SIP-Based Content Development for Wireless Mobile Devices with Delay Constraints

By

Elthea Trevolee Lakay

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

Supervisor: Prof Johnson I. Agbinya

January 2006

KEYWORDS

Session Initiation Protocol (SIP)

Mobile Devices

Wireless Networks

Internet Protocol (IP)

Messaging

SIP Services

Voice over IP (VoIP)

Delay

Security

ABSTRACT

SIP-Based Content Development for Wireless Mobile Devices with Delay Constraints

ET Lakay

MSc Thesis, Department of Computer Science, University of the Western Cape

SIP is receiving much attention these days and it seems to be the most promising candidate as a signaling protocol for the current and future IP telephony services. Realizing this, there is the obvious need to provide a certain level of quality comparable to the traditional telephone service signalling system. Thus, we identified the major costs of SIP, which were found to be delay and security. This thesis discusses the costs of SIP, the solutions for the major costs, and the development of a low cost SIP application. The literature review of the components used to develop such a service is discussed, the networks in which the SIP is used are outlined, and some SIP applications and services previously designed are discussed. A simulation environment is then designed and implemented for the instant messaging service for wireless devices. This environment simulates the average delay in LAN and WLAN in different scenarios, to analyze in which scenario the system has the lowest costs and delay constraints.

January 2006

DECLARATION

I declare that *SIP-Based Content Development for Wireless Mobile Devices with Delay Constraints* is my own work, that it has not been submitted for any degree or examination in any other university, and that all the sources I have used or quoted have been indicated and acknowledged by complete references.

Elthea Trevolee Lakay January 2006

Signed: ………………………….

ACKNOWLEDGEMENTS

I want to thank the one who gave me the wisdom, guidance and strength to finish this dissertation - GOD ALMIGHTY. You are my hero.

To my supervisor who guided me from nowhere to the end where the light shines so brightly. Thank you for encouraging me when I thought I would not make it and believing in me. Thank you and may God bless you with even more wisdom and knowledge.

I want to thank my sponsor, Telkom, for helping me financially. Without you, this would not have been possible.

Thanks to the staff and fellow students of the Department of Computer Science at the University of the Western Cape. You are one great team.

Lastly, I want to give hugs and kisses to my family and friends. No words can describe how much you mean to me and how your encouragement led me to the end. A big thank you to you all. Love you lots and lots.

CONTENTS

LIST OF FIGURES AND GRAPHS

LIST OF TABLES

ACRONYMS

.NET:	Microsoft XML Web Services platform
AES:	Advanced Encryption Standard
API:	Application Programming Interface
ASD:	Answer-Signal Delay
ASNM:	Asynchronous Subscribe and Notification Model
AUXL:	Auxiliary Link
BLER:	Block Error Rate
CDC:	Connected Device Configuration
CLDC:	Connected Limited Device Configuration
CMS:	Communications System
CN:	Correspondent Node
CRD:	Call-Release Delay
DAD:	Duplicate Address Detection
DES:	Data Encryption Standard
DHCP:	Dynamic Host Configuration Protocol
GPRS:	General Packet Radio Service
HTTP:	Hypertext Transfer Protocol
ID:	Identification Document
IETF:	Internet Engineering Task Force
IM:	Instant Messaging
IMS:	IP Multimedia Subsystem
IP:	Internet Protocol
IPSec:	Secure Internet Protocol (IETF)
IPSL:	Inter-Proxy-Server Link
IPv6:	Internet Protocol Version 6
J2ME:	Java 2 Platform Micro Edition
JAD:	Joint Application Development
JAR:	Java Archive
JSR:	Java Specification Request
LAN:	Local Area Network

LST:	Laplace-Stieltjes Transform
MA:	Mobile Agent
MEGACO:	Media Gateway Control Protocol
MGC:	Media Gateway Controller
MID:	Mobile Information Devices
MIDP:	Mobile Information Device Profile
MIME:	Multipurpose Internet Mail Extensions
MIPv6:	Mobile Internet Protocol Version 6
MMUSIC:	Multi party Multimedia Session Control
MN:	Mobile Node
MoIP:	Multimedia over IP
OTA:	Over the Air
PC:	Personal Computer
PDA:	Personal Digital Assistant
PDD:	Post-Dialling Delay
PDU:	Protocol Data Unit
PIM:	Personal Information Management
PP:	Personal Profile
PSTN:	Public Switched Telephone Network
QoS:	Quality of Service
RA:	Router Advertisement
RFC:	Request for Comment
RTP:	Real-Time Transport Protocol
RTSP:	Real-Time Streaming Protocol
S/MIME:	Secure/Multipurpose Internet Mail Extensions (IETF)
SDP:	Session Description Protocol
SIM:	Synchronous Invite Model
SIP:	Session Initiation Protocol
SIPC:	Simply Interactive Personal Computer
SIP-T:	Session Initiation Protocol for Telephony
SMS:	Short Message Service
SMTP:	Simple Mail Transport Protocol

SS7:	Signal System 7
SSH:	Secure Socket Shell
S-UMTS:	Satellite-UMTS
TCP:	Transmission Control Protocol
TLS:	Transport Layer Security
TU:	Transaction User
UA:	User Agent
UAC:	User Agent Client
UAS:	User Agent Server
UC:	Ubiquitous Computing
UDP:	Universal Datagram Protocol
UMTS:	Universal Mobile Telecommunication
URI:	Uniform Resource Identifier
URI:	Uniform Resource Identifier
VM:	Virtual Machine
VoIP:	Voice over IP
WAP:	Wireless Application Protocol
WLAN:	Wireless Network
WTK:	Wireless Toolkit

Chapter 1: Introduction

This thesis addresses the problem of creating quality, low communication cost, and fast wireless SIP applications for mobile devices with delay constraints. In order to do so several steps needs to be taken. Firstly, we need to study SIP, analyze the costs of SIP and then we need to find out how to develop a SIP service with cost constraints.

SIP are currently more and more used, and seems to be the most feasible signaling protocol for current and future IP telephony services, which is overpowering the plain old telephone services rapidly. This brings forward the realization of an obvious need to provide a high level of quality and security, comparing to the traditional telephone service signaling system 7 (SS7). After the desired levels of quality and security are obtained, developing SIP services will be so much more valuable, whereby more and more users will be comfortable switching from PSTN to SIP.

We studied SIP performance focusing on the cost of using SIP signalling on wireless networks and devices. From the study, security and delay/disruption time were identified as the most severe problems in SIP signalling. From a software engineering perspective, quality, security, and user satisfaction (low delay/disruption time), are the highest priority. Thus if these factors are satisfactory to the user, the system is regarded as a very well designed and quality system.

1.1 *Constraint and Motivation*

1.1.1 Focus and Constraint

The focus of the project is to develop a wireless SIP service that compares to the level of delay and security of PSTN or even better for mobile phones. There are many sets of tools and libraries that developers can use to create enhanced applications for smart mobile devices like, PDAs and handsets. The focus, however, will increasingly be on two main approaches:

1. The .NET Compact Framework, which currently targets Microsoft Pocket PCs and devices powered by Windows CE .NET, and
2. J2ME, which is not limited to Microsoft operating systems.

This thesis focuses on the J2ME approach for developing SIP applications for mobile phones.

The objective of this thesis is to research and implement SIP applications for mobile devices that take care of the two most severe constraints, viz. security, and delay. This thesis introduces the SIP Instant Messaging client for the mobile phone whereby the knowledge of the research in Chapter 3 is used to create a secure and low delay SIP application.

The following security characteristics should be guaranteed:

1. High service availability
2. Stable and error-free operation
3. Protection of the user-to-network and user-to-user network traffic.

The following delay characteristics should be limited:

1. Queuing Delay
2. Call Setup Delay
3. Message Transfer Delay
4. Session Setup Delay
5. Handoff Delay

1.1.2 Motivation

In an era where the number of users of mobile devices is increasing by the minute, the demand for faster and easier use of services will also increase. Thus in the wireless community, there is the need for fast, easy, small and cost-effective services for wireless Internet on handheld devices.

