC3261 (Rosenberg et. al, 2002), RFC3265 (Roach, 2002), RFC3428 (Rosenberg et. al, 2002) and RFC3856 (Rosenberg, 2004). It has two main process methods: processRequest() and processResponse(). These two methods are used to process a Request/Response received from a SIPProvider upon which the SIPListener is registered. Some related main steps of SIMPLEUA are selected and put into Chapin diagrams below (see Table 4.1 and Table 4.2).

Table 4.1 The processRequest method

<table>
<tr><td colspan="4">ProcessRequest (RequestEvent)</td></tr>
<tr><td colspan="4">ServerTransaction = RequestEvent.getServerTransaction()</td></tr>
<tr><td colspan="4">IF ServerTransaction = null
#T #F</td></tr>
<tr><td colspan="2"></td><td colspan="2">ServerTransaction =
SIPProvider.getNewServerTransaction(Request)</td></tr>
<tr><td colspan="4">IF RequestMethod = “MESSAGE”
#T #F</td></tr>
<tr><td rowspan="3">MessageProcessing
(Request)</td><td colspan="3">IF RequestMethod = “SUBSCRIBE”
#T #F</td></tr>
<tr><td rowspan="2">SubscribeProcessing
(Request,
ServerTransaction)</td><td colspan="2">IF RequestMethod = “NOTIFY”
#T #F</td></tr>
<tr><td>NotifyProcessing
(Request,
ServerTransaction)</td><td>ServerTransaction
(Response.NOT_AlLLOWED)</td></tr>
</table>

In this method, SIMPLEUA acts as a UAS. It deals with SUBSCRIBE and NOTIFY Requests for Presence function and MESSAGE for IM function using the following steps:

1) *Obtain server-transaction from SIPProvider and if server-transaction is null, it generates a new ServerTransaction to handle the incoming Request.*
2) *Find out the Request type and hand it over to correct class for further processing*

Table 4.2 The processRequest method

<table>
<tr><td colspan="4">ProcessResponse (ResponseEvent)</td></tr>
<tr><td colspan="4">ClientTransaction = ResponseEvent.getClientTransaction()</td></tr>
<tr><td colspan="4">IF ResponseCode = “200”||”202”
#T #F</td></tr>
<tr><td>IF ResponseMethod = “MESSAGE”
MessageProcessOK (Response)</td><td colspan="3">IF ResponseCode = “NOT_FOUND”||”TEMPERAILY_UNAVALIABLE”
#T #F</td></tr>
<tr><td>IF ResponseMethod = “SUBSCRIBE”
MessageProcessOK (Response, ClientTransaction)</td><td>IF ResponseMethod = “MESSAGE”
The Message can not be delivered</td><td colspan="2">IF ResponseCode = “DECLINE”||”FORBIDDEN”
#T #F</td></tr>
<tr><td>IF ResponseMethod = “NOTIFY”
MessageProcessOK (Response, ClientTransaction)</td><td>IF ResponseMethod = “SUBSCRIBE”
The Presentity can not be found</td><td>The Request has been rejected</td><td>IF ResponseCode = “PROXY_AUTHENTICATION_REQUIRED”||”UNAUTHORIZED”
Start the authentication process
Add Proxy-authorization header
Resend the Request</td></tr>
</table>

In this method, SIMPLEUA acts as a UAC. It deals with a Response using the following steps:

1) *Obtain client-transaction from SIPProvider and find out the Response type*
2) *If it is a2xx response, it hands it over to correct classes for further processing*
3) *If it is a 4xx Response, it reports that the desired partner cannot be found*
4) *If it is a 6xx Response, it reports that the Request has been rejected*
5) *If SIP proxy needs authentication, it performs authentication and resends the Request*

4.3 LPIDF

SIP normally carries a SDP packet describing an audio or video session, indicating end terminal capabilities. However, SDP can be replaced by other contents as payload for other purposes in extensions of SIP. SIP Presence services developed PIDF to transfer Presence information to express the users’ capability and willingness to communicate, and the notion of Presence expands far beyond indicating user status as “Online” or “Offline” (Wu, Radovanovic and Tucker, 2005). Figure 4.3 shows an example of a SIP message generated by a SIP Presence agent. The purpose of this project is to find a way to transfer location information using SIP in order to explore the possibility of expanding the SIP’s realm from IP telephony service to LBS. A good solution is to see the SIP message as a GEOPRIV LO and replace the SDP packet with users’ location information. This thesis defines a basic XML schema to contain minimum location information. It is called Location-enhanced PIDF (LPIDF),

an extension of PIDF, and used as a data format for containing geographic information. The basic function of LPIDF is to provide an extensible XML container for location information.

```
"NOTIFY sip:172.16.39.52:5070;transport=udp SIP/2.0
Call-ID: nist-sip-im-subscribe-callId1
CSeq: 1 NOTIFY
To: <sip:wilsonwatcher@nist.gov>;tag=8947
From: <sip:wilsonpresentity@nist.gov>;tag=9948
Max-Forwards: 70
Via: SIP/2.0/UDP
172.16.39.83:4000;branch=z9hG4bKdec36e2473cdab0fad3c36b960a425e
5
Event:  presence
Subscription-State: active;expires=3599
Content-Type: application/pidf+xml
Content-Length: 229

<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n
<presence xmlns=\"urn:ietf:params:cpim-presence:\"
entity=\"sip:wilson@nist.gov\" >\n
<tuple  id=\"NistSipIM_5762\" >\n
<status>\n
<basic>open</basic>\n
</status>\n
<note>online</note>\n
</tuple>\n
</presence>"
```


Figure 4.3 An example of a SIP message

This is an example of a SIP NOTIFY request sent from wilsonpresentity@nist.gov to wilsonwatcher@nist.gov. A SIP message consists of a start-line, one or more header fields, an empty line indicating the end of the header fields, and an optional message-body. Start-line indicates the type of messages, transport protocol and SIP version. Common Header fields include: "Call-ID", "Cseq", "To", "From", "Via"and "Contact". The definitions of these headers can be found in RFC3261.The Presence information carried in this SIP request indicate that wilsonpresentity@nist.gov is "online". LPIDF is defined in this thesis and carried by SIP messages to contain location information.

4.3.1 LPIDF data structure

There are two common ways to express a location address; either through geospatial coordinates or by the so-called civic addresses. Geospatial coordinates indicate longitude and latitude, while civic addresses indicate a street address. Geospatial coordinates and civic address are combined to identify the location of an object in this

project. Since each country has different administrative hierarchies, this project adopts a simple hierarchical notation that is then employed for each country. This assumes that five levels are sufficient for sub-national divisions: "country", "province", "city", "street", and "room" (see Table 4.3). The aforementioned five levels indicate in which respective region a user is (for example, the "country" level indicates in which country a user is). In order to provide privacy, these five levels of location information are controlled by a system parameter called policy (see research aim in chapter 1). The policy has five values: "countryLevel", "provinceLevel", "cityLevel", "streetLevel", and "roomLevel". These five values indicate up to which respective region a user is prepared to reveal. The policy is dynamically set by a user through a SIP real-time authorization process when a LIS requests the LIN user's location information.

Table 4.3 Five defined LO levels

Defined LO level	Equal level	Type	Example
Country	Country, Autonomous region	Civic	SA
Province	Province, State, Prefecture, Region	Civic	Western Cape
City	City, township, District	Civic	Cape Town
Street	The coverage of a GSM cell	Geospatial coordinates	165090_210 /E182815&S331954/CircleSector/3000/210
Room	The coverage of CGI –TA detecting area	Geospatial coordinates	E182733&S331954 2725716932/2300/165090_210/1099/

Civic format is used to express "Country", "Province" and "City" level addresses because these addresses are accessible through the Internet and easy to collect. But, more detailed addresses information like street and room names are not always accessible, and sometimes change to different names. Therefore, geospatial coordinates are used to represent "Street" and "Room" level addresses. MPC is used to generate the GSM network cell pattern, which is efficient and isolated to the name of physical constructions. The coverage of a GSM cell is defined as "street" level address while the coverage of CGI-TA detecting area is defined as the "room" level address. These two levels are further explained in Chapter 5.

4.3.2 LPIDF schema

XML enables developers to create their own markup languages for their own projects. These languages can be shared with individuals working on similar projects through a Name Space (NS) all over the world. Figure 4.4 shows the LPIDF schema with NS =" http://www.cs.uwc.ac.za/~wwu/PresLoc", which provides the meaning and usage of tag <PreLoc> and its sub-tags. A SIP LBS application retrieves and manipulates LPIDF data to accomplish location-aware tasks.

```
<?xml version="1.0" encoding="UTF-8"?>
  <xsd:schema
   targetNamespace="http://www.cs.uwc.ac.za/~wwu/PresLoc"
   xmlns:tns="http://www.cs.uwc.ac.za/~wwu/PresLoc"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xsd:complexType name="PreLoc">
     <xsd:sequence>
       <xsd:element name="Country" type="xsd:string"
         minOccurs="0" maxOccurs="1"/>
       <xsd:element name="Province" type="xsd:string"
         minOccurs="0" maxOccurs="1"/>
       <xsd:element name="Region" type="xsd:string"
         minOccurs="0" maxOccurs="1"/>
       <xsd:element name="city" type="xsd:string"
         minOccurs="0" maxOccurs="1"/>
       <xsd:element name="Street" type="xsd:string"
         minOccurs="0" maxOccurs="1"/>
       <xsd:element name="Room" type="xsd:string"
         minOccurs="0" maxOccurs="1"/>
       <xsd:element name="PostCode" type="xsd:string"
         minOccurs="0" maxOccurs="1"/>
       <xsd:element name="Additional" type="xsd:string"
         minOccurs="0" maxOccurs="1"/>
       <xsd:any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded"/>
     </xsd:sequence>
    </xsd:complexType>
  </xsd:schema>
```


Figure 4.4 LPIDF schema

The basic unit of storage in an LPIDF document is the tuple. LPIDF must contain an umbrella tag <PreLoc> which might contain five sub-tags: <Country>, <Province>, <City>, <Street> and <Room>. These five tags are not allowed to appear more than once in a single LPIDF tuple.

4.3.3 LPIDF Parser

The Simple API for XML (SAX) (http://www.saxproject.org) is used to implement the LPIDF parser to parse an LPIDF document. It is a free API for both commercial and non-commercial use. This API provides a standard architecture and has implemented some optional helper classes for XML parser developers. Software developers only need to implement a few interfaces to develop a SAX-conformant XML parser for their own projects. The LPIDF parser is a SAX-conformant XML parser developed in this thesis. It comprises five essential interfaces defined by SAX: XMLReader ContentHandler, DTDHandler, EntityResolver, and ErrorHandler. It is also based on the Listener/Provider event model. The XMLReader is the event-provider that parses an LPIDF document and generates XML events. ContentHandler, DTDHandler, EntityResolver, and ErrorHandler are event-consumers, which are used to handle different events generated by XMLReader (see Figure 4.5).

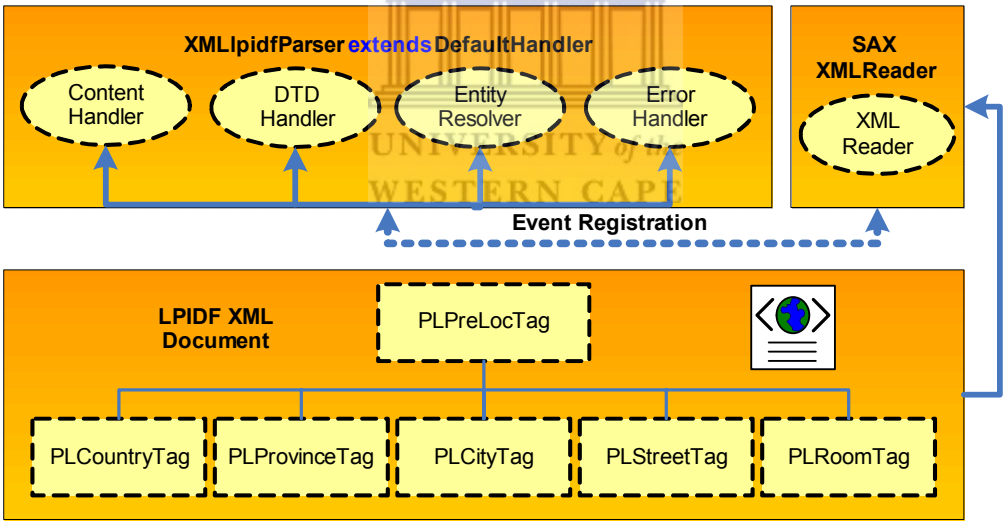


Figure 4.5 LPIDF architecture

The LPIDF parser is based on the Listener/Provider event model. XMLReader is the event-provider. ContentHandler, DTDHandler, EntityHandler, and ErrorHandler are event-consumers. These four interfaces are implemented in one class DefaultHandler. The Setxxx() method is used to register event-consumers to the event-provider (e.g. XMLReader registers a ContentHandler using setContentHandler(ContentHandler) method). LPIDF extends PIDF with six main classes: PLPreLocTag, PLCountryTag, PLProvinceTag, PLCityTag, PLStreetTag and PLRoomTag. XMLReader parses an LPIDF document as input and generates output events to be handed over to event-consumers.

Some related main steps of the LPIDF parser are selected and put into a Chapin diagram shown below in Table 4.4.

Table 4.4 XMLlpidfParser Class

<table>
<tr><td colspan="4">ParseLpidfString (String xmllpidfbody)</td></tr>
<tr><td colspan="4">SaxParser.parse (InputSource)</td></tr>
<tr><td>StartDoc ument ()</td><td>StartElement ()</td><td>EndElement ()</td><td>EndDoc ument ()</td></tr>
<tr><td rowspan="14">Start to parse XML LPIDF string</td><td>IF element = “PreLoc”
#T #F</td><td>IF element = “Country”
#T #F</td><td rowspan="14">XML LPIDF string parsed successfull y</td></tr>
<tr><td>Create a new PLPreLocTag</td><td>PLCountryTag.setCountry (String element)
PLPreLocTag.setCountryTag (CountryTag)</td></tr>
<tr><td>IF element = “Country”
#T #F</td><td>IF element = “Province”
#T #F</td></tr>
<tr><td>Create a new PLCountryTag</td><td>PLProvinceTag.setProvince (String element)
PLPreLocTag.setProvinceTag (ProvicneTag)</td></tr>
<tr><td>IF element = “Province”
#T #F</td><td>IF element = “City”
#T #F</td></tr>
<tr><td>Create a new PLProvinceTag</td><td>PLCityTag.setCity (String element)
PLPreLocTag.setCityTag (CityTag)</td></tr>
<tr><td>IF element = “City”
#T #F</td><td>IF element = “Street”
#T #F</td></tr>
<tr><td>Create a new PLCityTag</td><td>PLStreetTag.setStreet (String element)
PLPreLocTag.setStreetTag (StreetTag)</td></tr>
<tr><td>IF element = “Street”
#T #F</td><td>IF element = “Room”
#T #F</td></tr>
<tr><td>Create a new PLStreetTag</td><td>PLRoomTag.setRoom (String element)
PLPreLocTag.setRoomTag (RoomTag)</td></tr>
<tr><td>IF element = “Room”
#T #F</td><td rowspan="2"></td></tr>
<tr><td>Create a new PLRoomTag</td></tr>
</table>

XMLlpidfParser has four main methods: startDocument(), endDocument(), startElement() and endElement(). StartDocument() and endDocument() are used to indicate the beginning and the end of parsing a LPIDF document. If startElement() finds a matched LPIDF element, it will create a new correlated tag class. EndElement () is used to store location information into sub-tag class as well as store these sub-tags to the umbrella-tag PLPreLocTag.

4.4 SIMPLY PUSH and PULL

Two LBS models are proposed in this thesis: the PUSH and PULL location service. The SIMPLEUA implements both models based on the SIMPLE platform: Presence for the PUSH model and IM for the PULL model. In order to prevent unnecessary information from being revealed to a third party, a Peer-to-Peer (P2P) architecture is applied to the system. Location information is captured, stored, and processed on a local B2BUA. In order to let users have control over their own location information disclosure, the SIP authorization mechanism is applied to the system for both PUSH and PULL models. When a B2BUA sends a SUBSCRIBE request to another B2BUA, it will trigger a real-time authorization process with a user. On a P2P level, B2BUAs authorize one another personally with five predefined values of the policy: “countryLevel”, “provinceLevel”, “cityLevel”, “streetLevel” and “roomLevel”. Basically, policy is a system parameter used to restrict location information to show unprivileged users less information, but in an emergency situation, SIP PUBLISH request is used to distribute location information without policy constraint.

4.4.1 The PUSH model implementation

In the PUSH model, the Presentity is active and the Watcher is passive. Because Watcher is passive, it does not need to modify its standard behavior, except the capability of interpreting LPIDF by using the XMLlpidfParser class needs to be enhanced. The standard procedure of a Watcher can be found in RFC3265 (Roach, 2002) Subscriber behavior section. Most of the modifications are made on the Presentity side, which includes real-time authorization and policy decision-making. Hence, the focus of the discussion of the PUSH model is placed on the Presentity side.

When the SIMPLEUA acts as a Presentity, it needs to process incoming SUBSCRIBE Requests and sends out NOTIFY Requests when the Presentity’s location has changed. Figure 4.6 shows the main workflow when SIMPLEUA acts as Presentity. The procedure of a Presentity has the following three phases: authorization, location change, and notification.

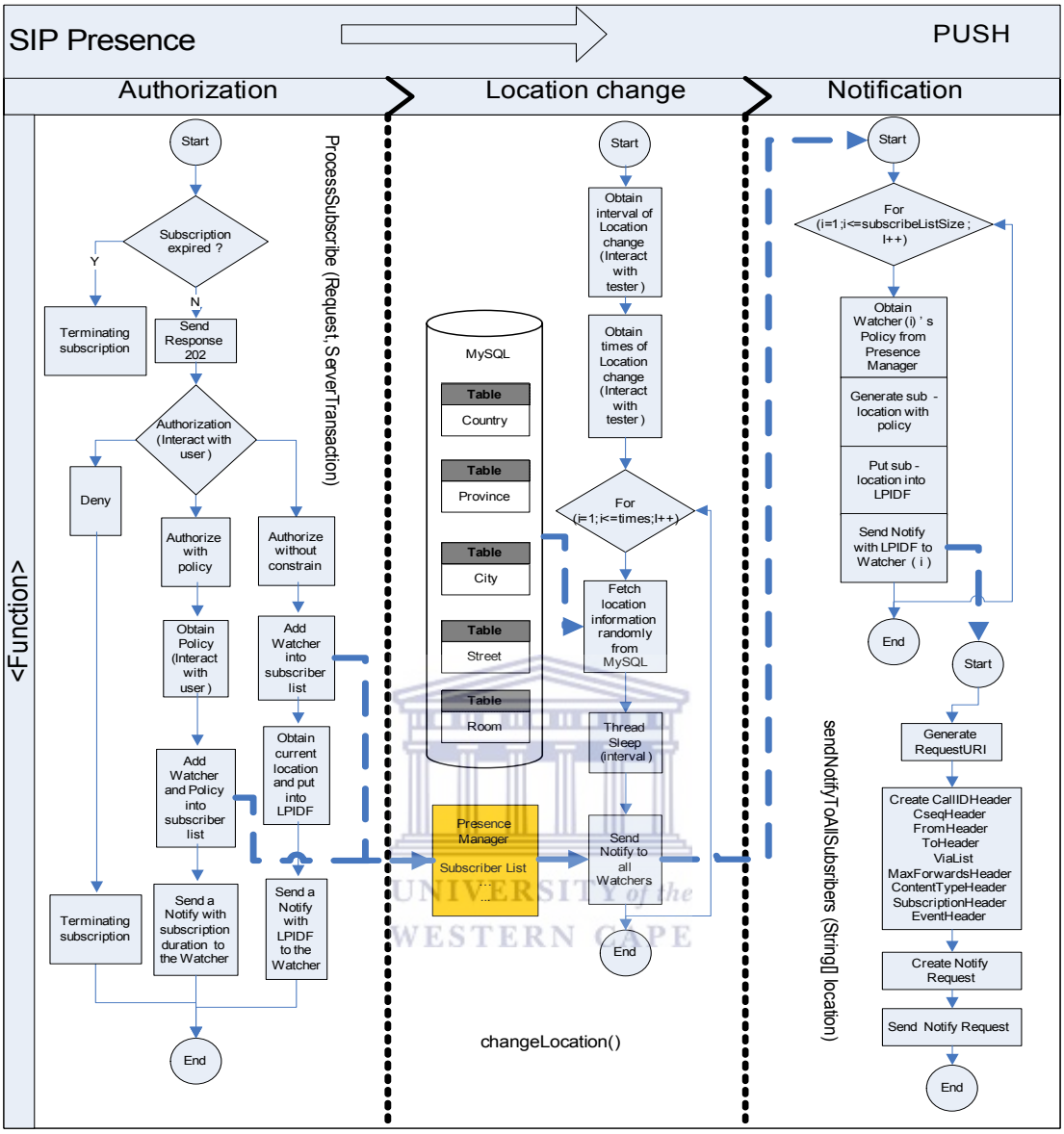


Figure 4.6 The PUSH model workflow

1) ***Authorization:*** *SIP authorization mechanism is used to provide privacy protection for SIMPLY users. The authorization must be approved by the LIN user before any LIS could get the LIN user's location information.*
2) ***Location change:*** *After the approval of authorization, the LIS name and its policy value are stored in the Presence Manager (PM) as part of the subscription information. Once the subscription is built into the PM subscription list, the LIN will notify its new location information to the LIS. The LIN's new location information is acquired from a simulated location database.*
3) ***Notification:*** *Once new location data is acquired from the simulated database, the Presentity will search the subscription list to get all the LIS names and their policy values, and send the location information to each LIS according to its policy value.*

Authorization

The SIP authorization mechanism is used to provide privacy protection for SIMPLY users. This is achieved through setting up the policy value, thus providing a means for users to dynamically decide who is going to get which level of their location information. When a Watcher sends out a SUBSCRIBE request to a Presentity, the ProcessSubscribe (Request, ServerTransaction) method is used by the Presentity to process the incoming request. First, it needs to check that the duration in the "Expires" header has not expired. Secondly, the Presentity will return a "202 Accepted" response instead of immediately creating the subscription, to indicate that the SUBSCRIBE request has been received and understood. Thirdly, a subsequent NOTIFY request is sent out to inform the Watcher that the Presentity is waiting for the user's input for authorization. It must contain a "Subscription-State" header with a value of "pending" to indicate that the subscription has been received, but that policy information is insufficient to accept or deny the subscription at this time. Its primary purpose is to serve as a reliability mechanism and hence does not generally contain any useful information beyond the subscription duration. Once the authorization is approved, a subscription will be created; it stores the Subscriber's name and the policy value as part of the subscription information in the Presence Manager (PM).

Location change

In the PUSH model a Presentity user's movements are seen as events which will trigger location notifications. New location information is randomly fetched from a location database to simulate the users' movements. First, a user needs to enter the time interval and number of location changes, and then the location information is extracted from five tables in the database according to the values set by the user. Secondly, the Presentity sends out NOTIFY Requests of location changes to Watchers according to the previously built subscriber list.

Notification

Presentity uses the sendNotifyToAllSubscribers(String[] location) method to generate and send NOTIFY requests to all Watchers. When a location change occurs, the Presentity should immediately construct and send NOTIFY requests according to the

subscription list. NOTIFY requests must contain a “Subscription-State” header with a value of “active” to indicate that the subscription has been accepted and authorized. A NOTIFY is sent for a particular Watcher, and the contents of the body within that notification, are subject to the authorization policy that governs the subscription. The NOTIFY request will contain a body indicating the location of the Presentity. The body of the NOTIFY must be sent using the type “application/lpidf+xml”. A valid SIP request formulated by SIMPLEUA must, at a minimum, contain the following header fields: To, From, CSeq, Call-ID, Max-Forwards, and Via; all of these header fields are mandatory in all SIP Requests.

4.4.2 The PULL model implementation

The SIP IM extension is used in the PULL model. SIP IM has only one type of Request, which is the MESSAGE request. The MESSAGE request is designed to be dedicated to carrying the actual communication data between participants. This model is actually not a pure IM model. It is developed on the basis of the PUSH model. The SIP Presence extension is still used to provide privacy protection for SIMPLY users. The PULL model inherits the real-time authorization mechanism from the PUSH model. SUBSCRIBE and NOTIFY Requests are used to initiate the partnership between the IM Subscriber and IM Notifier, then it hands the actual data communication over to MESSAGE Request. At the PULL model, a LIN is called an IM Notifier, which provides its current location information; while a LIS is called an IM Subscriber, which receives the LIN’s location information. The IM Subscriber performs periodic checks on the IM Notifier in order to get its location information in a certain pace decided by the user. Figure 4.7 shows the main workflow of the PULL model with the following four phases: authorization, location change, location request, and location response.

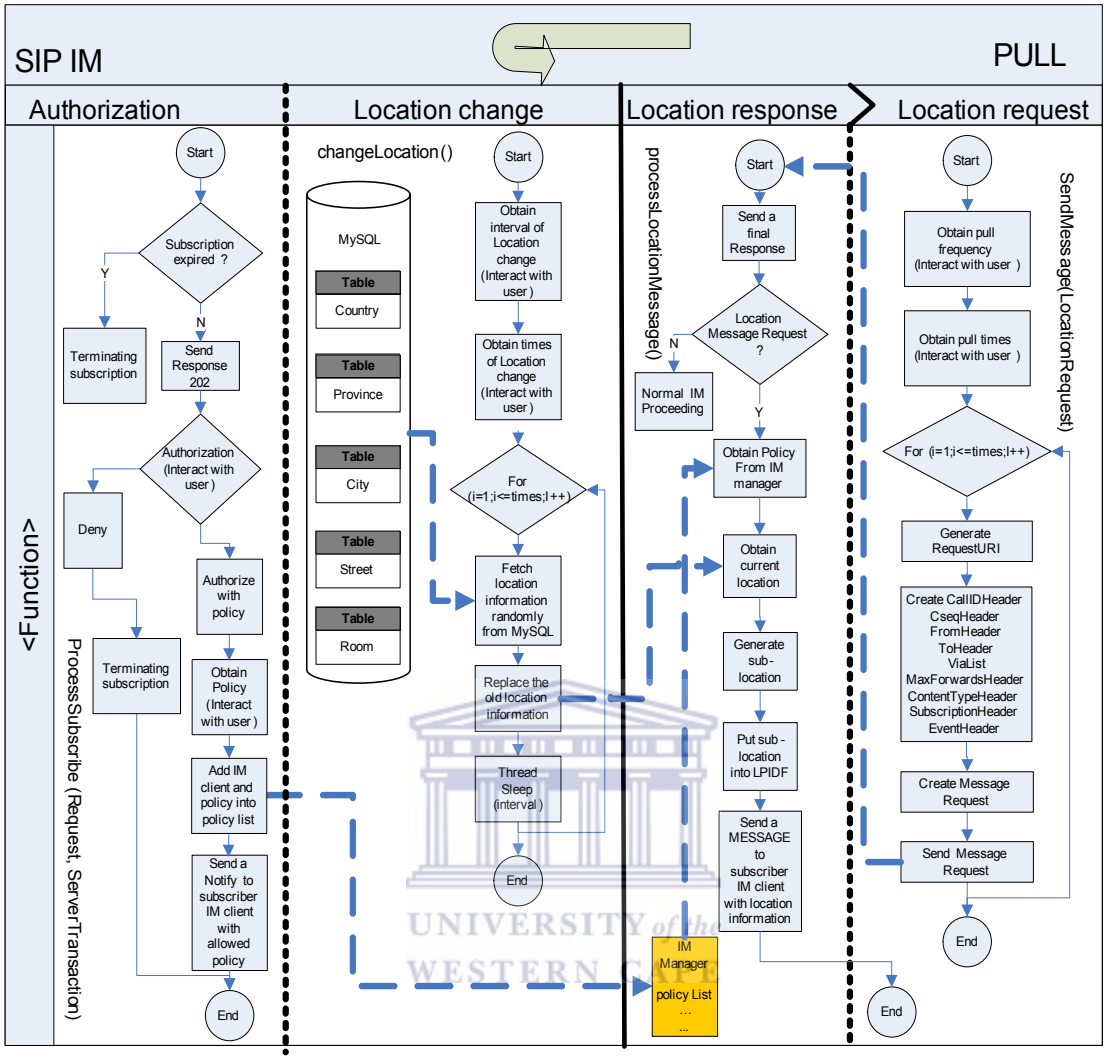


Figure 4.7 The PULL model workflow

1) ***Authorization:*** *This model inherits the SIP authorization mechanism from the PUSH model.*
2) ***Location change:*** *After the approval of authorization, the LIS name and its policy value are stored in the IM Manager (IMM) as part of the subscription information. In this model, location change does not trigger any action. An IM Notifier waits for an IM Subscriber sending a query to get its location information.*
3) ***Location request:****, An IM Subscriber performs periodic checks for subscribing location information of an IM Notifier because in the PULL model the IM Subscribe is active. The IM Subscriber user can decide how many times he/she wants to query the location information from an IM Notifier as well as the interval between each query.*
4) ***Location response:*** *Once an IM Subscriber sends a query and has been received by an IM Notifier, the IM Notifier will search the subscription list to get the IM Subscriber name and its policy value, and then send the location information to the IM Subscriber according to its policy value.*

Authorization

First, an IM Subscriber needs to send a SUBSCRIBE request to an IM Notifier to build up a partnership. Secondly, the IM Notifier returns a "202 Accepted" response to indicate that the SUBSCRIBE request has been received and understood. Thirdly, a subsequent NOTIFY request is sent out to inform the IM Subscriber that the IM Notifier is waiting for the user's input for authorization. Once the authorization is approved, a subscription will be created in the IM Manager, which stores the Subscriber's name and the policy parameter as part of the subscription information. When an IM Notifier changes a policy value for a particular IM Subscriber, a NOTIFY request will be sent out to inform the IM Subscriber of the policy change. The NOTIFY Request is only responsible for informing policy changes, but does not include any location data delivery.