Many of the major obstacles are addressed by SIP in the Internet telephony development. It is predicted that Internet telephony will change the way businesses and people talk to each other. However, SIP is much more than VoIP. SIP enables service providers to create their own services out of voice, data, and video. Motivating this, it means that SIP is a competent protocol to be used to develop services with costs constraints.

SIP is an easy to use protocol, meaning you only specify what you need to specify. Thus, it is a simple toolkit that service providers can use to build converged voice and multi-media services. SIP offers faster feature development than some other protocols, like H.323. This is due to SIP's ease of development, openness, lightweight design, and total integration with WEB technology.

Knowing the strengths and weaknesses of SIP, we can create reliable, secure, low delay and quality SIP applications for wireless devices, which comply with the users demand.

1.2 Research Questions and Problems

The main research question is "How to structure and build reliable, secure, low delay and quality SIP applications, that might eliminate most of the compatible protocols of SIP"?

Other research questions are outlined as follows:

- What are the most severe SIP security issues?
- How can the most severe SIP security issues be overcome?
- What are the most serve SIP delay issues?
- How can the most severe SIP delay issues be overcome?
- How must the SIP applications be structured to overcome the delay and security issues?

1.3 *Approach and Methodology*

The Incremental Evolutionary Software Model will be used to develop the low cost SIP service. In conjunction with the incremental evolutionary software model, two software specifications will be used, namely Connection Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP). Simulation will be based on delay for the designed service.

A study on the cost of SIP and SIP signaling will be undertaken. Using the knowledge of this study, a user-interface simulation for the mobile phone will be developed and applied, by using J2ME.

Evaluations will be done in different networks like, LAN, Wireless Network (WLAN) and combining LAN and WLAN to determine if the SIP applications are secure and the delay is low. The case study and the software development methods are the main methodologies used in the research. Simulations will be based on delay constraints.

This research expects the delay to be greater during peak hours than during off-peak hours. This is because more people are on the network during peak hours than off-peak hours. This thesis also expects the delay to be less in LAN than in WLAN, since over the air (OTA) analysis is used when testing in a WLAN, where the message is more prone to data loss and delay. This research's focus will however be more on WLAN testing to develop a wireless SIP service with delay constraints.

1.4 *Thesis Layout*

1.4.1 Chapter 2

Chapter 2 is the literature review chapter. It describes and explains the main protocol, SIP, used for this project, the networks in which SIP is used, the problems within the different networks where SIP is used, some SIP services and applications and we outline some SIP on-going projects.

1.4.2 Chapter 3

In Chapter 3, the cost of SIP services is outlined. The main costs of SIP were found to be security and delay. These costs are described and discussed in detail including the solutions found and what still needs to be solved within these two main cost areas. This chapter also outlines the known attacks on SIP and what methodologies are used to take care of them.

1.4.3 Chapter 4

In Chapter 4, the knowledge gained in Chapter 2 and Chapter 3 is used to design and implement a low cost SIP service. In this chapter, Chapter 4, the tools, and libraries are given and described in developing a low cost SIP service. In our case, we focused on developing an instant messaging (IM) client. After a clear description of the tools and libraries used, we show the actual design and its implementation.

1.4.4 Chapter 5

Chapter 5 outlines how the SIP IM service experiments were performed including the results. The SIP IM client was tested in a LAN and WLAN, each having different test scenarios. In each scenario, the delay during peak hours and off-peak hours was calculated and graphically presented.

1.4.5 Chapter 6

This chapter, Chapter 6, is the concluding and future work chapter. The chapter outlines if the entire research questions could be answered and if the focus and constraints could be achieved.

Chapter 2: Literature Review

2.1 Introduction

In this chapter a literature review of SIP is given. The different networks where SIP can be used are discussed, as well as the problems in SIP networks and some of the solutions that were found. Since this project is about creating SIP applications with delay constraint, a few SIP applications are also discussed. Some on-going SIP projects, related to this research are then reviewed.

2.2 Session Initiation Protocol (SIP)

2.2.1 A Historical Snapshot

SIP emerged in the mid 1990s from the research of Henning Schulzrinne [19] and his research team at Columbia University, USA. Professor Schulzrinne is the co-author of the Real-Time Transport Protocol (RTP) [10], and co-wrote the Real-Time Streaming Protocol (RTSP) [11]. RTP transmits data over the Internet in real-time [10]. RTSP is a proposed standard for controlling streaming audio-visual content over the Web [11].

Schulzrinne's intent was to define a standard for Multi-party Multi-media Session Control (MMUSIC). SIP evolved from 1996, by being a key element in a submitted draft to the IETF to becoming a SIP specification RFC 3261 [1] in 2001 by IETF. The first specification RFC 2543 [13] was issued by IETF in 1999, whereby Schulzrinne removed extraneous components regarding media content [19].

The advent of RFC 3261 signalled that the fundamentals of SIP were in place. Since then, enhancements to security and authentication, amongst other areas, have been issued in several additional RFCs. RFC 3262 [14], for example, governs Reliability and Provisional Responses. RFC 3263 [15] establishes rules to locate SIP Proxy Servers while RFC 3264 [16] provides an offer/answer model and RFC 3265 [17] determines specific event notification.

Vendors began to launch SIP-based services in 2001 and today the enthusiasm is growing more and more for the protocol. Sun Microsystems' Java community Process define application programme interfaces (APIs) using Java for building SIP components and applications for service providers and enterprises. SIP is on its way to become one of the greatest protocols since HTTP and SMTP, since the number of SIP players is entering the market-place with promising new services.

2.2.2 Overview of SIP

"SIP is an application-layer control and mobility support protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls" [20]. SIP supports variety capability features, like:

- Inviting participants to existing sessions
- Adding and removing media from a session
- Supports personal mobility
    - Transparently supports name mapping and re-directing of services
    - Users can maintain a single visible identification regardless of network location.

SIP supports five facets of establishing and terminating multi-media communications:

- **User location:** Determination of the end system to be used for communication;

- **User availability:** Check of the willingness of the called party to engage in communications;

- **User capabilities:** Discovering of the media and media parameters to be used;

- **Session setup:** "Ringing", establishment of session parameters at both called and calling party;

- **Session management:** This includes transfer and termination of sessions, modifying session parameters, and invoking services.

SIP can be used with other IETF protocols to build a complete multimedia architecture, but the basic functionality and operation of SIP does not depend on any of these protocols. These architectures include protocols such as:

- RTP
- RTSP
- Media Gateway Control Protocol (MEGACO) (RTF 3015 [12]) for controlling gateways to the Public Switched Telephone Network (PSTN); and
- Session Description Protocol (SDP) (RFC 2327 [3]) for describing multimedia sessions.

Rather than SIP providing services, it provides primitives used to implement different services. For example: SIP locates users and delivers objects to a location, but these primitives can be used to implement the ‘caller ID’ service, meaning using the primitives to deliver the caller’s photo and session description. Since SIP does not provide network resource reservation capability, its messages and sessions established can pass through different networks.

The nature of the services provided make security particularly important. SIP provides a suite of security services, which include denial-of-service prevention, authentication (both user to user and proxy to user), integrity protection, and encryption and privacy services, which is discussed in Chapter 3.

2.2.3 Structure of SIP

SIP is structured as a layered protocol, as shown in Figure 1, which means that its behaviour is described in terms of a set of fairly independent processing stages with only a loose coupling between each stage. Each layer has a set of rules that needs to be complied with, for an element to contain a layer. Thus not all layers are contained in every element. Each SIP element is logical and not physical.

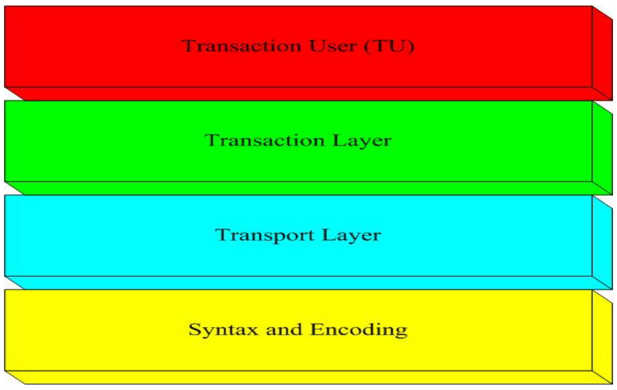


Figure 1: SIP Structure

The lowest layer of SIP is its syntax and encoding (see Figure 1). Its encoding is specified using an augmented Backus-Naur Form grammar.

The second layer in Figure 1 is the transport layer, which defines how a client and server send requests and receives responses over the network. All SIP elements contain a transport layer.

The transaction layer is the third layer, as shown in Figure 1. Transactions are a fundamental component of SIP. A request sent by a client transaction, by using the transport layer, to a server transaction is called a transaction [1]. This goes along with the responses during the request sent from the server to the client. The transaction layer handles various situations, like:

- application-layer re-transmissions,
- matching of responses to requests, and
- Application-layer timeouts.

User agents (UA) contain a transaction layer, as do stateful proxies. Stateful proxies are proxies that maintain state information [22]. Stateless proxies do not contain a transaction layer. Stateless proxies are proxies that do not keep any state information [22]. The transaction layer has a client and a server component, each of which are represented by a finite state machine that is constructed to process a particular request [1].

The layer above the transaction layer the fourth layer in Figure 1 is called the transaction user (TU). Each of the SIP entities is a transaction user, except the stateless proxy. When a TU wishes to send a request, it creates a client transaction instance and passes the request along with the destination IP address, port, and transport to which to send the request. A TU can create and cancel a client transaction.
The SIP elements – user-agent-clients and servers, stateless and stateful proxies, and registers – contain cores that differentiate from each other. Only the stateless proxy does not include the TU layer.

Register and invite are very essential in SIP. Since registrations play an important role in SIP, a user agent server (UAS) that handles a register is given the special name ‘registrar’ [1]. The most important method in SIP is the ‘invite’ method, which is used to establish a session between participants [1]. A session is a collection of participants, and streams of media between them, for the purposes of communication [1].

2.2.3.1 Characteristics

The characteristics of SIP lies in its ability to create, modify, and terminate sessions with a single or multiple participants. Some of the sessions well known and used these days are: Internet multimedia conference, Internet/IP Network telephone call, and multimedia distribution. Participants in a session communicate through multicast or mesh of unicast or a combination of the two.

Mobility is a big demand in today's life, and SIP support user mobility by proxying and redirecting request to wherever the user are situated.

2.2.3.1.1 SIP enabled functions

SIP has a lot of functions that raise it quality and characteristics standards. Here are some of the important functions:

- Name translation and user location ensure that the call reaches the called party wherever they are located and carries out any mapping of descriptive location information. It ensures that the details of nature of the call/session are supported.

- Feature negotiation allows the group involved in a call to agree on the features supported. This is to ensure that the participants support the same level of features.

- Call Participant management feature allows a participant can bring other users onto the call or cancel connections to other users, during a call. Users could also be transferred or placed on hold. This ensures that the user has complete control over whom and when he/she wants to talk to whom.

- The call feature changes feature enables a user to change the call characteristics during the course of the call.

2.3 SIP Networks and Services

Creation of SIP services depends on the network it will be used in. In this section a discussion is given on the different networks SIP can be used in, some problems within the different networks and some well known application used in the different networks.

2.3.1 SIP for Wireless Networks

Earlier in the thesis a brief mention was made on the importance of mobility and service provision. SIP within wireless networks enable to provide such mobility to users. Within [21] it is found that SIP was designed to support and is the key to high-value, high-margin wireless services, which are about peer-to-peer communication. SIP can run on any wireless network.

SIP is used in packet-based IP communications networks - so in the mobile field it will come into play with the launch of new high-speed 3G systems and through existing technologies such as wireless local area network (LAN) and GPRS (General Packet Radio Service).

Tests for the developed application will be done in this network.

2.3.2 SIP for Fixed-line Networks

The majority of SIP deployments to date have been in fixed-line IP networks. Over the past several years it has become clear that SIP is the winner over legacy protocols such as H.323. This is due to SIP's ease of application development, openness, lightweight design and total integration with Web technologies [21]. VoIP technologists agree that all significant VoIP network deployments going forward will be based on SIP.

Tests will be done in this network scenario as well as a combination of fixed-line networks and wireless networks.

2.3.3 SIP Mobility

SIP supports personal mobility, i.e., a user can be found independent of user location and network device (PC, laptop, IP phone, etc). The step from personal mobility to IP mobility support is basically the roaming frequency, and that a user can change location (IP address) during a traffic flow [20]. Therefore, in order to support IP mobility, we need to add the ability to move while a session is active. It is assumed that the mobile host belongs to a home network, on which there is a SIP server which receives registrations from the mobile host each time it changes location. This is similar to home agent registration in Mobile IP. The host does not need to have a statically allocated IP address on the home network.

When the correspondent host sends an ‘invite’ to the mobile host, the redirect server has current information about the mobile host’s location and redirects the invite there (see Figure 2).

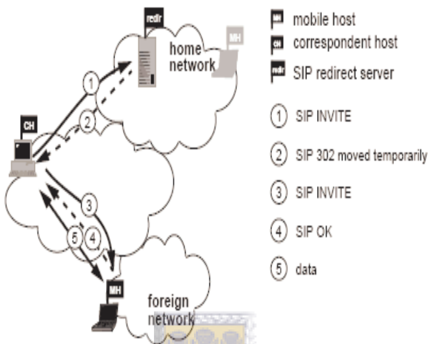


Figure 2: SIP Mobility - setting up a call [20]

If the mobile host moves during a session, it must send a new ‘invite’ to the original call setup, as shown in Figure 3. It should put the new IP address in the contact field of the SIP message, which tells the correspondent host where it wants to receive future SIP messages.

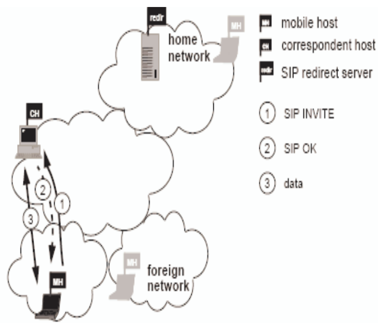


Figure 3: SIP Mobility: mobile host moves [20]

No tests will be done in SIP mobility, but it could be done in further development.

2.3.3.1 Error Recovery

If the correspondent host for some reason has an outdated address of the mobile host, it must have a fall-back mechanism to break the error situation. In order to avoid situations like this, a host can send retransmissions of invitations also to the SIP server on the mobile host's home network. Since the SIP server has a fixed address, the mobile host can always send registrations to it. In this fashion, the correspondent host can re-locate a mobile host that has been lost.

2.3.3.2 Security

In the SIP specification there is support for both authentication and encryption of SIP messages, using either challenge-response or private/public key cryptography. This is discussed in Chapter 3, Section 2.

2.4 Problems with SIP in a Network

SIP can be used in any network, but for the purpose of this project, the focus will be on local area networks (LAN) and wireless networks (WLAN). Some of these SIP networks are discussed in Section 2.3.

Since IP is receiving much attention and it seems to be the most feasible candidate as a signalling protocol for the current and future IP telephony services, it is worth knowing and even solving the problems that still persist in SIP networks. In Chapter 3 it is shown what the most severe problems found in SIP services, i.e. delay and security. In this section a discussion is given on the problems within the SIP networks. This need to be known, since this project, relies on it for testing.

The common problems of SIP in different networks are security, delay, packet loss and jitter. Undesirable delay causes packet loss, which is detrimental to applications such as VoIP or streaming video with stringent quality of service (QoS) requirements [28].