Location change

As in the PUSH model, new location information is randomly fetched from a location database to simulate the users' movements. The location notifications are triggered by the users' movements. But in the PULL model, location change at the IM Notifier side will not trigger a location notification.

Location request

After authorization has been granted by an IM Notifier, an IM Subscriber can start to pull location information from the IM Notifier at anytime by sending a location MESSAGE Request. First, a user needs to enter the number of location queries and time interval between each query. The Request must contain a ContentTypeHeader with a value of "Location". To, From, CSeq, Call-ID, Max-Forwards and Via headers must be included to form a valid SIP Request.

Location response

First, an IM Notifier receiving a MESSAGE Request responds with a final Response immediately. A 2xx indicates that the message was delivered successfully. A 4xx or 5xx response indicates that the message was not delivered successfully. A 6xx response means it was delivered successfully, but refused. Secondly, IM Notifier inspects the value of the ContentTypeHeader to see if it is a location MESSAGE

Request or not. A non-location Request is processed as a normal MESSAGE while for a location MESSAGE Request, it sends back one MESSAGE with location content. The content of the body within the LPIDF format is subjected to the authorization policy. This model does not provide a retransmission mechanism to guarantee MESSAGE delivery.

4.5 Meeting the privacy requirements

According to the above-presented implementation, the answers to the five privacy requirements mentioned in chapter two are discussed below.

Dynamic response to circumstance

When a LIS wishes to subscribe to a particular state for a resource, it forms a SUBSCRIBE message. The SUBSCRIBE request will be confirmed with a final response. 200-class responses indicate that the subscription has been accepted, and that a NOTIFY will be sent immediately. A 202 response merely indicates that the subscription has been understood, and that authorization may or may not have been granted. Whenever a request comes, a LIN should be able to accept or deny the request, and reveal part of the geographic location information to the LIS. SIP provides the authorization mechanism to dynamically accept or deny an incoming SUBSCRIBE request. The authorization mechanism is used to provide real-time interaction. Hence, a user can give dynamic responses to the incoming requests. The current project offers three options for users: “authorize”, “authorize with privacy concern” and “reject”.

Level of deniability

A NOTIFY request will be sent to a Subscriber after sending a 202 response. RFC 3265 specifies that a NOTIFY message is sent immediately after any 200-class response to a SUBSCRIBE request, regardless of whether the subscription has already been authorized. 200-class responses to SUBSCRIBE requests do not generally contain any useful information beyond subscription duration. Their primary purpose is to serve as a reliability mechanism. The NOTIFY requests must contain a “Subscription-State” header with a value of “active”, “pending” or “terminated”. The

"pending" value indicates that the subscription has been received, but that policy information is insufficient to accept or deny the subscription at this time. This approach provides a degree of deniability, as a "Subscription-State" might be "pending" due to technical failures, lack of actual data, restricted access, or for some other possible reasons.

Coarse-grained control

When the authorization has been approved, a user can decide which parts of the LPIDF he/she wants to reveal by using the policy. The Policy has five values: "countryLevel", "provinceLevel", "cityLevel", "streetLevel" and "roomLevel".

User feedback

The Notifier contains a dynamic subscription list (Presence Manager for the PUSH model and IM Manager for the PULL model). When a subscription is created in the Notifier, it stores the "Event" header "id" parameter and the policy value as part of the subscription information into the Notifier. A subscription recorder is deleted when a Notifier sends a NOTIFY Request with a "Subscription-State" of "terminated". According to the subscription list, the LIN is able to indicate who is getting geographic information and at which level of location information the Subscriber is getting it.

Dealing with emergency situations

This question is presented at this point because it is not part of the PUSH and PULL models. Figure 4.8 shows the SIP PUBLISH flow for an emergency situation.

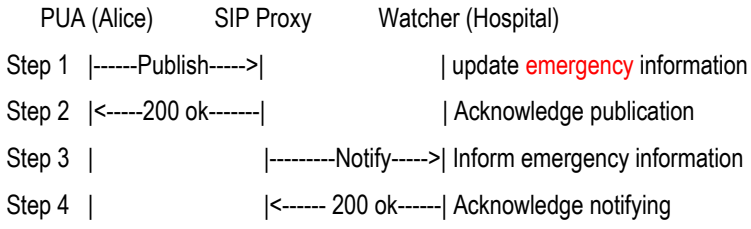


Figure 4.8 The SIP Message flow

When Alice encounters an emergency situation, Presence User Agent (PUA) publishes her geographic information to a SIP proxy, and the SIP Proxy notifies a hospital.

RFC3903 (Niemi, 2004) defines the type of SIP PUBLISH request. It is used in this project to push the location information from a Presence User Agent (PUA) to a SIP proxy. The SIP proxy acts as a re-distributor of that location information. The SIP Proxy can then notify the location information to public emergency response units like a hospital, fire department or police station. This does not require the modification of the PUA and Proxy's standard behaviors except their capability of interpreting the LPIDF document. The standard procedure can be found in RFC3261 (Rosenberg et. al, 2002) and RFC3903 (Niemi, 2004).

4.6 Summary

This chapter presented the implementation of the SIMPLY system and its key components. In order to communicate, a four-layer architecture uses the Listener/Provider event model to enable any SIP agent to send Requests to its desired SIP partner as well as receive Responses from its partner. On the top of the 4-layer structure, SIMPLEUA implements a SIPListener interface to process SIP messages according to the RFC recommendation, which includes Presence and Instant Messaging function. SIMPLEUA carries LO in LPIDF format instead of an SDP packet as payload in order to transfer location information. The basic function of LPIDF is to provide a common and extensible XML container for LO. Both the PUSH and PULL models have the capability of interpreting LPIDF. For the PUSH model, most of the modifications are made at the Presentity side which includes real-time authorization and policy decision-making. Based on the foundation of the PUSH model, the SIP MESSAGE Request is introduced in the system to form the PULL model for dedicatedly carrying location data between participants. The policy applies on both models in order to protect unwanted disclosure of sensitive information. But under emergency situation, SIP "PUBLISH" is used to distribute location information without policy constraints. The experimental design for the performance of the PUSH and the PULL model is going to be discussed in the next chapter.

CHAPTER 5 Experimental design

This chapter presents the experimental design for the comparison of the PUSH and PULL models. Experimental design plays a prominent role within the empirical evaluation methodology in answering the research questions. It is used to evaluate the quality of the proposed solutions. The experimental design is composed of the following parts: preparation of the testing environment, execution of the experiments, and collection of the experimental data.

5.1 Laboratory simulation-generating source location information

For economic and convenience reasons, real users are not involved in participating in the experiments. Some tools and methods are therefore used to create a fictive environment in the laboratory to simulate the real world. The GSM network is chosen as a potential positioning platform in this thesis because of its wide coverage, which in fact, can be replaced by any alternative positioning technology.

5.1.1 Understanding Street and Room level location

The Ericsson MPC map tool[2] is used to generate street and room level location data in a database through CGI-TA positioning technology on the GSM platform. With Ericsson MPC Map Tool, it is possible to create a mobile network for any chosen area, as well as routes with moving phones. A GSM cell is the basic unit of a mobile system and is defined as the geographical area where the radio coverage is given by one GSM base station. There are two types of cell (see Figure 5.1). In rural areas, it is common to place the base station in the middle of the cell and let the antenna be omni-directional (360°), which gives the widest geographical coverage. In urban areas, sector cells are often used. One base station is then placed where three smaller cells meet, with 120° coverage for each antenna.

[2] It is a free software that can be downloaded from Ericsson Mobility World http://www.ericsson.com/mobilityworld/sub/open/technologies/mobile_positioning/tools/mps_60_tool

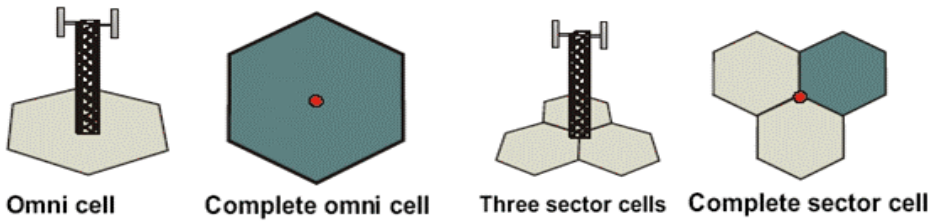


Figure 5.1 Two cell types

The omni cell covers 360° and gives the largest geographical coverage but with low capacity. The sector cell covers 120° and gives a small geographical coverage but high capacity. The coverage of a cell is treated as the "street" level location data in this thesis.

Timing Advance (TA) is a technique used to detect the distance between a mobile device and a base station. Technically, a mobile telephone signal from a base station far away requires a longer time to reach the mobile telephone than a closer located one. Based on this fact, the distance from the base station can be calculated according to the length of the consumed time. Figure 5.2 shows the coverage of CGI-TA detected area, which is more accurate than a full cell.

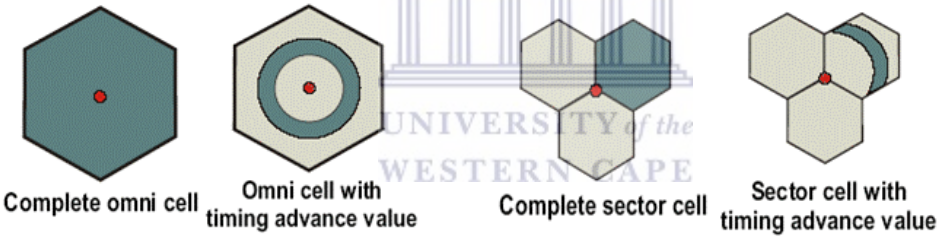


Figure 5.2 Cells with TA

CGI-TA is used to pinpoint a cell phone's location. The dark area is the result of the timing advance positioning method. It is treated as the "room" level location data.

In order to generate Room level data, a mobile phone is assumed to be moving constantly to cross a cell at a certain speed (for instance: 10km/hour), and then detecting its current position once every time offset (for instance: every 10 seconds). Depending on the time elapsed, the offset time marks every position the mobile telephone passed along (see Figure 5.3).

Figure 5.3 Detecting a moving mobile telephone's location by using CGI-TA

This simulates a mobile telephone moving cross a cell at a certain speed. TA is used to detect the mobile telephone's position every 10 seconds in order to get Room level location data.

5.1.2 Generating a location database for experiments

In order to run experiments for testing SIMPLY, a single location data must be enclosed in an LO and be transferred by SIP through the PUSH and PULL models. It is not necessary to collect data from the real world; a simulated worldwide location database will suffice. Location data contains five parts as mentioned in Chapter 4: country, province, city, street, room.

Purposive Sampling (Patton, 1990) is used to build up this database. It documents unique or variations of an interest field, and merges them to identify a common pattern for convenience reasons. According to Purposive Sampling, and to simplify the process, an assumption has been made to form an ideal five-level-structured world as follows: "Each country is exactly the same as South Africa which includes 9 provinces. Every province is exactly like Western Cape, which contains 106 cities (see definition of the city in Table 4.3 on page 43). Every city in every province is like Cape Town." According to the International Organization for Standardization (ISO) 3166 country list, there are 240 countries in the world. Therefore, up to city level, the database contains 240*9*106 cities, which are exactly like Cape Town.

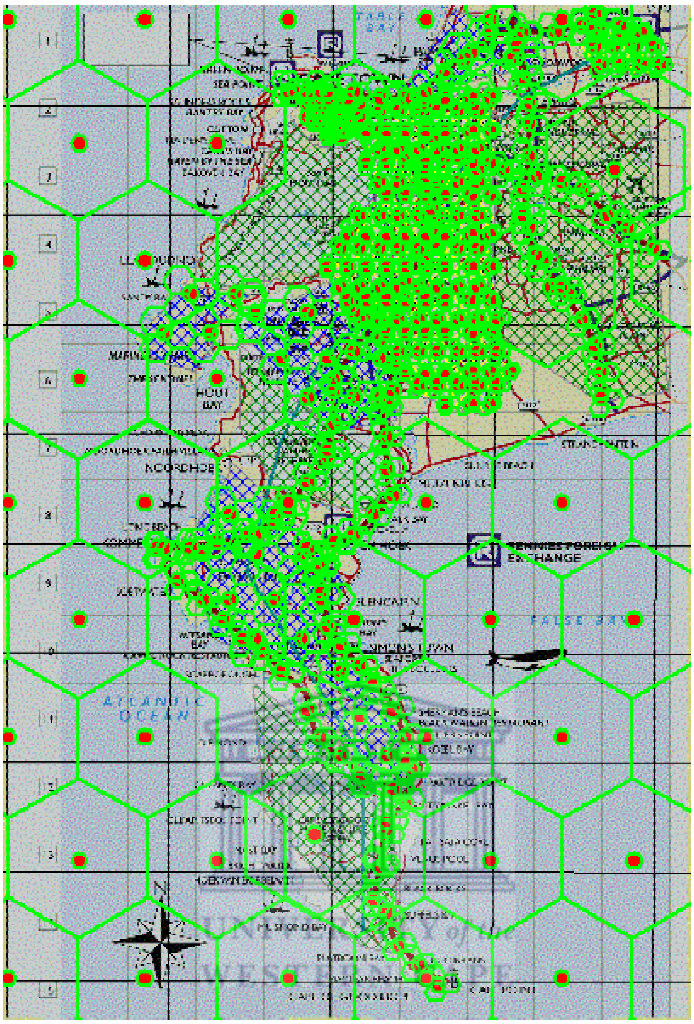


Figure 5.4 Generated GSM cell pattern of Cape Town

This is the GMS cell pattern of Cape Town generated by MPC map tool.

A GSM cell pattern is then generated for Cape Town, which contains 614 GSM cells (see Figure 5.4). At room level, let a mobile telephone move along a fictive route with a certain speed (decided by MPC map tool) to pass a randomly chosen GSM cell (Cell ID 244113_330) with a certain offset time (10 seconds). It takes 260 seconds to pass through the cell and generates 26 records of room level data. As a result of this, a worldwide five-level-structured database is generated (see Figure 5.5). The total number of records in this database is: 240*9*106*614*26 = 3,655,117,440. Any record of this database can be enclosed with LPIDF format and transferred through the SIP for testing purposes.

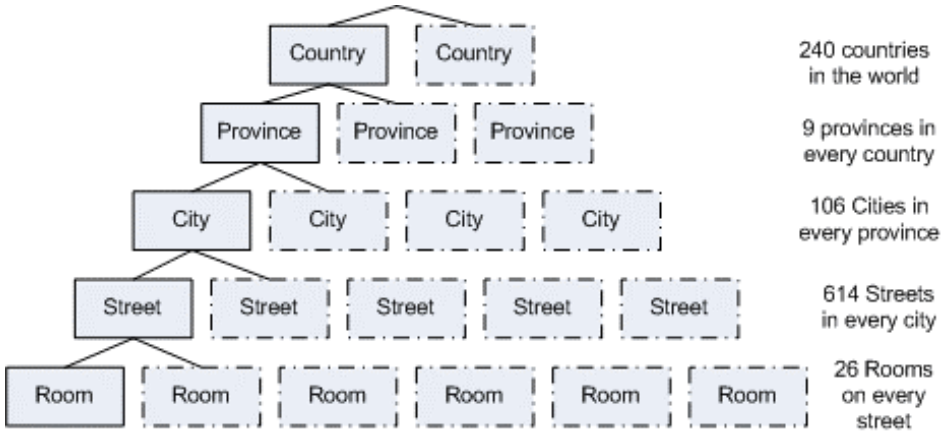


Figure 5.5 Generated database

This is the worldwide five-level-structured database generated for testing the PUSH and PULL models.

5.2 Executing the experiments

Four experiments are designed in this thesis: a system functionality test, a reliability test, a stability test, and a redundancy test. The first experiment is designed to test if the SIMPLY system is implemented as required. The remaining three experiments are designed to test the performance of the PUSH and PULL models under the situations which include fast location changes, large number of location changes, and long-term location data transfer. The goal of these three experiments is to understand the performance of the SIP when it is used as a LBS protocol by testing it under the PULL and PUSH models separately. By comparing them, the result of these experiments is expected to provide some guidance on how to use SIP as a LBS protocol. First of all, it is important to understand how SIP is used to deliver a LO in these two models (see Figures 5.8 and 5.9).

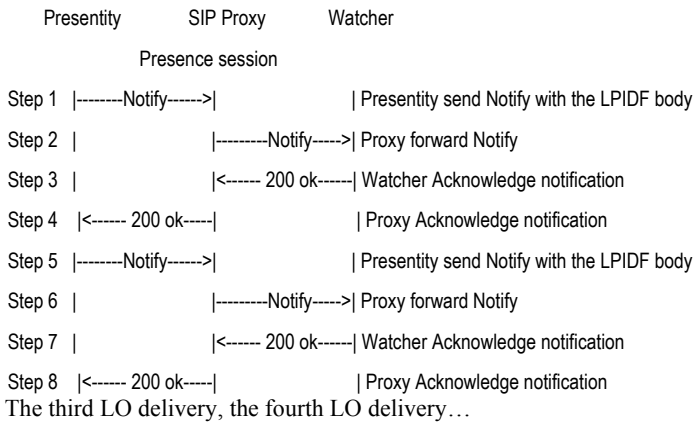


Figure 5.8 PUSH model LO delivery flow

For every single LO delivery, the PUSH model uses 4 SIP messages. The location information is contained in the LPIDF body and carried by a NOTIFY request from the Presentity. Step 1 to step 4 is used to deliver the first location change, step 5 to step 8 is used to deliver the second location change. The Push model keeps all the LO deliveries in one big session.

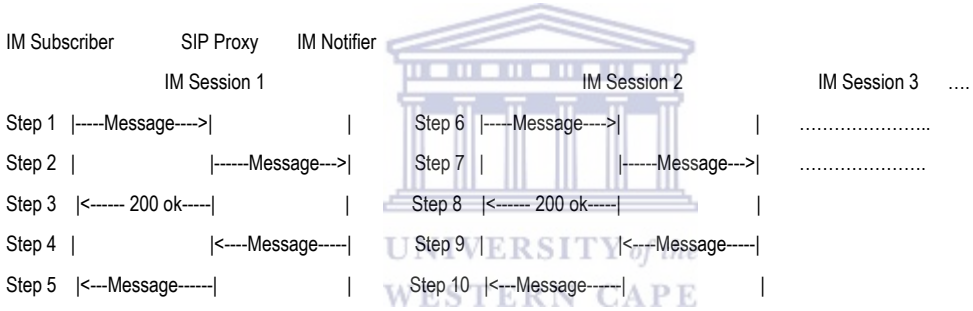


Figure 5.9 PULL model LO delivery flow

For every single LO delivery, the PULL model uses 5 SIP messages. The location information is contained in the LPIDF body and carried by a MESSAGE request from the IM Notifier. The PULL model terminates the old session and starts a new session for another LO deliverance.

5.2.1 System functionality test

The aim of this experiment is to verify whether the SIP can be used to transfer location data and if the policy is properly embedded in the SIP authorization mechanism to control the disclosure of location information. This includes two aspects. The first aspect is to know if the location information is correctly enclosed and transferred at the LIN side in the LPIDF format, received and correctly parsed at the LIS side. The second aspect is to know if the policy is properly applied on the LPIDF body.

System functionality: PUSH model test

The location data is randomly extracted from the simulated location database and then sent to the Presentity with the policy value changing from countryLevel to roomLevel (see Figure 5.6). The original location data from the database is compared with the received location data at the Watcher side to see if policy is properly applied.

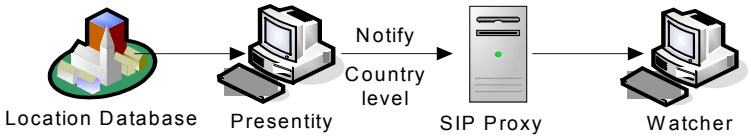


Figure 5.6 PUSH Model testing

***Policy test (policy** = countryLevel, provinceLevel, cityLevel, streetLevel, and roomLevel)*

1) *Set up the policy value*
2) *Extract location data from the location database and provide it to the Presentity*
3) *Collect generated LPIDF body from the Presentity to see if location data is properly enclosed.*
4) *Collect received location information from the Watcher to see if the policy is correctly applied*

System functionality: PULL model test

Similar to the PUSH model, the location data is randomly extracted from the generated location database and then sent to the IM Notifier with policy changing from countryLevel to roomLevel (see Figure 5.7). The IM Subscriber is used to pull the location information from the IM Notifier. The original location data from the database is then compared with the obtained location data at the IM Subscriber side to see if policy is properly applied.

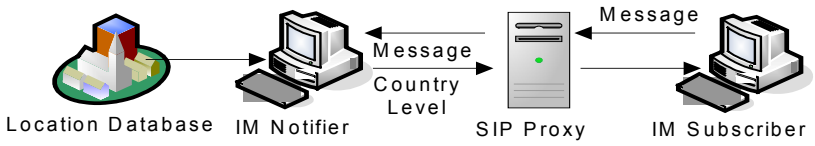


Figure 5.7 PULL Model testing

***Policy test (policy** = countryLevel, provinceLevel, cityLevel, streetLevel, and roomLevel)*

1) *Set up the policy value*
2) *Extract location data from the location database and provide it to the IM Notifier*
3) *Let the IM Subscriber pull the location information from the IM Notifier*
4) *Collect generated LPIDF body from the IM Notifier to see if location data is properly enclosed*
5) *Collect received location information from the Watcher to see if policy is correctly applied*

5.2.2 Reliability test

The goal of this experiment is to show the performance of the SIP when it is used as a LBS protocol by testing it under the PULL and PUSH models separately with extremely fast location changes. The data transfer speed is the key to test the transmission capabilities of these two models because they use different transmission methods (stateful and stateless). Under extremely fast data transfer conditions, there are always some data that fails to be delivered. But, as a reliable service, it should maximally tolerate the fast data transfer. The service should be able to reach 100% data delivery at a relatively fast speed. This speed is defined as the Minimum Reliable Interval (MRI) in this thesis. The purpose of this experiment is to find the MRI of each model for the comparison. The MRI is the scale against which the performance of each model is measured. The value of MRI reflects the reliability of each model because the smaller the MRI value is, the faster data transfer speed the model can bear.

Reliability: PUSH model test

This experiment repeats a designed action 100 times in order to deliver 100 LOes. The action entails randomly extracting a location data from the simulated location database and sending it to the Presentity. The action is repeated 100 times with a preset interval between each other, and these 100 actions are called a procedure. The Presentity and the Watcher must be kept connected within a single procedure in order to avoid any SIP session being terminated accidentally. A procedure is repeated three times at each preset interval. The interval starts from 0 milliseconds and increases by 25 milliseconds each time. If these 100 LOes are 100% delivered in all three times in a preset interval, then the interval is the MRI. Otherwise, the interval increases 25ms for another three procedures until all LOes are delivered in all three procedures (see Figure 5.10).

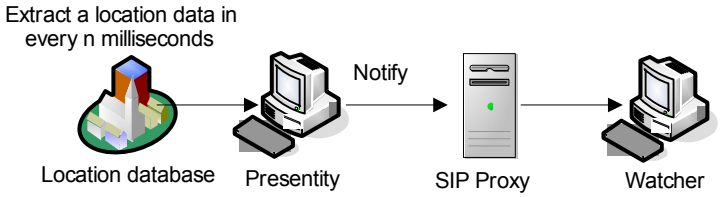


Figure 5.10 PUSH model reliability testing

Steps of a procedure with the preset interval n (n =0ms, 25ms, 50ms, 75ms…)

1) *Set up location change interval at n milliseconds*
2) *Randomly extract a location data from the simulated location database and send it to the Presentity every n milliseconds 100 times*
3) *Repeat step 2 three times*

Reliability: PULL model test

This experiment repeats a designed action 100 times in order to deliver 100 LOes. The action entails that the IM Subscriber pulls a location data from the IM Notifier. The action is repeated 100 times with a preset interval between each other, and these 100 actions are called a procedure. The IM Subscriber and the IM Notifier must be kept connected within a single procedure in order to avoid any SIP session being terminated accidentally. A procedure is repeated three times at each preset interval. The interval starts from 0 milliseconds and increases by 25 milliseconds each time. If these 100 LOes are 100% delivered all there times in a preset interval, then the interval is the MRI. Otherwise, the interval increases 25ms for another three procedures until all LOes are delivered in all three procedures (see Figure 5.11).

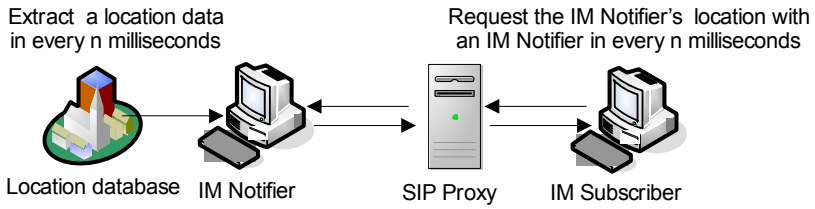


Figure 5.11 PULL model reliability testing

Steps of a procedure with the preset interval n (n =0ms, 25ms, 50ms, 75ms…)

1) *Set up the interval pulling location data for n milliseconds*
2) *Let the IM Subscriber pull a location data from the IM Notifier in every n milliseconds 100 times*
3) *Repeat step 2 three times*

5.2.3 Stability test

The goal of this experiment is to show the performance of the SIP when it is used as a LBS protocol by testing it under the PULL and PUSH models separately with fast and a large number of location changes. There is always a situation where people change their location very often and fast. The number of times a LIS can update its location easily exceed 1000, for example, a LIN user drives a car on the highway and grants a LIS with the policy value "roomLevel". Especially, when a LIS wants a long-term subscription, for instance, a trucking company sometimes needs to monitor a truck's movement for a whole day. The longer subscription means more location updates. With fast and a large amount of location updates, the service is expected to be stable. A stable service is self-balanced. It should not rely on occupying more and more physical resources to handle extreme situations, and thus cause instability in the computer on which the service runs such as occupying more and more memory or CPU capacity of the computer leading to the eventual collapse of the whole system. The purpose of this experiment is to examine the possibility that each model could cause the system to become unstable under the situations of fast and large number of location changes. For high-level debugging, SIMPLY writes what is occupied in the memory by each model into a log file in real time. Hence the log file size dynamically reflects the memory occupation of this model. It is the scale used in this experiment to measure the performance of each model.

Stability: PUSH model test

This experiment carries out 10 trials. Each trial repeats a designed action a different number of times in order to deliver a different number of LOes. An action is the process of delivering a LO. The number of actions starts from 100 until 1000 and increases by 100 per trial. The action entails randomly extracting a location data from the simulated location database and sending it to the Presentity. The interval between each action remains 1 second for every trial. The Presentity and the Watcher must be kept connected within a single trial in order to avoid any SIP session being terminated accidentally. The log file size of each trial is recorded (see Figure 5.12).

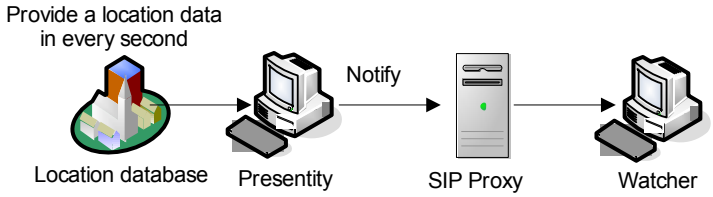


Figure 5.12 PUSH model stability testing

Steps of a trial with n actions (n =100, 200, 300, 400, 500, 600, 700, 800, 900, 1000)

1) *Set up location change interval at 1 second*
2) *Randomly extract a location data from the simulated location database and send it to the Presentity every 1 second for n times*
3) *Record the log file size*

Stability: PULL model test

This experiment carries out 10 trials. Each trial repeats a designed action a different number of times in order to deliver a different number of LOes. An action is the process of delivering a LO. The number of actions starts from 100 untill 1000 and increases by 100 per trial. The action entails that the IM Subscriber pulls a location data from the IM Notifier. The interval between each action remains 1 second for every trial. The IM Subscriber and the IM Notifier must be kept connected within a single procedure in order to avoid any SIP session being terminated accidentally. The log file size of each trial is recorded (see Figure 5.13).

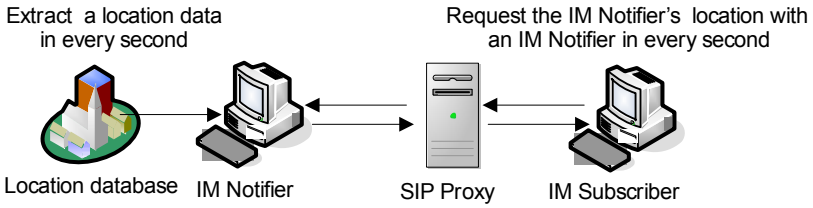


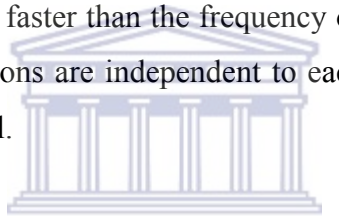
Figure 5.13 PULL model stability testing

Steps of a trial with n actions (n =100, 200, 300, 400, 500, 600, 700, 800, 900, 1000)

1) *Set up the interval pulling location data for 1 second*
2) *Let the IM Subscriber pull a location data from the IM Notifier every 1 second for n times*
3) *Record the log file size*

5.2.4 Redundancy test

The redundancy test is an experiment only for the PULL model because the redundancy only happens in the PULL model. In the PUSH model, only the LIN participates in the location updating. It delivers a location change to a LIS as soon as they are available without being queried by the LIS. Hence, a LIN will not push the same location change to a LIS twice. However, in the PULL model both LIS and LIN participate in the updating of location information which includes two actions: the IM Notifier updating its location information and the IM Subscriber's pulling location data from the IM Notifier; and these two actions have their own frequencies. The IM Subscriber might actually fetch the same location information because the frequency of the LIN updating its location is slower than the frequency of IM Subscriber pulling action. The IM Subscriber might miss some location changes because the frequency of LIN updating its location is faster than the frequency of the IM Subscriber pulling action. Because these two actions are independent to each other, the PULL mode is seen as an asynchronous model.