2.4.1 Security

SIP provides an elegant application layer mobility support that solves the problems associated with lower layer mobility protocols in next generation heterogeneous wireless access networks [26]. Since sessions can be between domains, securing the process will involve inter-realm authentication and authorisation, which gives rise to most issues such as user privacy and authorisation granularity.

The solution of Sicker et al [26] for the security problem was based on three modular functions:

1. Resource Registration – allows a user to register within local domain.
2. Resource Discovery – allows a user to locate other users from within the same domain as well as other domains.
3. Call Initiation – allows a user to setup a session with another user.

Each of these modular functions is independent of each other. They are protected separately. Their goal is to make use of practical security functions while providing a robust level of privacy to the end user.

Rao and Li [27] solved the security and privacy issue by securing the ubiquitous computing (UC) system and secondly by securing the user.

1. Securing the UC system – protects the UC system from unauthorised access and ensures the possibility of tracking down “who does the evil” [27].
2. Secure the user – UC systems based on similar designs will not be accepted for wide deployment, as almost every user action might be captured by the system.

2.4.2 Delay, Jitter and Packet Loss

Banerjee et al [28] proposed a SIP-based architecture that supports soft handoff for IP centric wireless networks alleviating the problem of packet loss. Their architecture ensures that there is no packet loss and the end-to-end delay jitter is kept under control, thus maintaining two important parameters dictating the QoS for streaming multi-media applications.

Although SIP-based mobility management solves the problem posed by Mobile IP route optimization, in some cases it introduces unacceptable handoff delays for multi-media applications with stringent QoS requirements. Moreover, SIP entails application layer processing of the messages which may introduce additional delay [28]. To alleviate this problem Banerjee et al [28] proposed SIP-based soft handoff mobility architecture for next generation wireless networks. A testbed was setup and the results showed that the architecture is capable of ensuring zero packet loss and controlled delay and jitter.

Although security, delay, jitter and packet loss problems exist in SIP networks, most of them have been solved or are in the process of being solved. Thus SIP provides an elegant application layer mobility support that solves the problems associated with lower layer mobility protocols in next generation heterogeneous wireless access networks and other networks for that matter.

2.5 SIP-Based Applications

In this section a discussion will be given on the different applications in which SIP has been used. From the overview section, in Section 2.1.2, it is seen that SIP supports name mapping and redirection services, which support personal mobility. Since SIP can be used with other IETF protocols, a complete multi-media architecture can be built out of SIP. It needs to be made clear that SIP does not provide services, but provides primitives that can be used to implement different services.

From the characteristics section, in Section 2.1.3.1, it is seen that SIP can create, modify and terminate sessions. This statement is part of SIP’s definition. These sessions include Internet multi-media conferences, Internet or any IP Network telephone calls and multi-media distribution.

2.5.1 Mobile Agent Communication Using SIP

An investigation into communication issues was done by Tsai and few in [29] between mobile agents (MA) and a reliable communication protocol was proposed which integrates SIP as a signalling protocol. They presented a synchronous Invite model (SIM) and asynchronous Subscribe & Notify model (ASNM) to establish a framework for inter-agent communication. “The two models can solve communication failure and message chasing problems” [29]. The communication cost and utilization rates were analyzed. In the Zhou et al [30], Tsai and few [29] found two problems in the mobile agent communication, which are communication failure and message chasing:

- **Communication failure** is defined as a message delivered to the host that the receiver agent stays; receiver agent has migrated to the next host [29]. Thus the receiver agent receives unsuccessful messages. It is not guaranteed that the receiver host can get all messages sent to it, since messages cannot be delivered instantly, even in an error free network. This is a common existing problem and is not yet solved in mobile agent systems.
- **Message Chasing** is defined as when a receiver host migrates frequently and each time a message reaches the host, receiver host has moved and advanced to the next host, resulting in a race condition. This may corrupt the collaboration of mobile agent systems.

In this research Tsai and few [29] propose a reliable communication protocol that can solve the above-mentioned problems. This protocol integrates SIP to locate agents and deliver messages in order to improve communication fault-rate and thus enhanced system reliability. SIP specifies the architecture of user agents and servers to support agent tracking, call routing and call redirection. SIM and ASNM were used to solve communication failure and message chasing problems. SIP handles normal inter-agent communication [29]. ASNM is invoked as agents are moving and messages are forwarded indirectly and distributive.

The communication cost of SIM is calculated as follows -

$$Cost(SIM) = T_{inv} + T_{msg} \text{ [29] } \ldots\ldots\ldots\ldots\ldots\ldots.(1)$$

and ASNM communication cost is calculated as -

$Cost(ASNM) = T_{inv} + T_{sub} + T_{mig} + T_{mig}(R) + T_{not} + T_{msg}$ [29] ……………(2)

Hereby -

T_{inv} – the duration from the moment an agent sends an 'invite' to the moment it receives either moved temporarily or temporarily unavailable.

T_{msg} – the duration from the moment an agent sends a message to the receiver to the moment it gets an OK response.

T_{not} – the time needed to send 'notify' to an agent from the home server

T_{sub} – the duration from the moment an agent sends 'subscribe' to home server to the moment it gets an OK.

T_{mig} – the moment an agent leaves a node to the moment it reaches another node.

The two models (1) and (2) integrate SIP to exploit its signalling feature.

2.5.2 An Advanced Architecture for Push Services

Push technology is going to play a relevant role in the mobile scenario since it allows for new appealing services which foresee the distribution of information useful to mobile users like weather forecasts, traffic news, and sport news or simply advertising. Tosi [31] proposes new reference architecture to support advanced Push services, based on presence, location, instant messaging and build-over SIP.

IM is one of the possible outstanding applications for mobile data services, while Presence and Push are considered as very interesting enabling services. Tosi [31] uses Presence concept and IM solution, which is based on the SIP protocol and is integrated with wireless application protocol (WAP) architecture. The proposed architecture relies on various standardized technologies and is built on top of different advanced services as Presence, IM user profile and terminal capabilities.

The proposed architecture has been designed to fit into IP multi-media subsystem (IMS) architecture, since in this respect Push is viewed as an enabling service.

2.5.3 Context-Aware Communication Application

Pervasive computing applications must be engineered to provide unprecedented levels of flexibility to reconfigure and adapt in response to changes in computing resources and user requirements. In the project of McFadden et al [32], they demonstrate the use of a disciplined, model-based approach to engineer a context-aware, SIP-based communication application.

This application [32] exploits a rich set of contextual information in order to self-configure and adapt, including the location of people and devices, activity types, device network status and power. In addition, it relies on system policies and user preference information to evaluate the suitability of available communication channels for the current context of use. Policies and preferences can be modified on-the-fly, in order to support highly flexible behaviour and accommodate an evolving set of system resources and user requirements.

The application in [32] leverages context and preference information to allow for seamless communication between individuals. It relies on context and preference evaluation for both call initiation and mid-stream call transfer in response to context change. It also controls communications devices using the SIP protocol

The cause for the problem mentioned in [32] is that users' preference sets typically capture many different requirements, and although our preference model makes it relatively easy to specify each of these in isolation, the outcome of combining them is not always intuitive. "Adjustment of the relative weights assigned to the individual preference is sometimes needed to bring the device selection in line with the user's expectation" [32]. Currently, McFadden [32] perform a manual adjustment, but anticipates that learning algorithms could be used to exploit user feedback to automatically evolve the preference set.

In the next section a discussion of SIP projects is given.

2.6 On-Going SIP Projects

In sections 2.2 to 2.4 discussions have been given on SIP: what it is, the different networks in which SIP is used and the services of SIP. In this section a discussion is given on how and where SIP is used, through reference to different projects. In the Internet world, SIP is one of the most importances for advanced telephony services [23].