There are two factors that could cause redundancy in the PULL model. One cause of redundancy is avoidable. The solution is to let the frequency of the LIN updating its location be equal to the frequency of IM Subscriber pulling action. The other cause of redundancy is unavoidable because the behavior of the data link layer (the layer 2 of the IP network) is not deterministic. An application-level data is broken down into many IP packets to be transferred through the IP network. These IP packets are re-assembled to the original application-level data at the destination computer. The application layer does not have direct control of the data link layer. If the LIS sends out the requests at high speed, the timing order of these requests is not guaranteed to arrive in the same timing order at the LIN side. Hence, even if letting the frequency of the IM Notifier and the frequency of IM Subscriber be equal to one another before being transferred through the IP network, these two frequencies are not guaranteed equal to one another after the transmission of the IP network. However, the rate of redundancy is reduced when both the IM Subscriber and the IM Notifier reduce their

frequencies. In order to eliminate avoidable redundancy, these two frequencies are kept equal in this experiment. The purpose of this experiment is aimed at understanding how the unavoidable redundancy is affected by this frequency.

Redundancy: PULL model test

In order to eliminate avoidable redundancy, the interval of the IM Notifier and the interval of IM Subscriber are kept equal to one another in the experiment. This interval is defined as Common Interval (CI) in this thesis. This experiment carries out 3 trials in order to test redundancy at three different values of the CI: 10ms, 100ms, and 1000ms. A trial repeats the same procedure five times at each value of the CI. Each procedure repeats a designed action 1000 times in order to deliver 1000 distinct LOes. The action entails that the IM Notifier updates its location data with the preset value of the CI; the IM Subscriber pulls a location data from the IM Notifier with the preset value of the CI. The IM Subscriber and the IM Notifier must be kept connected within a single procedure in order to avoid any SIP session being terminated accidentally. The number of distinctive LOes received at the IM Subscriber side is recorded.

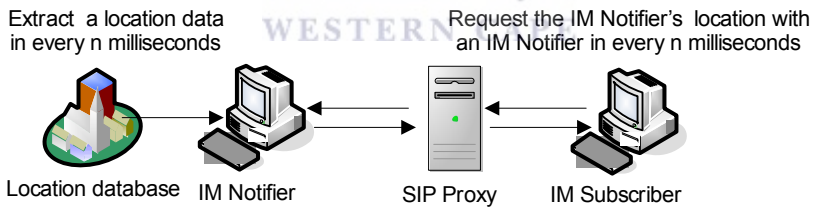


Figure 5.14 PULL model redundancy testing

Steps of a trial with the preset value of the CI n (n =10ms, 100ms, 1000ms)

1) *Set up the interval of updating the location data at the IM Notifier side for n milliseconds*
2) *Let the IM Subscriber pull a location data from the IM Notifier in every n milliseconds 1000 times*
3) *Count only the number of distinct LOes received at the IM Subscriber side*
4) *Repeat step 2 and 3 five times*

5.3 Data collection procedure

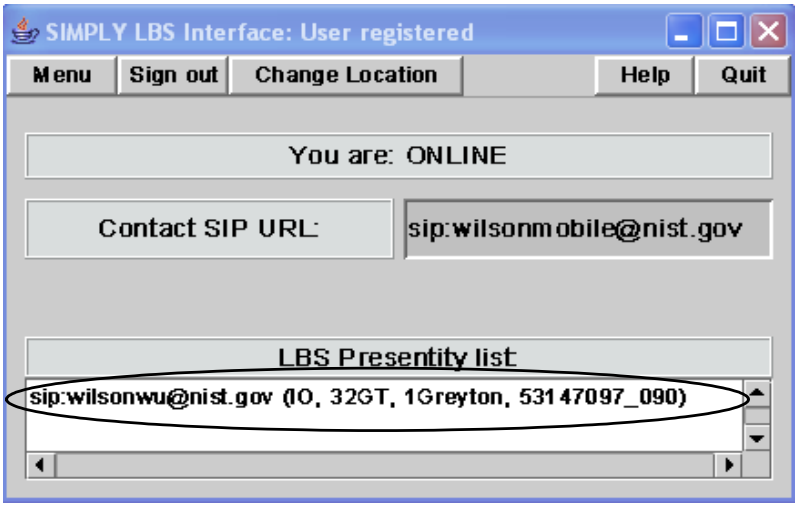
This section describes how the data is collected for aforementioned four experiments, but not the actual data collection. The actual data collection is presented in Chapter 6.

5.3.1 System functionality: data collection

The first experiment is to test the basic functionality of the SIMPLY system. Four type of data needs to be collected in this experiment: original location data from the location database, policy, generated LPIDF body at the LIN side, and received location data at the LIS side. The original location data, the policy value and generated LPIDF body are printed out by the NetBeans IDE (see Figure 5.15). The received location data is indicated by the SIMPLY interface (see Figure 5.16).

This data illustrates if an LO is correctly enclosed in LPIDF format and if the policy is properly applied on the LO at the LIN side.

Figure 5.15 LIN data



PUSH model

PULL model

Figure 5.16 LIS data

These data indicate if a LO is received and correctly parsed at the LIS side.

5.3.2 PUSH and PULL performance data collection

The reliability, stability and the redundancy experiments are carried out to test the performance of the PUSH and PULL models. These experiments require that the same action be repeated over hundreds of times under the same conditions. They use a software agent to collect data on a potentially large and ongoing basis automatically. The software agent includes two parts. One part is a MySQL database which is used to collect all the received LOes at the LIS side. The other part is the call-flow traces viewer[3], which is a simple graphical user interface for visualizing SIP signaling traces. The traces viewer retrieves a trace log file from the proxy and displays message sequence as a set of arcs that pass between LIS and LIN. LIN and LIS are identified by their IP addresses and ports (see Figure 5.17).

[3] Source code is available at http://dns.antd.nist.gov/proj/iptel/

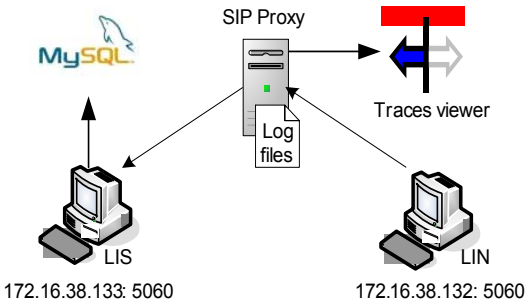


Figure 5.17 The software agent for data collection

All received LOes are stored into a MySQL database. The number of received LOes reflects the reliability for both PUSH and PULL models and the rate of redundancy for the PULL model. SIP Proxy stores all the SIP Requests and Reponses generated between the LIS and the LIN in log files. The log file size dynamically reflects the memory occupation of each model. The traces viewer is used to read these log files and visualize them for data analysis.

The traces viewer is used to help analyze the performance of each model. It groups the SIP sessions by their call identifiers. The SIP messages in each session are sorted by time, and Responses are matched with Requests and appropriately color coded (see Figure 5.18).

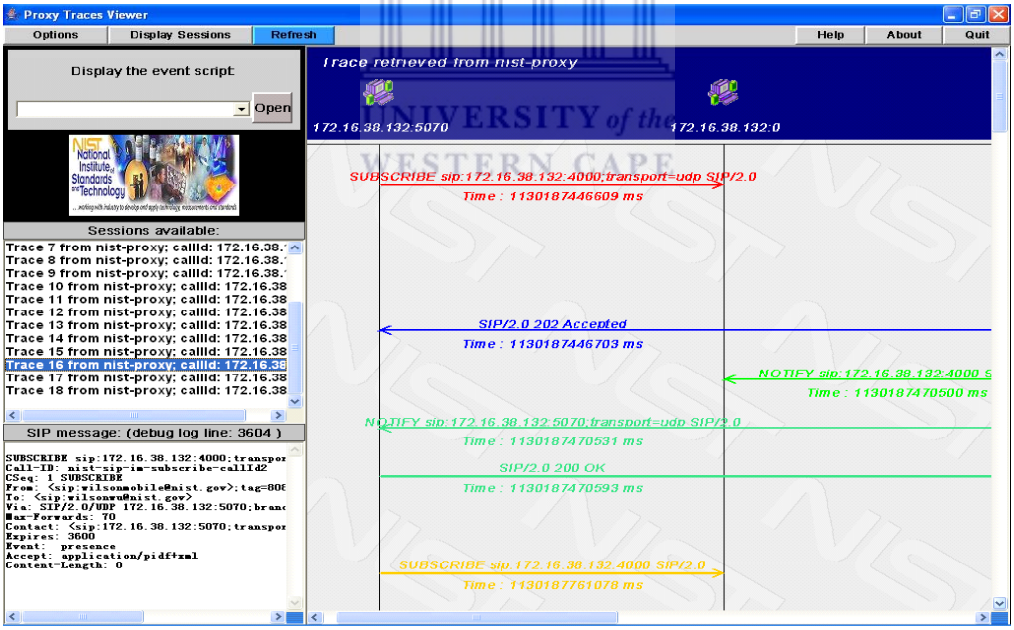


Figure 5.18 Traces viewer

The traces viewer shows all sessions generated in one experiment. It visualizes all the messages within a session with time information which indicates the consumed time of processing a stateful/stateless message. It also shows the number of messages passed through the stateful/ stateless proxy, and the number of messages failing to pass through.

5.4 Summary

In order to run experiments for testing the SIMPLY system, location data needs to be enclosed in an LO and be transferred by SIP through the PUSH and PULL models Laboratory simulation is used to generate a worldwide five-level-structured database for this purpose. The aim of this chapter is to show whether SIP can be used as a LBS protocol to transfer location information in a private manner, and more importantly to measure its performance with each model. This includes four experiments: a system functionality test, a reliability test, a stability test and a redundancy test. The experiment data is collected by SIMPLY itself for the first experiment, while for the other three experiments; data is collected by a software agent due to the potentially large and ongoing process of the experiment. Chapter 6 presents the visualization and the analysis of the actual collected data.

CHAPTER 6 Data collection and analysis

This chapter includes the data collection, visualization and analysis of the four pre-described experiments in Chapter 5: the system functionality test, the reliability test, the stability test and the redundancy test. The purpose of the first experiment is to test if the SIMPLY is functioning as required. The other three experiments are designed to test the performance of the PUSH and PULL models. The data reveals the advantages and disadvantages of each model which are discussed in detail.

6.1 System functionality test

SIP is used as a GEOPRIV "Using Protocol" in this thesis for transferring location information. In order to provide privacy, the policy is embedded in the SIP authorization mechanism to control the disclosure of the location information. The aim of this experiment is to verify whether the implemented SIMPLY system is functioning as required. This includes two aspects. The first aspect is to know if the location information is correctly enclosed and transferred at the LIN side in LPIDF format, received and correctly parsed at the LIS side. The second aspect is to know if the policy is properly applied on the LPIDF body.

6.1.1 System functionality data collection

Four types of data are collected in this experiment: the original location data from the location database, the policy value, the generated LPIDF body at the LIN side, and received location data at LIS side. The original location data, the policy value and generated LPIDF body are printed out by the NetBeans IDE and the received location data is indicated by the SIMPLY interface (see Chapter 5). The PUSH and PULL model data is collected and put in tabular form in Table 6.1 and Table 6.2.

Table 6.1 The PUSH model data of the system functionality test

Policy	The location data from the simulated database	Properly enclosed in LPIDF body	Received location data
countryLevel	KE 111LP 3DeKelders 47156096_210 117E182829S332859	Yes	KE
provinceLevel	UY 229GT 1Buffelsjagbaai 45221120_330 395E182847S332950	Yes	UY 229GT
cityLevel	SV 65NW 7Montagu 76235092_090 471E182828S332854	Yes	SV 65NW 7Montagu
streetLevel	BA 28GT 1Robertson 79242130_330 566E182928S333032	Yes	BA 28GT 1Robertson, 79242130_330
roomLevel	YT 138EC 5Wellington 4214118_210 349E182914S333018	Yes	YT 138EC 5Wellington 4214118_210 349E182914S333018

The policy value is set to change from countryLevel to roomLevel. The location data is randomly extracted from the simulated location database and then sent to the Presentity which means that the database is correctly connected to the SIMPLY system. The generated LPIDF body is collected to see if the location information is correctly enclosed. The received location is collected data at the Watcher side to compare with the original location data from the database to see if the policy is properly applied

Table 6.2 The PULL model data of the system functionality test

Policy	The location data from the simulated database	Properly enclosed in LPIDF body	Received location data
countryLevel	TW 209EC 5LambertsBay 16128105_090 35E182838S332925	Yes	TW
provinceLevel	CD 51NC 9Franskraal 49200087_330 251E182904S333008	Yes	CD 51NC
cityLevel	TG 214ML 4Uniondale 92070175 11E182839S332929	Yes	TG 241ML 4Uniondate
streetLevel	LK 201WC 6Saldanaha 30202112_090 267E182828S332854	Yes	LK 201WC 6Saldanaha 30202112_090
roomLevel	EE 68GT 1Trawal 33235098_330 482E182908S333011	Yes	EE 68GT 1Trawal 33235098_330 482E182908S333011

The policy value is set to change from countryLevel to roomLevel. The location data is randomly extracted from the simulated location database and then sent to the IM Notifier which means that the database is correctly connected to the SIMPLY system. The generated LPIDF body is collected to see if the location information is correctly enclosed. The received location data is collected at the IM Subscriber side to compare with the original location data from the database to see if the policy is properly applied

6.1.2 System functionality data analysis

The location data extracted from the simulated database represents the new location of the LIN host. The MySQL database is used to store the location data. Java Database Connectivity (JDBC) API is used to connect MySQL with the SIMPLY system because it is implemented in Java. The extracted location information is controlled by the policy. The policy is actually a system parameter (countryLevel, provinceLevel, cityLevel, streetLevel and roomLevel) dynamically set by a user on a LIN host. This task is accomplished through triggering a real-time authorization process. According to the implementation (described in Chapter 4), a real-time

interaction authorization process should be initiated when a LIS host sends a location request to a LIN host. This is achieved through the LIS host sending the first SUBSCRIBE request to the LIN. In order to prevent unnecessary information from being disclosed to a third party, a location data is captured, stored, and processed on a local B2BUA (see chapter 4). The policy is therefore applied on the location information before being sent out. After the policy value has been set by the LIN user, the location data should be revised according to the policy value and enclosed in a LPIDF body through the LPIDF parser. After the location information is revised by the policy and enclosed in LPIDF format, the LPIDF body is inserted in the payload of a SIP message and delivered to the LIS. The received LPIDF body is parsed by the LPIDF parser at the LIS side. This experiment shows that the MySQL database is correctly connected to the SIMPLY system and new location data is randomly extracted from it. The policy was accurately set on the LIN device through the real-time interaction authorization. The location data is correctly encLOesed in the LPIDF format by the LPIDF parser at LIN side, delivered by SIP messages and correctly parsed by the LPIDF parser at the LIS side.

6.2 Reliability test

This experiment tests the reliability of the PUSH and PULL models with extremely fast location changes. The MRI is defined to measure the performance of each model (see Chapter 5). The value of the MRI reflects the reliability of each model because the smaller the MRI value is, the higher the data transfer speed this model can bear. The purpose of this experiment is to find the MRI of each model for the comparison.

6.2.1 PUSH model data collection

This experiment repeats a designed action 100 times with a preset interval between each other. These 100 actions are called a procedure. The purpose of a procedure is to deliver 100 LOes with different transfer speeds. A procedure is repeated three times at each preset interval. The interval starts from 0 milliseconds and increases 25 milliseconds each time until all LOes are delivered in all three procedures. All the received LOes are stored in a MySQL database and the number of received LOes at

the LIS side is counted for every procedure. The PUSH and PULL model data is collected and shown in Table 6.3 below.

Table 6.3 Number of LOes received per time interval for the PUSH model

Interval / Procedure No.	0ms	25ms	50ms	75ms	100ms	125ms	150ms
1	13	12	30	58	61	63	100
2	5	17	38	100	59	100	100
3	6	20	30	58	60	100	100
Average received number	8	16	33	72	60	88	100

This table records the number of received LOes for every procedure. A procedure is repeated three times at each preset interval in order to calculate an average number of received LOes. This number represents the number of received LOes at this certain preset interval. It is used to draw a figure for comparison.

Table 6.4 Number of LOes received per time interval for the PULL model

Interval / Procedure No.	0ms	25ms	50ms	75ms	100ms	125ms	150ms
1	98	100	100	100	100	100	100
2	100	100	100	100	100	100	100
3	100	100	100	100	100	100	100
Average received number	99	100	100	100	100	100	100

The table records the number of received LOes for every procedure. A procedure is repeated three times at each preset interval in order to calculate an average number of received LOes. This number represents the number of received LOes at this certain preset interval. It is used to draw a figure for comparison.

6.2.2 Reliability data visualization

The average received numbers in Table 6.3 and 6.4 are used to draw Figure 6.1. The figure shows that the MRI of PUSH model is 150ms, while the MRI of the PULL model is 25ms. This figure also displays that the PUSH model has poor and unstable performance, while the PULL model has a stable performance with high reliability at high location change rates.

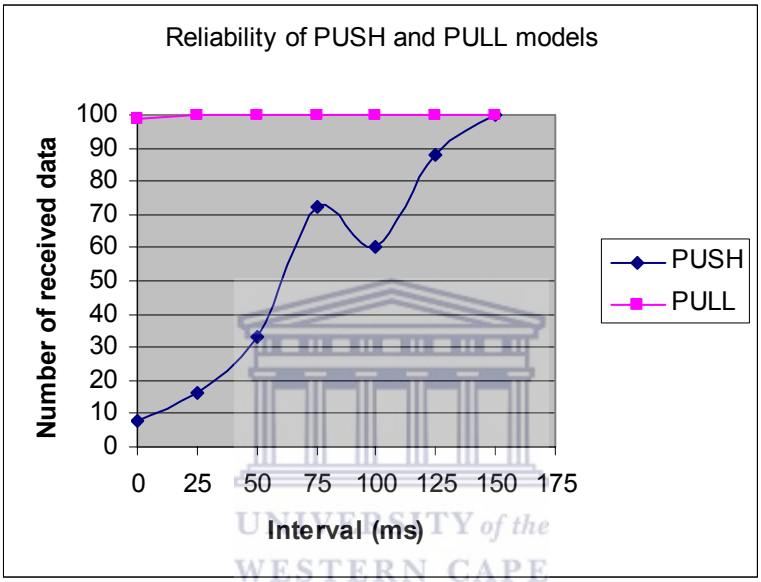


Figure 6.1 Reliability test

The figure shows that with 0ms interval, only 8 LOes were received in the PUSH model while 99 LOes were received in the PULL model. In the PUSH model, the number of received LOes rises above 70 when the interval is 75ms and drops to 60 when the interval is 100ms, which reflects that the performance of the PUSH model is unstable before the interval reaches the value of the MRI. In the PUSH model, the number of data increases as the interval increases up untill 100% while in the PULL model the data received is constantly above 99%.

6.2.4 Reliability data analysis

The PUSH model

The PUSH model builds on the SIP Presence model, and initiates a Presence session between a Watcher and a Presentity through the first SUBSCRIBE request. All of the following SIP messages are confined within this session. In order to maximize the delivery of SIP messages (see Figure 4.2 on page 39), stateful transaction is used to handle application-layer retransmissions, matching of responses to requests, and

application-layer timeouts. In every LO delivery, the SIP proxy acts as a stateful proxy which constructs a new server transaction for an incoming Request and a new client transaction to forward the Request (see Figure 6.2). This process consumes 37ms on average as recorded by the Traces viewer on a 2.4GHz CPU desktop, and this becomes the bottleneck of the PUSH model under a fast location change rate. Furthermore, the PUSH model contains all the LO delivery in a large session which makes it a fragile and vulnerable model. Any LO delivery failure will cause the failure of the rest of the following LO deliveries. Hence, only relatively few LOes were received at the Watcher side.

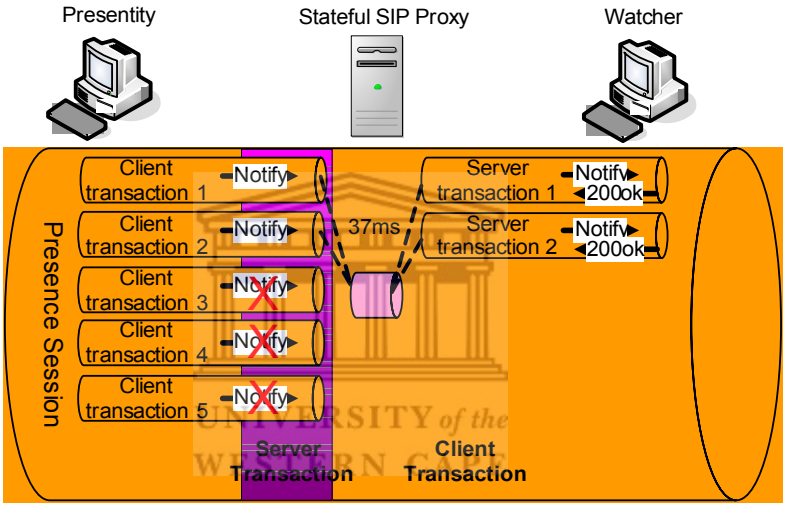


Figure 6.2 PUSH model LO delivery process

The PUSH model uses stateful transactions. All LO deliveries are kept in one Presence session. For each LO delivery, a stateful proxy constructs a new server transaction to handle an incoming Request and a new client transaction to forward the request. This process consumes 37ms on average as recorded by the Traces viewer on a 2.4GHz CPU desktop. Because only a single session is used in this model, one delivery failure will cause the failure of other LO deliveries.

The PULL model

In the PULL model, the MESSAGE request carries the actual communication data between participants. This avoids the LO deliveries in the pre-built Presence session by the first SUBSCRIBE request, and consequently forming a large and ever growing session. The intention of the PULL model is to separate every LO delivery into an individual session (see Figure 6.3). Therefore, a stateless transaction is used to deliver

an LO because it does not have any notion of transaction. The SIP proxy simply acts as a message forwarder. It takes messages, both requests and responses, directly from the transport layer without creating client and server transactions. This means that the PULL model is able to fully utilize the bandwidth instead of depending on the speed of creating transactions. Therefore, a large number of SIP messages can pass through a SIP proxy in a short time. In contrast with the PUSH model, the PULL model keeps the number of SIP messages within a session small and constant, thus eliminating the likelihood of errors occurring (see Figure 5.9). Therefore, a single LO failure will not affect other LO deliveries.

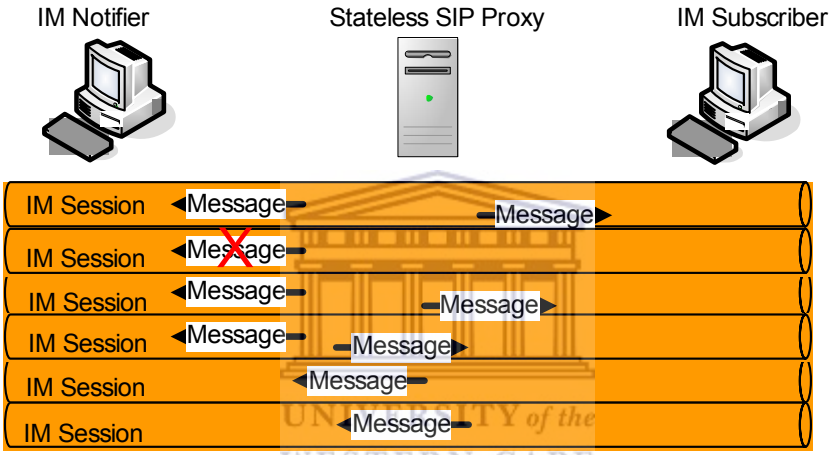


Figure 6.3 PULL model LO delivery process

The PULL model uses stateless transactions. It neither creates any transactions nor keeps all the LO deliveries in a large session. Each LO is delivered in an individual session with 5 SIP Messages. This keeps the number of SIP messages within a session small and constant. Because this model uses multiple sessions, one delivery failure will not affect another.

6.3 Stability test

This experiment tests the performance of the PULL and PUSH models with fast and a large number of location changes. A stable service is self-balanced. It should not rely on occupying more and more physical resources to handle extreme situations. The log file size which reflects the memory occupation is recorded. The content of the log file is used to help analyze the cause of instability (see Chapter 5). The purpose of this experiment is to examine the possibility of what could cause the system instability in each model under the situation of the fast and a large number of location changes.

6.3.1 Stability data collection

This experiment carries out 10 trials in order to deliver different a number of LOes. The number of LOes starts from 100 until 1000 and increases by 100 per trial. The interval between each LO delivery remains 1 second for every trial. The log file size of each trial is recorded. The PUSH and the PULL model data is collected and entered in the tabular form in Table 6.5 below.

Table 6.5 The size of generated log files for the PUSH and the PULL models (Mb)

Number of messages / Model	100	200	300	400	500	600	700	800	900	1000
PUSH	2.2	7.6	16.1	27.6	42.4	60.2	81.0	105.3	132.5	162.9
PULL	0.9	1.7	2.5	3.3	4.2	5.0	5.9	6.7	7.5	8.4

This table records the size of the generated log files of every trial. These data are used to draw a figure for comparison of the PUSH and PULL models.

6.3.2 Stability data visualization

The NIST stack supports logging of SIP messages into a log file. The log file is used to record the output from the proxy for high-level debugging. The Figure 6.4 shows that the size of the generated log files of both the PUSH and PULL models keep increasing when the quantity of location changes grows. However, the size of the generated log file of the PULL model keeps a normal linear increase, while the size of the generated log file of the PUSH model has an exponential increase which is abnormal. The exponential increase indicates that the memory occupation of the PUSH model grows along with the increasing of the number of LOes.

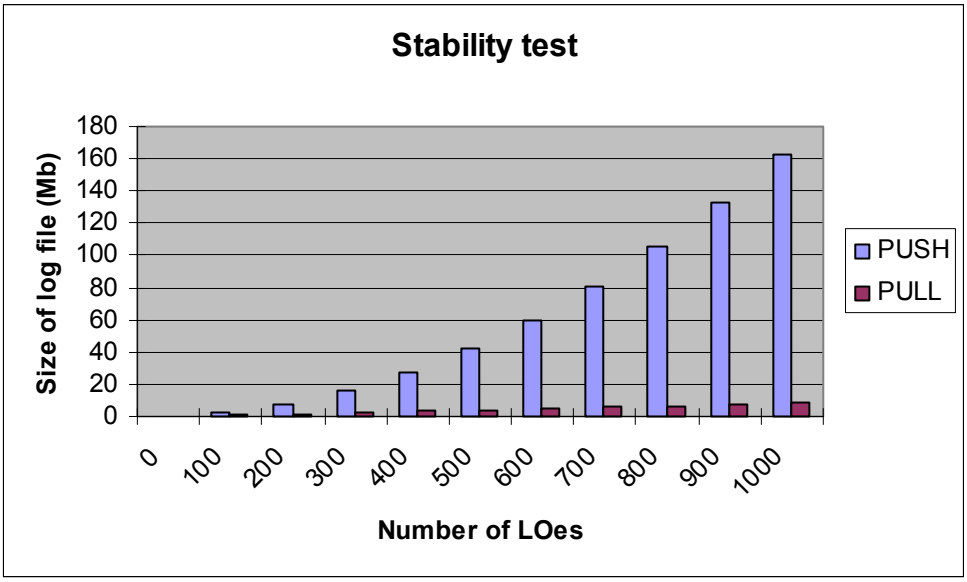


Figure 6.4 Stability test

With an increasing number of LOes, the size of the log file ought to increase. It should increase linearly when the number of LOes increases linearly. However, this figure shows that the size of the PUSH model log file has an abnormal increase. It grows much quick than the size of the PULL model log file. The size of the PUSH model log file boosts to above 160Mb while the PULL model log file size remains below10Mb when LIS's location has been updated 1000 times.

6.3.3 Stability data analysis

With the stateful transaction, the PUSH model needs to maintain a transaction table in order to record each transaction's state. The table maps a server transaction to its corresponding client transaction. With the stateless transaction, the PULL model does not require such a transaction table. The log file records the procedure of processing each message. Its size reflects the computer memory usage of each model. The life cycle of each transaction is managed by a Finite State Machine (FSM). The FSM consists of four states - Trying, Proceeding, Completed and Terminated. Each transaction created by an LO delivery will go through some of these states before it can be discarded from the transaction mapping table. When the speed of the discard of old transactions cannot catch up with the speed of creating new transactions, the transaction mapping table will keep growing, and make the SIMPLY system unstable. The SIP proxy, which runs on a 2.4GHz desktop, takes a longer time to process an incoming request. Because the transaction table keeps growing, this occupies more

and more computer memory. It also generates a huge size log file on the computer hard drive, and eventually threatens the whole system.

6.4 Redundancy test

Data redundancy only occurs in the PULL model because it includes two frequencies. One is the frequency of the IM Notifier updating its location information. The other is the frequency of the IM Subscriber's pulling action. In order to eliminate avoidable redundancy, these two frequencies should be kept equal to one another (see Chapter 5). In the PULL model, however, the action of the IM Subscriber pulling location data is independent of the IM Subscriber updating its location data. Hence, the PULL model is seen as an asynchronous model in which the IM Subscriber might pull duplicated data as well as some updated data missed from the IM Notifier.