2.6.1 Ubiquitous Computing Using SIP

In the past decade, the goal of ubiquitous computing has been pursued in a large number of prototypes, making computational resources or communication more widely available [24]. This project took ubiquitous computing a step further by moving from “special-purpose, one-of-a-kind systems to more widely deployable systems that scale to the global Internet” [24]. The project developed a securable, capable of being administered by multiple, independent administrators and operators, and integrated off-the-shelf hardware and software systems. The core tenets of computing included: multi-media, device integration, event-based, location-aware, privacy-conscious and invisibility to user computing.

This system was designed to support a range of activities, from home-based settings to collaboration between distant sites. It was designed so that participation only requires standard SIP-speaking tools such as SIPC or Microsoft Windows Messenger or even a basic handheld device or landline phone. The system derives its scalability from the underlying SIP architecture, which was discussed in Section 2.3.1.

2.6.2 Service Deployment and Development in H.323 and SIP

Comparisons at the system level between SIP and H.323 have been studied and published, but this project compares the methods to implement services in SIP and H.323 focusing in the service architectures and their capabilities regarding the implementation of new features.

Glasmann et al compared the H.323 and SIP according to standardisation status, supporting services, supplementary service architecture, proprietary extension and negotiation mechanisms, inter-operability of services and features, inter-working with GSTN, and service creation issues.

Glasmann et al [23] found that H.323 provides better functionality, inter-operability and inter-networking with respect to supplementary services. SIP has advantages with respect to the design of low cost terminals. Glasmann et al [23] found that even though the protocols were approaching each other, their focus and applicability is different and it can be expected that neither of the two will dominate over the other.

2.6.3 Application-Layer Mobility Using SIP

"Supporting mobile Internet multi-media application requires more than just the ability to maintain connectivity across subnet changes" [25]. Schulzrinne and Wedlund [25] describe how SIP can help provide terminal, personal, session and service mobility to applications ranging from Internet telephony to presence and instant messaging. A lot of effort has been made over the years to allow computers and communication devices to continue communication, even when mobile.

System design should make it easier for devices to move between different wireless networks, depending on population density, speed of movement and propagation characteristics. Schulzrinne and Wedlund [25] describe a possible architecture that allows the supporting of a full range of mobility options, independent of the underlying technology. They primarily focused on the provision of multi-media services, but allude briefly to "data" services. They also explored how application-layer support is necessary to offer more than just handoff between base stations and subnets, as well as how it can, under certain circumstances, compensate for the lack of deployment of mobile IP. SIP was used at the heart of the effort as the protocol.

Schulzrinne and Wedlund [25] concluded that application-layer mobility can either partially replace or complement network-layer mobility. They tried to show that for inter-active sessions, SIP-based mobility can be used to provide all common forms of mobility, including terminal, personal and service mobility. They found that for terminal mobility, an IPv6-based solution is preferable, as it applies to all IP-based applications, rather than just Internet telephony and conferencing. When the home agents are absent, SIP-based mobility can provide mobility services to the most important current application telephony systems.

2.7 Summary

This chapter, Chapter 2, gave a literature review on SIP and its various components. The Session Initiation Protocol is a signalling protocol used for establishing sessions in an IP network. The different networks, fixed line, wireless and mobile, in which SIP can be used, are discussed, since the developed SIP system will be tested within these different network platforms. The problems in the SIP networks are discussed. It was found that security, delay, jitter and packet loss were some of these problems. A discussion was also given on the various applications of SIP. Some of the on-going projects of SIP were also discussed.

A study was done on the cost of SIP services, since one of the aims of this project is to develop low cost SIP applications. In the next chapter, Chapter 3, the cost of SIP on various IP services is discussed.

Chapter 3: Cost of SIP on Various IP Services

3.1 Introduction

IP is currently receiving much attention and seems to be the most promising candidate as signalling protocol for current and future IP telephony services, and which is becoming a real competitor to the plain old telephone service. For the realization of such a scenario, there is the obvious need to provide a certain level of quality and security, comparable to that provided by the traditional telephone system.

This thesis studied the major costs of SIP and SIP signalling. From this study, security and delay/disruption time was found to be the most severe problems in SIP signalling. In this chapter these costs are discussed.

3.2 Security

SIP is not an easy protocol to secure. Its use of intermediaries, its multi-faceted trust relationships, its expected usage between elements with no trust at all, and its user-to-user operation make security far from trivial. Security solutions are needed that are deployable today, without extensive coordination, in a wide variety of environments and usages. In order to meet these diverse needs, several distinct mechanisms applicable to different aspects and usages of SIP will be required.

Although security and privacy should be mandatory for IP telephony architecture, most of the attention during the initial design of the IETF IP telephony architecture and its signalling protocol (SIP) has been focused on the possibility of providing new dynamic and powerful services, and simplicity. Less attention has been paid to security features.

If the new IP telephone architecture and SIP is to replace PSTN, it should provide the same basic telephony service with a comparable level of QoS (Quality of Service) and network security.

The following security characteristics should be guaranteed:

1. High service availability
2. Stable and error-free operation
3. Protection of the user-to-network and user-to-user network traffic.

SIP messages may contain information a user or server wishes to keep private. The headers can reveal information about the communication patterns and content of individuals, or other confidential information. The SIP message body may also contain user information (media type, codec, address and ports, etc) that should not be revealed.

Securing SIP header and body information can be motivated for two different reasons:

1. Maintain private user and network information in order to guarantee a certain level of privacy
2. Avoiding SIP sessions being set up or charged by someone faking the identity of someone else.

The mechanisms that provide security in SIP can be classified as end-to-end or hop-by-hop protection. The end-to-end mechanism involves the caller and/or called SIP user agents and is realized by features of the SIP protocol specifically designed for this purpose (e.g. SIP authentication and SIP message body encryption). Hop-by-hop mechanisms secure the communication between two successive SIP entities in the path of signalling messages. SIP does not provide specific features for hop-by-hop protection and relies on network-level or transport-level security. Hop-by-hop mechanisms are needed because intermediate elements may play an active role in SIP processing by reading and/or writing some parts of the SIP messages. End-to-end security cannot apply to these parts of messages that are read and written by intermediate SIP entities.

Two main security mechanisms are used with SIP: authentication and data encryption. Data authentication is used to authenticate the sender of the message, and to ensure that some critical message information was unmodified in transit. This is to prevent an attacker from modifying and/or replaying SIP requests and responses. Data encryption is used to ensure confidentiality of SIP communications, letting only the intended recipient decrypt and read the data. This is usually done using encryption algorithms such as Data Encryption Standard (DES) and Advanced Encryption Standard (AES).

3.2.1 Security Mechanisms in SIP

The fundamental security services required for the SIP protocol are: preserving the confidentiality and integrity of messaging, preventing replay attacks or message spoofing, providing for the authentication and privacy of the participants in a session, and preventing denial-of-service attacks. Bodies within SIP messages separately require the security services of confidentiality, integrity, and authentication. Rather than defining new security mechanisms specific to SIP, SIP re-uses wherever possible existing security models derived from the HTTP and SMTP space.

Full encryption of messages provides the best means to preserve the confidentiality of signalling - it can also guarantee that any malicious intermediaries do not modify messages. However, SIP requests and responses cannot be naively encrypted end-to-end in their entirety because message fields such as the Request-URI, Route, and via need to be visible to proxies in most network architectures so that SIP requests are routed correctly. Note that proxy servers need to modify some features of messages as well (such as adding via header field values) in order for SIP to function. Proxy servers must therefore be trusted, to some degree, by SIP UAs. For this purpose, low-layer security mechanisms for SIP are recommended, which encrypt the entire SIP requests or responses on the wire on a hop-by-hop basis, and that allow endpoints to verify the identity of proxy servers to whom they send requests.

SIP entities also have a need to identify one another in a secure fashion. When a SIP endpoint asserts the identity of its user to a peer UA or to a proxy server that identity should in some way be verifiable. A cryptographic authentication mechanism is provided in SIP to address this requirement. An independent security mechanism for IP message bodies supplies an alternative means of end-to-end mutual authentication, as well as providing a limit on the degree to which user agents must trust intermediaries.

3.2.1.1 Transport and Network Layer Security

Transport or network layer security encrypts signalling traffic, guaranteeing message confidentiality and integrity. Oftentimes, certificates are used in the establishment of lower-layer security, and these certificates can be used to provide a means of authentication in much architecture.

Two popular alternatives for providing security at the transport and network layer are, respectively, TLS [45] and IPSec [46]. IPSec is a set of network-layer protocol tools that collectively can be used as a secure replacement for traditional IP (Internet Protocol). IPSec is most commonly used in architectures in which a set of hosts or administrative domains have an existing trust relationship with one another. IPSec is usually implemented at the operating system level in a host, or on a security gateway that provides confidentiality and integrity for all traffic it receives from a particular interface (as in a VPN architecture). IPSec can also be used on a hop-by-hop basis. In many architectures IPSec does not require integration with SIP applications; IPSec is perhaps best suited to deployments in which adding security directly to SIP hosts would be arduous. UAs that have a pre-shared keying relationship with their first-hop proxy server are also good candidates for IPSec use. Any deployment of IPSec for SIP would require an IPSec profile describing the protocol tools that would be required to secure SIP.

TLS provides transport-layer security over connection-oriented protocols (for the purposes of this document, TCP); "tls" (signifying TLS over TCP) can be specified as the desired transport protocol within a Via header field value or a SIP-URI. TLS is most suited to architectures in which hop-by-hop security is required between hosts with no pre-existing trust association. For example, Alice trusts her local proxy server, which after a certificate exchange decides to trust Bob’s local proxy server, which Bob trusts, hence Bob and Alice can communicate securely. TLS must be tightly coupled with a SIP application. Note that transport mechanisms are specified on a hop-by-hop basis in SIP, thus a UA that sends requests over TLS to a proxy server has no assurance that TLS will be used end-to-end. The TLS_RSA_WITH_AES_128_CBC_SHA cipher suite [44] must be supported at a minimum by implementers when TLS is used in a SIP application. For purposes of backwards compatibility, proxy servers, redirect servers, and registrars should support TLS_RSA_WITH_3DES_EDE_CBC_SHA [44]. Implementers may also support any other cipher suite.

3.2.1.2 SIP URI Scheme

The SIP URI scheme adheres to the syntax of the SIP URI, although the scheme string is "sips" rather than "sip". The semantics of SIP are very different from the SIP URI, however. SIP allows resources to specify that they should be reached securely.

A SIP URI can be used as an address-of-record for a particular user - the URI by which the user is canonically known (on their business cards, in the ‘From’ header field of their requests, in the ‘To’ header field of register requests). When used as the Request-URI of a request, the SIP scheme signifies that each hop over which the request is forwarded, until the request reaches the SIP entity responsible for the domain portion of the Request-URI, must be secured with TLS. Once it reaches the domain in question, it is handled in accordance with local security and routing policy, quite possibly using TLS for any last hop to a UAS. When used by the originator of a request (as would be the case if they employed a SIP URI as the address-of-record of the target), SIP dictates that the entire request path to the target domain be so secured.

The SIP scheme is applicable to many of the other ways in which SIP Uri's are used in SIP today in addition to the Request-URI i.e. inclusion in addresses-of-record, contact addresses (the contents of Contact headers, including those of register methods), and Route headers. In each instance, the SIP URI scheme allows these existing fields to designate secure resources. The manner in which a SIP URI is dereferences in any of these contexts has its own security properties which are detailed in [43].

The use of SIP in particular entails that mutual TLS authentication should be employed, as should the cipher suite TLS_RSA_WITH_AES_128_CBC_SHA. Certificates received in the authentication process should be validated with root certificates held by the client; failure to validate a certificate should result in the failure of the request.

Note that in the SIP URI scheme, transport is independent of TLS, and thus "sip:alice@atlanta.com;transport=tcp" and "sip:alice@atlanta.com;transport=sctp" are both valid (although note that UDP is not a valid transport for SIP). The use of "transport=tls" has consequently been deprecated, partly because it was specific to a single hop of the request. This is a change since RFC 2543.

Users that distribute a SIP URI as an address-of-record may elect to operate devices that refuse requests over insecure transports.

3.2.1.3 HTTP Authentication

SIP provides a challenge capability, based on HTTP authentication, which relies on the 401 and 407 response codes as well as header fields for carrying challenges and credentials. Without significant modification, the reuse of the HTTP Digest authentication scheme in SIP allows for replay protection and one-way authentication.

3.2.1.4 S/MIME

As discussed above, encrypting entire SIP messages end-to-end for the purpose of confidentiality is not appropriate because network intermediaries (like proxy servers) need to view certain header fields in order to route messages correctly, and if these intermediaries are excluded from security associations, then SIP messages will essentially be non-routable.

However, S/MIME allows SIP UAs to encrypt MIME bodies within SIP, securing these bodies end-to-end without affecting message headers. S/MIME can provide end-to-end confidentiality and integrity for message bodies, as well as mutual authentication. It is also possible to use S/MIME to provide a form of integrity and confidentiality for SIP header fields through SIP message tunneling.

3.2.2 Attacks and Thread Models

The security mechanisms must be combined properly to obtain a trusted network scenario. These attacks assume an environment in which attackers can potentially read any packet on the network - it is anticipated that SIP will frequently be used on the public Internet. Attackers on the network may be able to modify packets (perhaps at some compromised intermediary). Attackers may wish to steal services, eavesdrop on communications, or disrupt sessions.

SIP communications are susceptible to several types of attack:

1. The simplest attack in SIP is *snooping,* which permits an attacker to gain information on users' identities, services, media, and network topology and so on. This information can be used to perform other types of attacks.

2. *Modification attacks* occur when an attacker intercepts the signalling path and tries to modify SIP messages in order to change some service characteristics. This kind of attack depends on the kind of security used.

3. *Snoofing* is used to impersonate the identity of a server or user to gain some information provided directly or indirectly by the attacked entity.

4. Finally, SIP is especially prone to *denial of service attacks* that can be performed in several ways, and can damage both servers and user agents.

The attack techniques may cause memory exhaustion, processor overload, and so on.

3.2.3 Limitations

Although the security mechanisms provided with SIP can reduce the risk of attacks, there are some limitations in the scope of the mechanisms that must be considered.

3.2.3.1 HTTP Digest

One of the primary limitations of using HTTP Digest in SIP is that the integrity mechanisms in Digest do not work very well for SIP. Specifically, they offer protection of the Request-URI and the method of a message, but not for any of the header fields that UAs would most likely wish to secure.

The existing replay protection mechanisms described in RFC 2617 also have some limitations for SIP. The next-nonce mechanism, for example, does not support pipelined requests. The nonce-count mechanism should be used for replay protection.

Another limitation of HTTP Digest is the scope of realms. Digest is valuable when a user wants to authenticate themselves to a resource with which they have a pre-existing association, like a service provider of which the user is a customer (which is quite a common scenario and thus Digest provides an extremely useful function). By way of contrast, the scope of TLS is inter-domain or multi-realm, since certificates are often globally verifiable, so that the UA can authenticate the server with no pre-existing association.

3.2.3.2 S/MIME

The largest outstanding defect with the S/MIME mechanism is the lack of a prevalent public key infrastructure for end users. If self-signed certificates (or certificates that cannot be verified by one of the participants in a dialogue) are used, the SIP-based key exchange mechanism is susceptible to a man-in-the-middle attack with which an attacker can potentially inspect and modify S/MIME bodies. The attacker needs to intercept the first exchange of keys between the two parties in a dialogue, remove the existing CMS-detached signatures from the request and response, and insert a different CMS-detached signature containing a certificate supplied by the attacker. Each party will think they have exchanged keys with the other, when in fact each has the public key of the attacker.

It is important to note that the attacker can only leverage this vulnerability on the first exchange of keys between two parties – on subsequent occasions the alteration of the key would be noticeable to the UAs. It would also be difficult for the attacker to remain in the path of all future dialogues between the two parties over time (as potentially days, weeks, or years pass).