6.4.1 Redundancy data collection

This experiment carries out 3 trials in order to test redundancy at three different values of the CI: 10ms, 100ms, and 1000ms. A trial repeats the same procedure five times at each value of the CI. Each procedure repeats a designed action 1000 times in order to deliver 1000 distinct LOes from the IM Notifier side. All the received LOes at the IM Subscriber side are stored in a MySQL database (see Chapter 5). The number of distinct received LOes is counted and entered in Table 6.6 below.

Table 6.6 The number of distinct LOes received at the LIS side

Procedure No. / Common Interval (CI)	1	2	3	4	5	Average number	Redundancy rate
10ms (1st Trial)	907	578	840	895	901	824	17.6%
100ms (2nd trial)	839	835	820	844	834	834	16.6%
1000ms (3rd trial)	945	950	946	950	952	949	5.1%

The table records the number of distinct LOes of every trial. A trial repeats the same procedure five times at each value of CI in order to calculate an average number of received LOes. This number represents the number of received LOes at this certain CI. The redundancy is calculated from the formula: Redundancy = (1000 – average number)/1000

6.4.2 Redundancy data visualization

The redundancy rates presented in Table 6.6 are used to draw Figure 6.5. This figure shows that the redundancy reduces along with the increase of CI, which indicates that the LIS pulls less redundant location data under a relatively slow location requesting. It also means that less location data is missed.

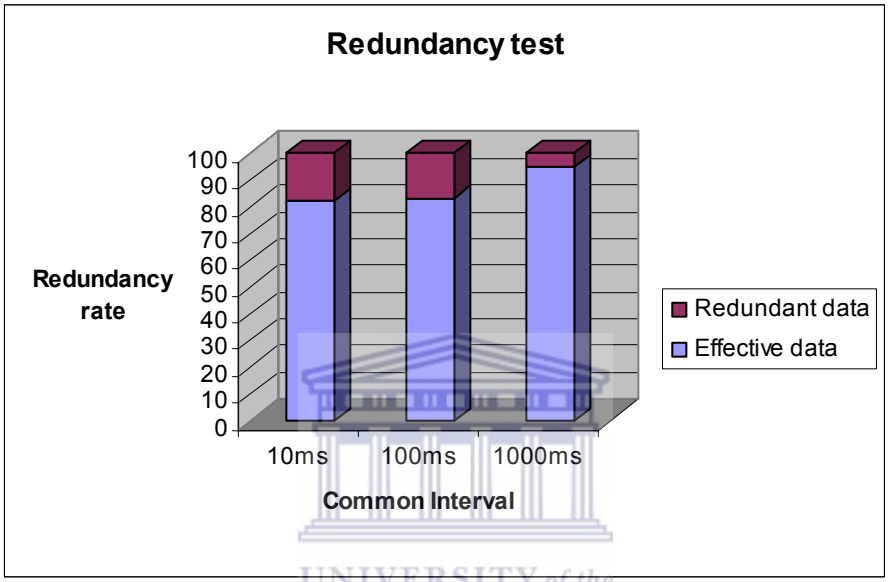


Figure 6.5 Redundancy test

The top part of each column indicates the redundant data of each trial. This figure shows that at the 10ms the redundancy rate is around 20%. The redundancy rate is slightly reduced at 100ms. However, the redundancy rate is greatly deceased at 1000ms; the value is around 5%.

6.4.3 Redundancy data analysis

In the PULL model, MESSAGE requests are directly forwarded by the SIP proxy. This greatly increases the message processing speed. The IM Subscriber sends the MESSAGE requests in a time order. When a large number of messages are transmitted in a short time from the IM Subscriber to the IM Notifier, the messages' timing order is shifted during the transmission process. As seen in Figures 6.6 and 6.7, if two MESSAGE requests arrive in the first time slot they both receive duplicated location data. If none of the messages arrive in the second time slot, the updated location data in this time slot is missed. Hence, in the PULL model, the efficiency and accuracy of location data is largely affected by the stability of the network

conditions. However, efficiency and accuracy increases with the reduction of pulling speed because the instability of the network is better tolerated by a longer time slot.

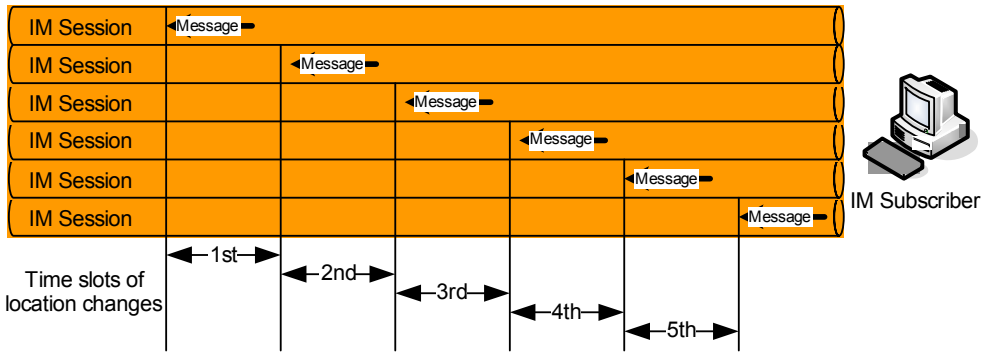


Figure 6.6 The MESSAGE Requests sending order

In expecting to obtain every location change from the IM Notifier, the IM Subscriber sends the MESSAGE requests at a frequency the same as the frequency the IM Notifier updates its location data. The first Message is expected to arrive within the 1st time slot, while the IM Notifier updates its first location at the beginning of the same time slot.

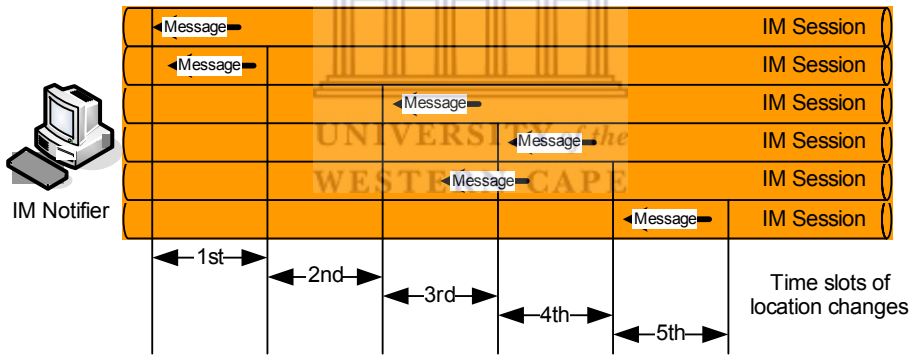


Figure 6.7 The MESSAGE Requests arrival order

Compare to Figure 6.6, the timing order is shifted along the way during the process of transmission, and thus causes the unexpected information redundancy as well as some missed location data.

Table 6.6 (on page 83) shows that there is no big difference between the redundancy rate of the 1st and 2nd trials. However, the number of received LOes in the 1st trial is fluctuating, while the number of received LOes in 2nd trial keeps steady. This is because the level of timing disorder is unpredictable under the conditions of extremely fast location requesting. The timing order could be largely affected by any small instability in network conditions. When the value of CI increases, the timing

disorder becomes more predictable. As shown in Table 6.6, the redundancy rate is greatly deceased in the 3rd trial. Generally, the LIS pulls less redundant location data as well as misses less location data along with the increase of CI.

6.5 Overall analysis

SIP is designed as an application-layer protocol for control purposes to make communication possible, while the communication itself needs to be achieved by another means. This project sees SIP from a different angle. SIP is used as an application-layer data transfer protocol to carry LOes. The comparison of the PUSH and PULL models is in fact an evaluation of SIP's stateful and stateless data transmission.

SIP entities that have a notion of transactions are called stateful. Such entities usually create a state associated with a transaction that is kept in memory for the duration of the transaction. When a request or response comes, a stateful entity tries to associate the request/response with existing transactions. If such a transaction exists then its state gets updated from the message. Otherwise, the stateful entity creates a new transaction to handle the message. The ability to associate SIP messages with transactions enables stateful proxies to perform more complicated tasks, which include accounting, forking, and Network Address Translation (NAT) traversal, and so on. In spite of all the benefits the stateful transaction could gain, stateful entities' data transfer performance is quite poor because they must maintain the state for the duration of the transactions, and some transactions last fairly long, thus causing the instability in the system. The experiments in this chapter demonstrate that a stateful entity cannot adjust well under the fast location changes. Compared to stateful entities, stateless entities ignore the complexity of the transaction process, thus making it simpler, faster and more stable. Stateless proxies are simply used as message forwarders. They forward messages independently of each other instead of arranging them into a big session which reduces the risk of errors occurring.

Stateless proxies do not keep the states of each transaction, which makes the PULL model a stable model when a large quantity of messages needs to be processed.

6.6 Summary

The System functionality experiment is used to verify whether SIP can be used to transfer LO and, for privacy purposes, if the policy can be embedded in the SIP authorization mechanism to control the disclosure of location information. The test shows that the location data can be transferred with the help of the LPIDF parser, and the policy is accurately set on the LIN device through the real-time interaction authorization and applied on the extracted location data. The SIP is extended in this thesis to adopt new features for the LBS but the robustness of this new usage still needs to be proved. Hence, three experiments are carried out to test the performance of the usage with situations such as fast location data transfer, large number of data transfers, and long-term data transfer. The advantages and disadvantages of each model are examined through these experiments. Reliability and stability tests show that the PUSH model has poor and unstable performance, while the PULL model has stable performance with high reliability under high location change rate conditions. However, redundancy test shows that the PULL model is seen as an asynchronous model in which the IM Subscriber might pull duplicated data as well as misses some location data from the IM Notifier. The result of these four experiments provides the important guidance for the use of the SIP to provide LBS, which is explained in Chapter 7.

CHAPTER 7 Conclusion and future research

The purpose of this chapter is to answer the research questions and draw a conclusion. The PUSH and PULL models provide a general framework for the SIP to provide LBS. SIMPLY was developed as a RI of the SIP-based LBS. The performance of each model is tested under the situations of fast location data transfer, large number of data transfers, and long-term data transfer. This thesis has a clear scope, which is to partially fulfill the requirements of GEOPRIV as a starting point of the SIP-based LBS. Hence, the direction for further development is also presented.

7.1 Addressing the research questions

This thesis starts with the IETF GEOPRIV working group, which endeavors to provide standard LBS protocols capable of transferring geographic location information for diverse location-aware applications. The GEOPRIV group points out the research direction of this thesis. This thesis shares its research principle. Instead of developing a new protocol, the GEOPRIV group tries to select already standardized formats and protocols, enhance selected formats and protocols and then recommend them for use. The GEOPRIV group also provides an important framework and guidance for this thesis. Two of their important concepts (LO and "Using Protocol") are employed in the thesis. Through careful observation, it is found that the SIP and GEOPRIV "Using Protocol" share many similarities. The research question is then proposed: "How can SIP be used as a "Using Protocol" for LBS to fulfill GEOPRIV requirements?" The research question is divided into two sub-questions: "How can the SIP be used to transfer location information in a private manner?" and "How the can SIP-based solution proposed by this thesis be used to provide stable services for LBS?"

In answering the first sub-question, the SIMPLY system was developed as a RI of the SIP-based LBS. This thesis defines five levels of sub-national divisions to represent a location address: "country", "province", "city", "street" and "room". In order to

contain these five levels of location information, the LPIDF, which is an extension of the PIDF, is defined in this thesis. It provides an extensible XML container for location information. The LPIDF body is inserted into the SIP payload and can be conveyed among the SIP entities. In order to provide privacy, SIP real-time authorization process is applied to dynamically set a system parameter called policy. The policy is used to control the disclosure of the aforementioned five levels of location information. On the basis of previous LBS privacy research described in Chapter 2, five privacy requirements have been summarized: dynamic response to circumstance, built-in plausible deniability, coarse-grained control, feedback and special exceptions for emergencies. All these privacy requirements are considered part of the original system design to provide privacy protection. The methodology used for the software development in this thesis is based on Exploratory Prototyping (Sommerville, 2000). The system functionality experiment was carried out to verify whether the implemented SIMPLY system is functioning as required.

In answering the second sub-question, the PUSH and PULL models were proposed as the potential solutions of the SIP-based LBS. Both of these two models are implemented on the basis of the SIMPLE platform: the SIP Presence function to provide a PUSH model LBS and the IM function a PULL model LBS. They are implemented in this thesis not only for accommodating different users' needs, but also because they use different data transfer methods in order to determine the unique advantages and disadvantages of each model. An empirical evaluation methodology (Weiderman, et. al, 1987) is used to evaluate the robustness of each model. Three experiments are designed to compare the performance of the PUSH and PULL models: the reliability test, the stability test, and the redundancy test. The advantages and disadvantages of each model are examined through situations such as fast location data transfer, large number of data transfers, and long-term data transfer. Some important guidance provided by the results of these three experiments is discussed in section 7.3 ahead.

7.2 Pursuing the research aim

This thesis endeavors to offer a pioneering effort by providing a general framework for the use of the SIP-base LBS. The thesis has a clear scope, which is to partially fulfill the requirements of GEOPRIV as a starting point of the SIP-based LBS, but leaves its advanced development (such as embedding privacy-protecting instructions inside the SIP message) to further research. The research aim described in Chapter 1 was achieved with following four aspects:

1. *A basic XML schema should be defined to contain minimum location information.*

This thesis defined a basic XML schema to contain minimum location information. It is called the LPIDF, an extension of the PIDF, and is used as a data format for containing basic location information. The LPIDF extends PIDF with an umbrella tag <PreLoc>, which might contain five sub-tags: <Country>, <Province>, <City>, <Street> and <Room>. The basic function of the LPIDF is to provide an extensible XML container for location information. Other location application developers can also extend the LPIDF format through XML Name Space for their own projects.

2. *The disclosure of location information should be directly and dynamically controlled by the end user.*

This thesis defined a system parameter called policy with five values: “countryLevel”, “provinceLevel”, “cityLevel”, “streetLevel”, and “roomLevel”. These five values indicate up to which level a user wants to reveal his/her location information. The policy is dynamically set through real-time interaction with a user. The authorization process is triggered when a LIS sent out the first SUBSCRIBE request to a LIN.

3. *Multimodal location-based services should be implemented in order to accommodate different users’ needs.*

This thesis proposed two concepts: “PUSH” and “PULL”. The “PUSH” concept means that a LIN is active; the LIN pushes every LIN’s location change to the LIS. The “PULL” concept means that a LIS is active; the LIS pulls every LIN’s location

change from the LIN. From a user point of view, a PUSH model user (LIS) does not need to query every time when he wants to know where his partner (LIN) is, but only needs to send one request to build a connection between them, and all his partner's location changes will be pushed to him through that connection. But, sometimes, it is not necessary to know every location change his partner made; the user might feel irritated by unwanted location updates. To avoid this situation, the PULL model helps the user to obtain his partner's location information whenever he wants to or at a certain pace decided by himself. Moreover, in the PULL model a constant connection is not needed. For instance, if he decided to check his partner's location once an hour then there is no need for them to keep connected for the whole hour.