SSH is susceptible to the same man-in-the-middle attack on the first exchange of keys; however, it is widely acknowledged that while SSH is not perfect, it does improve the security of connections. The use of key fingerprints could assist to SIP, just as it does for SSH. For example, if two parties use SIP to establish a voice communications session, each could read off the fingerprint of the key they received from the other, which could be compared against the original. It would certainly be more difficult for the man-in-the-middle to emulate the voices of the participants than their signalling (a practice that was used with the Clipper chip-based secure telephone).

The S/MIME mechanism allows UAs to send encrypted requests without preamble if they possess a certificate for the destination address-of-record on their key ring. However, it is possible that any particular device registered for an address-of-record will not hold the certificate that has been previously employed by the device's current user, and that it will therefore be unable to process an encrypted request properly, which could lead to some avoidable error signalling. This is especially likely when an encrypted request is forked.

The keys associated with S/MIME are most useful when associated with a particular user (an address-of-record) rather than a device (a UA). When users move between devices, it may be difficult to transport private keys securely between UAs; how a device might acquire such keys is outside the scope of this document.

Another, more prosaic difficulty with the S/MIME mechanism is that it can result in very large messages, especially when the SIP tunneling mechanism described in Section 3.3.4 is used. For that reason, it is recommended that TCP should be used as a transport protocol when S/MIME tunneling is employed.

3.2.3.3 TLS

The most commonly voiced concern about TLS is that it cannot supplant UDP; TLS requires a connection-oriented underlying transport protocol, which for the purposes of this document means TCP.

It may also be arduous for a local outbound proxy server and/or registrar to maintain many simultaneous long-lived TLS connections with numerous UAs. This introduces some valid scalability concerns, especially for intensive cipher suites. Maintaining redundancy of long-lived TLS connections, especially when a UA is solely responsible for their establishment, could also be cumbersome.

TLS only allows SIP entities to authenticate servers to which they are adjacent; TLS offers strictly hop-by-hop security. Neither TLS, nor any other mechanism specified in this work, allows clients to authenticate proxy servers to whom they cannot form a direct TCP connection.

3.2.3.4 SIP URIs

Actually using TLS on every segment of a request path requires that the terminating UAS must be reachable over TLS (perhaps registering with a SIP URI as a contact address). This is the preferred use of SIP. Much valid architecture, however, use TLS to secure part of the request path, but rely on some other mechanism for the final hop to a UAS, for example. Thus, SIP cannot guarantee that TLS usage will be truly end-to-end. Note that since many UAs will not accept incoming TLS connections, even those UAs that do support TLS may be required to maintain persistent TLS connections as described in the TLS limitations section above in order to receive requests over TLS as a UAS.

Location services are not required to provide a SIP binding for a SIP Request-URI. Although location services are commonly populated by user registrations, various other protocols and interfaces could conceivably supply contact addresses for an AOR, and these tools are free to map SIP URIs to other SIP URIs as appropriate. When queried for bindings, a location service returns its contact addresses without regard for whether it received a request with a SIP Request-URI. If a redirect server is accessing the location service, it is up to the entity that processes the Contact header field of a redirection to determine the propriety of the contact addresses.

Ensuring that TLS will be used for all of the request segments up to the target domain is somewhat complex. It is possible that cryptographically authenticated proxy servers along the way that are non-compliant or compromised, may choose to disregard the forwarding rules associated with SIP. Such malicious intermediaries could, for example retarget a request from a SIP URI to another SIP URI in an attempt to downgrade security.

Alternatively, an intermediary might legitimately retarget a request from a SIP to a SIP URI. Recipients of a request whose Request-URI uses the SIP URI scheme thus cannot assume on the basis of the Request-URI alone that SIP was used for the entire request path (from the client onwards).

To address these concerns, it is recommended that recipients of a request whose Request-URI contains a SIP or SIP URI inspect the To header field value to see if it contains a SIP URI (though note that it does not constitute a breach of security if this URI has the same scheme but is not equivalent to the URI in the To header field). Although clients may choose to populate the Request-URI and To header field of a request differently, when SIP is used this disparity could be interpreted as a possible security violation, and the request could consequently be rejected by its recipient. Recipients may also inspect the via header chain in order to double-check whether or not TLS was used for the entire request path until the local administrative domain was reached. S/MIME may also be used by the originating UAC to help ensure that the original form of the To header field is carried end-to-end.

If the UAS has reason to believe that the scheme of the Request-URI has been improperly modified in transit, the UA should notify its user of a potential security breach.

As a further measure to prevent downgrade attacks, entities that accept only SIP requests may also refuse connections on insecure ports.

End users will undoubtedly discern the difference between SIP and SIP URIs, and they may manually edit them in response to stimuli. This can either benefit or degrade security. For example, if an attacker corrupts a DNS cache, inserting a fake record set that effectively removes all SIP records for a proxy server, then any SIP requests that traverse this proxy server may fail. When a user, however, sees that repeated calls to a SIP AOR are failing, they could on some devices manually convert the scheme from SIPS to SIP and retry. Of course, there are some safeguards against this (if the destination UA is truly paranoid it could refuse all non-SIP requests), but it is a limitation worth noting. On the bright side, users might also divine that ’SIP’ would be valid even when they are presented only with a SIP URI.

3.3 Delay

The effects of delay on SIP services were also studied. The SIP-T signalling system is a mechanism that uses SIP to facilitate the interconnection of PSTN with carrier-class VoIP network. Based on IETF, the SIP-T signalling system not only promises scalability, flexibility, and inter-operability with PSTN but also provides call control function of MGC (Media Gateway Controller) to set up, tear down and manage VoIP calls in carrier-class VoIP network. The performance analysis of the SIP-T signalling system plays an essential role in optimizing network QoS (Quality of Service). Queuing size, mean of queuing delay and the variance of queuing delay are the major performance attributes of MGC in carrier class VoIP networks. Wu et al [33] assumed a mathematical model of the M/G/1 queue with non-pre-emptive priority assignment to represent the SIP-T signalling system. They also presented the formulas for queuing size, queuing delay and delay variation for the non-pre-emptive priority queue from queuing theory respectively.
Since a satellite component has been identified within the Universal Mobile Telecommunication System (UMTS), there is a need to support SIP-based sessions over Satellite-UMTS(S-UMTS) as well as to achieve an end-to-end seamless IP-based terrestrial/satellite network integration. Kueh et al [35] aimed at evaluating the performance of SIP-based session setup over S-UMTS, taking into account the larger propagation delay over the satellite as well as the impact of the UMTS radio interface.

3.3.1 Queuing Delay

The queuing size was analyzed [33] using imbedded Markov chain and Semi-Markov processing while the queuing delay and delay variation was analyzed using the LST (Laplace-Stieltjes Transform) [41]. Queuing size is defined as the number of SIP-T messages in the system.

Table 1 and Table 2 below shows the comparative data between the mathematical and simulation results including buffer size, mean queuing delay and standard deviation of queuing delay for Pu=0.004. Pu denotes the error probability of a SIP-T message.

The increase of the size of the queue, mean queuing delay, and standard deviation of queuing delay, vary slowly at SIP-T messages arrival rate of less than 450 messages. However, the values increase dramatically when the arrival rate of SIP-T messages exceed 450 messages. An intuitive and reasonable explanation for this phenomenon is that the SIP-T message arrival rate approaches the processing capability of the system for heavy traffic intensity [33].

From the results in Table 1 and Table 2, we can see that the theoretical estimates are shown to be in excellent consistence with simulation results. Thus, we can determine how much the ratio of cost to performance can tolerate and how the planning and design is needed to meet the requirements of the carrier class VoIP network.