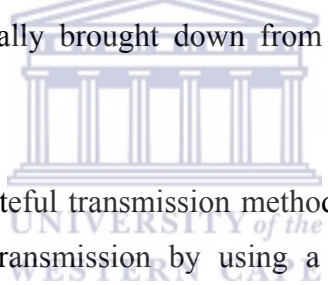
4. *Different data transfer methods should be investigated for consideration before recommending for use.*

The PUSH and PULL models are implemented in this thesis not only for accommodating different users' needs, but also because they use different data transfer methods in order to find the unique advantages and disadvantages of each. The PUSH model uses a transaction layer to deal with stateful transmission. The transaction layer handles application-layer retransmissions, matching of responses to requests, and application-layer timeouts. The PULL model is simpler than the PUSH model. The transaction layer is not in the notion of the PULL model because this model does not take care of transactions. Three experiments are designed to compare the performance of these two models in order to provide general recommendations for the use of the SIP to provide LBS.

7.3 Conclusion and recommendations

SIP, from its name (Session Initiation Protocol), indicates that it is a typical controlling protocol. The purpose of the SIP is to make communication possible by managing multimedia sessions among its participants; the communication itself must be achieved by another means (possibly other protocols such as RTP and SDP). This thesis sees the SIP from another perspective. It sees SIP as a transport protocol for

actual data transfer. SIP is used to transfer the LOes from a SIP entity to another SIP entity. Nevertheless, SIP is an application-level protocol, as GEOPRIV requires: a "Using Protocol" needs to inspect the content of LOes for the appropriate distribution, protection, usage, retention, and storage of the LOes instead of transfer them directly. SIP entities also need to inspect text-based SIP messages before delivering them. Its performance cannot be as good as transport layer protocols like TCP or UDP. But to some extent SIP can be seen as a transport-layer protocol for the use of LBS. SIP was originally conceived by the telephony industry as a protocol to set up and handle telephone calls. It is rare for a telephone call to last for several hours or even a whole day. It is very rare to transfer controlling signals over thousands times for managing a single audio or video session. However, the LBS requires that a trucking company, for example, could monitor its trucks' location for a whole day, and a truck could update its location information over a thousand times at a fast speed. Hence, SIP is not physically but hypothetically brought down from the application layer to the transport layer in this thesis.



The PUSH model uses the stateful transmission method that can be seen as TCP. It tries to guarantee the data transmission by using a transaction layer to handle application-layer retransmissions, matching of responses to requests, and application-layer timeouts. This method enables stateful SIP entities to perform more complicated tasks. A stateful proxy is able to calculate the billing charges, because it knows the time of the start of a call and its ending. A stateful proxy can try to reach a user's cell phone when the user is not in the office to pick up the office telephone. However, stateful transaction requires its participants to initiate a big session between them and all of the following location notifications are confined within this session, which makes it an unreliable model. Any LO delivery failure will cause the failure of the rest of the following LO deliveries. The reliability test proves that only relatively few LOes were received at the Watcher side under the fast location changes conditions. Each location notification is a transaction-creating request in SIP. The life cycle of each transaction is managed by a FSM. The FSM consists of four states - Trying,

Proceeding, Completed and Terminated. Each transaction created by an LO delivery will go through some of these states before it can be discarded from the transaction mapping table. When the speed of the discard of old transactions cannot catch up with the speed of creating new transactions, the transaction mapping table will keep growing and make the system unstable. The stability test shows that after 1000 location notifications, the log file size boosts to 162Mb.

The PULL model uses the stateless transmission method that can be seen as UDP. It does not use extra effort to guarantee the data transmission. The Stateless proxies are simply used as message forwarders. Therefore, this transmission method is able to fully utilize the bandwidth; a large number of SIP messages can pass through a SIP proxy in a short time. The stateless transmission method reduces the complexity of the stateful transmission method by getting rid of the transaction layer. SIP entities forward messages independently of each other instead of arranging them into a big session that reduces the risk of errors occurring. Therefore, a single LO failure will not affect other LO deliveries. The reliability test proves that above 99% LOes were received at the IM Subscriber side under fast location change conditions. The stateless transmission does not keep the states of each transaction in memory. It is a self-balanced transmission method which does not rely on occupying more and more system resources to handle extreme situations. The stateless transmission keeps a steady memory occupation, which makes the PULL model a stable model when a large quantity of messages needs to be processed. The stability test shows that the log file size maintains a normal linear increase. However, the disadvantage of this model is that the IM Subscriber might pull duplicated data as well as some updated data missed from the IM Notifier. This is because both the IM Subscriber and the IM Notifier participate in the updating of location information which includes two actions, which are independent to each other. Fortunately, this is not technical issue - it can be solved by borrowing the “PUSH” concept; let only one participant (IM Notifier) be involved in the updating of location information (if we neglect the users’

need who want to pull their partner's location information at a pace decided by themselves).

In conclusion, the SIP by itself is a controlling protocol. The SIP Presence extension (PUSH model) changes the SIP from a signal control protocol into a semi-data transfer protocol; the IM model extension (PULL model) changes the SIP to a full data transfer protocol. The PUSH model has part of the SIP controlling feature that enables it to accomplish more complicated tasks. If a LIN cannot deliver its location information to a LIS device, the LIN can try to redirect the location information to another LIS device for the same user. The SIP Presence is only suitable for low-speed data transfer. The data transfer speed is extremely crucial to this model. It is a fragile and vulnerable model under the high-speed data transfer conditions. The PULL model does not possess any SIP controlling feature. It cannot perform complicated tasks such as the PUSH model does, but it is suitable for heavy loads and fast data transfers. It reduces the complexity of the PUSH model by getting rid of the transaction process, thus making it simpler, faster and stronger. It is a proven reliable and stable model for fast data transmission. The recommendations for the use of the SIP to provide LBS presented by this thesis are as follows:

1. Separate the authorization process and location updating into different sessions

In the PUSH model, both the authorization process and the location updating are mixed in the same Presence session. The NOTIFY Request is responsible for both notify the LIS user its granted policy value and new location data. This makes the NOTIFY Request's responsibility unclear. The policy value is not an often-changed data. The user's location is an often-changed data. These two types of data should be separated into different sessions. It is suggested that Presence model should be used for the policy value updating and the IM model for the users' location data updating.

2. *Break a big session into many small sessions for the delivery of LOes under fast location change conditions*

The PUSH model uses 4 SIP messages for a single LO delivery while the PULL model uses 5 SIP messages for a single LO delivery. The PUSH model supposes to be more efficient than the PULL model. But because the PUSH model uses the re-transmission mechanism, this model ends up sending more SIP messages under the conditions of fast location changes. In addition, the PUSH model keeps all the LO deliveries in a large session, which is a big risk that could cause the delivery failure. The PULL model terminates the old session and starts a new session for another LO delivery, thus eliminating the likelihood of errors occurring.

7.4 Further research

This thesis provided an extensible XML container for location information. The GEOPRIV LO is far less defined for real use. Geography Markup Language (GML) is an XML-based encoding standard for geographic information developed by OGC. It is the most widely supported open specification for representation of geographic information. GML is one of the promising candidates that can be used to enhance the LPIDF. It provides a variety of objects for describing geography including coordinate reference systems, geometry, topology, time, units of measure and generalized values.

Embedding privacy-protecting instructions inside the SIP message is not addressed in this thesis. The XCAP is defined by the Presence community as a policy protocol and schema set. The XCAP is a direction for further research through which privacy rules can travel along with location information, and thus guide the Presence entities dealing with the location information.

Privacy concern in general is closely related to human psychology, social custom and a wide range of conceptions. The laboratory simulation approach used in this thesis is only a means to measure the SIP-based LBS application from a technical point of

view. In order to really understand privacy, real users are required to be involved in the experimental trails for further research.

BIBLIOGRAPHY

[1] Abowd G.D., Atkeson C.G., Hong J.I., Long S., Kooper R. and Pinkerton M., (1997). Cyberguide: A Mobile Context-Aware Tour Guide, *ACM Wireless Networks*, 3(5): 421-433.

[2] Altman I., (1975). *The Environment and Social Behavior: Privacy, Personal Space, Territory and Crowding*, Monterey, CA: Brooks/Cole publishers, ISBN 0818502096.

[3] Altschuk M. and Coleman R., (1994). Petition of the Cellular Telecommunications Industry Association for a Rulemaking to Establish Fair Location Information Practices. Rcd 6170, *FCC*.

[4] Beresford, A. and Stajano F., (2003). Location Privacy in Pervasive Computing, *IEEE Pervasive Computing*, 2: 46-55.

[5] Bossen C., (2002). The Parameters of Common Information Spaces: The Heterogeneity of Cooperative Work at a Hospital Ward, *Proc. of ACM Conference Computer-Supported Cooperative Work*, New Orleans, Louisiana, 176-185.

[6] Campbell B., Rosenberg J., Schulzrinne H., Huitema C. and Gurle D., (2002). Session Initiation Protocol (SIP) Extension for Instant Messaging, RFC 3428, *IETF*.

[7] Canal, G. and Cuda, A., (2001). Why SIP will pave the way towards NGN, *Proc. of the 7th ITU International Conference on Intelligence in Networks (ICIN '2001)*, Bordeaux, France.

[8] Costa J. and Tang H., (2002), Application of Spatial Location Information to SIP, *ACM Cluster Computing*, 399–410.

[9] Cuellar J., Morris J., Mulligan D., Peterson J. and Polk J., (2004). Geopriv Requirements, RFC 3693, *IETF*.

[10] Day M., Aggarwal S., Mohr G. and Vincent J., (2000). Instant Messaging / Presence Protocol Requirements, RFC 2779, *IETF*.

[11] Day, M., Rosenberg, J. and Sugano H., (2000). A Model for Presence and Instant Messaging, RFC 2778, *IETF*.

[12] Handley M., Schulzrinne H. and Rosenberg J., (1999). Session Initiation Protocol, RFC 2543, *IETF*.

[13] Hilbert D. M. and Redmiles D. F., (1998). Agents for Collecting Application Usage Data Over the Internet, *Proc. of the 2nd International conference on Autonomous agents*, Minneapolis, Minnesota, 149-156.

[14] Hindus D., Mainwaring S.D., Leduc N., Hagström A.E., Bayley O. and Casablanca, (2001). Designing Social Communication Devices for the Home, *ACM Human Factors in Computing Systems*, 325-332.

[15] Hong J. and Landay J. A., (2004). An Architecture for Privacy-sensitive Ubiquitous Computing, *Proc. of the 2nd International Conference on Mobile Systems, Applications, and Service (Mobisys '04)*, Boston, MA., 177-189.

[16] Iso T., Kurakake S. and Sugimura T., (2003). Image Capture by Cell phone Camera", *IEEE International Conference on Image Processing*, 3, 341-342.

[17] Kanamaru A., and Yoshitsugu, (2004). Fieldcast2: Flexible P2P architecture for presence information sharing, *Proc. of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW)*, 98-102.

[18] Lederer S. and Hong J., (2004). Personal privacy through understanding and action: five pitfalls for designers, *ACM Personal and Ubiquitous Computing*, 8, 440–454.

[19] Munoz M. A., Rodriguez M., Favela J., Martinez-Garcia A. I. and Gonzalez V. M., (2003). Context-Aware Mobile Communication in Hospitals, *IEEE Computer*, 36, 38-46.

[20] Niemi A., (2004). Session Initiation Protocol (SIP) Extension for Event State Publication, RFC3903, *IETF*.

[21] Patton M. Q., (1990). Qualitative evaluation and research methods (2nd ed.), *London: Sage publishers*, ISBN 0803937792.

[22] Pospischil G., Stadler J. and Miladinovic I., (2001). A location-based push architecture using SIP, *Proc. of the 4th International Symposium on Wireless Personal Multimedia Communications (WPMC 2001)*, Aalborg, Denmark, 295-300.

[23] Priyantha N.B., Chakraborty, A. and Balakrishnan H., (2000). The Cricket Location-Support System, *Proc. of the 6th Annual International Conference on Mobile Computing and Networking*, Boston, Massachusetts, 32-43.

[24] Rao B. and Minakakis L., (2004). Assessing the Business Impact of Location Based Services, *Proc. of the 37th Hawaii International Conference on System Sciences (HICSS'04)*, 184 -192.

[25] Reddy M. and Dourish P., (2002). A Finger on the Pulse: Temporal Rhythms and Information Seeking in Medical Work, *Proc. of ACM Conference Computer-Supported Cooperative Work*, New Orleans, Louisiana, 344-353.

[26] Roach A. B, (2002). Session Initiation Protocol (SIP)-Specific Event Notification, RFC 3265, *IETF*.

[27] Rosen B., Costa J., Korkea M., Ylianttila M., Mahy R., Takahashi K. and Farrell S., (2000). Spatial Location Protocol Requirements, Spatial internet draft, *IETF*.

[28] Rosenberg J., (2004). A Presence Event Package for the Session Initiation Protocol (SIP), RFC3856, *IETF*.

[29] Rosenberg J., Schulzrinne H., Camarillo U.G., Johnston A., Sparks J. R., Handley M. and Schooler E., (2002). Session Initiation Protocol, RFC 3261, *IETF*.

[30] Rosenberg J., Schulzrinne H., Huitema C. and Gurle D., (2002). Session Initiation Protocol (SIP) Extension for Instant Messaging, RFC 3428, *IETF*.

[31] Shekhar S. and Yoo J. S., (2003). Processing In-Route Nearest Neighbor Queries: A Comparison of Alternative Approaches, *Proc. of the 11th ACM international symposium on Advances in geographic information systems*, 9-16.

[32] Sommerville I., (2000). *Software engineering (6th edition)*, Pearson education publisher, ISBN 0-201-39815-X.

[33] Spinney J., (2003). A Brief History of LBS and How OpenLS Fits into the New Value Chain, *ESRI Location-Based Services, www.mobilein.com/LBS Open LIS History_final.pdf*.

[34] Sugano H., Fujimoto S., Klyne, G., Bateman A., Carr W. and Peterson J., (2004) Presence Information Data Format (PIDF), RFC 3863, *IETF*.

[35] Want R., Hopper A., Falcao V. and Gibbons J., (1992). The Active Badge Location System, *ACM Trans. Information Systems*, 10, 91–102.

[36] Want R.., Schilit B., Adams N., Gold R., Petersen K., Ellis J., Goldberg D. and Weiser M., (1995). The PARCTAB ubiquitous computing experiment, *Technical Report*, Xerox Palo Alto Research Center.

[37] Weiderman N. H., Habermann A. N., Borger M.W. and Klein M. H., (1987). A methodology for evaluating environments, *Proc. of the 2nd ACM SIGSOFT/SIGPLAN software engineering symposium on practical software development environments*, Palo Alto, California, 199-207.

[38] Westin A.F, (1967), Privacy and Freedom, Bodley head publisher, ISBN 0370013255.

[39] Woodruff A. and Aoki P. M., (2003). How push-to-talk makes talk less pushy, *Proc. of the international conference on supporting group work*, Sanibel Island, Florida, 170–179.

[40] Wu Y., Radovanovic A. and Tucker W. D. (2005). SIP Presence Location Service. *Proc. South African Telecommunications Networks & Applications Conference, (SATNAC 2005)*, Drakensberg, South Africa, 1, 371-378.