Mathematical			
SIP-T message (messages/s)	**Mean (ms)**	**Std Dev (ms)**	**Buffer Size**
50	0.49	0.49	0.12
100	0.62	0.68	0.23
150	0.78	0.89	0.37
200	1.00	1.14	0.52
250	1.29	1.46	0.72
300	1.71	1.90	0.97
350	2.37	2.57	1.33
400	3.57	3.75	1.92
450	6.39	6.41	3.17
500	25.00	18.57	8.20

Table 1: Mathematical Results [33]

Simulated				
SIP-T message (messages/s)	**Mean and 95th percent (ms)**	**Std. Dev (ms)**	**Buffer Size**	**Sample Size (SIP-T message)**
59.2	0.49±0.01	0.47	0.12	4925
100.7	0.61±0.01	0.65	0.25	10074
150.4	0.79±0.01	0.88	0.39	15044
201.8	0.98±0.02	1.10	0.54	20178
251.4	1.27±0.02	1.40	0.76	25138
301.1	1.70±0.02	1.91	1.03	30115
348.9	2.36±0.03	2.57	1.40	34891
399.5	3.62±0.04	3.98	1.97	39955
450.7	6.49±0.06	6.67	3.49	45067
502.5	21.08±0.18	20.12	8.85	50251

Table 2: Simulation Results [33]

3.3.2 Call Setup Delay

The SIP protocol enables a wide set of applications and multimedia over IP (MoIP) is one of them. SIP-based video-telephony is one of the most challenging multimedia over IP applications. A test provided by Cursio and Lundan [34] has been carried out in a 3G network emulator, to measure **post-dialling delay, answer-signal delay and call-release delay**. These delays happen during the lifetime of a SIP call. These results were compared to local, national, international and overseas Intranet LAN calls.

Post-Dialling Delay (PDD) is also called post-selection delay or dial-to-ring delay. This is the time elapsed between when the caller clicks the button of the terminal to call another caller and the time the caller hears his terminal ringing.

Answer-Signal Delay (ASD) is the time elapsed between when the called party picks-up the phone and the time the caller receives indication of this.

Call-Release Delay (CRD) is the time elapsed between when the releasing party (the caller) hangs-up the phone and the time the same party can initiate/receive a new call.

These delays are shown in Table 3 below:

	Local SIP Call	**National SIP Call**	**International SIP Call**	**Overseas SIP Calls**	**3G SIP Calls**
PDD	24ms	38ms	153ms	24ms	62ms
ASD	23ms	31ms	147ms	237ms	45ms
CRD	11ms	30ms	138ms	230ms	50ms

Table 3: 3G SIP Calls vs. Intranet Calls Delay Results [34]

Tests in the case of bandwidth limitations have been done [34]. The call success rate was always 100%, thus bandwidth would not stand in the way of a successful SIP call.

PDD for 2kbps bandwidth was less that 1 second. PDD for 5kbps bandwidth was approximately 420ms. Values of PDD decrease with increasing bandwidth. At the maximum bandwidth of 254kbps, the PDD was around 50ms. In realistic 3G network configurations, the bearer allocated for signalling could be a few kbps [34].

ASD values were constantly 45ms for channels of at least 5kpbs, but ASD increased for very narrow channels (2kpbs to 166 ms). This would be attributed to the fact that the loss is much higher for narrow bandwidth, whereby a 200/OK must be retransmitted [34].

CRD could not be measured, because the media packets were queued up in the simulator, and they blocked the channel for a long time [34].

Globally, these SIP signalling values are well in line with the Grade of Service (GoS) bounds proposed by the ETSI TIPHON QoS classes [47]. Air interface losses and narrow channels have a great impact on the overall SIP call setup time. This yields large call setup times. However, it is expected that UDP/IP header compression and SIP message compression algorithms would greatly reduce the SIP call setup delay over 3GPP networks.

3.3.3 Message Transfer Delay

Tests done by Kueh et al [35] were for message transfer delay for the SIP 'invite' at block error rates of 0%, 10% and 20%. Since the 'invite' method is deemed to be the most important method in SIP, as it is the only method used to establish a session between participants and normally contains the description of the session to be setup, it is interesting to ascertain its performance.

It was found that the message transfer delay increases as the message size increases. This was expected. It was also found that the transfer delay of the 'invite' request is substantially reduced compared to when no link-layer retransmission is employed; whereby the delay reduction increases as the message size and the block error rates increase. This is because without retransmission at the link-layer, a segment that is lost means that the whole message cannot be recovered at the receiver side and thus, the whole message needs to be retransmitted at the session-layer, according to the SIP reliability mechanism.

The tests also show that the delay is lower with the unsolicited and solicited status report option set, compared to only having the solicited feedback. This is because that by incorporating unsolicited feedback on top of solicited, the missing protocol data units (PDU) can be recovered faster since retransmissions of missing PDUs can be performed before polling; also the reduction in delay is greater at a higher block error rate (BLER).

3.3.4 Session Setup Delay

From tests done by Handley and Jacobson [3], it was shown that session setup delay and call blocking probability for a simple call setup sequence, consists only of the sending of the 'invite' request and the 180 ringing response. Results were presented for the different status report trigger settings with Tgood ranging between 0.5s and 10s, and Tbad equal to 0.5s, 2s and 4s. Comparing both schemes, it can be seen that when the channel is good, i.e. for a lower Tbad value a higher Tgood value, there is hardly any difference in performance, but as the channel gets worse, combining both unsolicited and solicited feedback options gives a lower delay and blocking probability.

Due to the inherent characteristics of SIP signalling being transactional-based and generous in size, the transport of these packets over the radio interface is not sufficient and when transverse over the error-prone wireless link plus a larger satellite propagation delay, the session establishment delay can be rather large. In our opinion SIP over Satellite is not advisable yet. From a study [35], it was shown that with the presence of RLCAM, the session setup performance can be substantially improved. Also it was shown that the combination of unsolicited and solicited status report triggers gives a better performance than just solicited alone in terms of delay and blocking probability in a more hostile environment.

3.3.5 Handoff Delay

Total handoff delay (D) is defined [36] as the time between detachment from the old access medium and establishment of communication with the correspondence node (CN). It consists of three (3) components:

1. Time for switching lower layer medium to access network (D1).
2. Time for detecting a new router and a new link (D2)
3. Time for recovery of communication with a CN after detecting a new link (D3).

D3 is the one that Nakajima et al [36] concentrated on. There are two main factors which contribute to delay D3; Duplicate Address Detection (DAD) and router selection. DAD imposes a delay between receiving a Router Advertisement (RA) and sending a packet out of the interface with auto configured IPv6 address. The purpose of DAD is to confirm the uniqueness of the IPv6 address on the link.

Nakajima et al [36] also measured the handoff delay of SIP terminal mobility in their IPv6 testbed. Two different scenarios have been considered:

a) SIP mobility without kernel modification
b) SIP mobility with kernel modification.

Measurements have been performed for scenarios a) and b) above. The following table shows the handoff delay, D3, which is related to signalling:

Handoff case	a)	b)
H12	38290.0 ms	171.4 ms
H23	3932.2 ms	161.6 ms
H31	1934.7 ms	161.1 ms

Table 4: Handoff Delay of Signaling [36]

The following table shows the results of another handoff delay related to voice communication:

Handoff case	a)	b)
H12	38546.3 ms	420.8 ms
H23	4187.7 ms	418.6 ms
H31	1949.4 ms	408.4 ms

Table 5: Handoff Delay of Media UDP Packet [36]

It is observed from Tables 4 and Table 5, modified kernel has reduced handoff delay but from a perspective the delay figures are not acceptable for real-time multimedia communications. To complete the study by Nakajima et al [36], they then integrated the MIPL MIPv6 in their testbed and performed the same experiment as done for SIP mobility. They observed that the handoff delay for signalling is about 2ms and the handoff delay for media UDP packet is less than 31ms. The results showed that MIPL MIPv6 with modified kernel outperforms the SIP mobility with modified kernel. However, the author believes that application layer mobility, such as SIP mobility, is a potential candidate to support real-time applications.

3.4 Disruption Time

In providing the VoIP service in the convergence of wireless technologies, the most viable concern is the amount of disruption time to process the handoff of ongoing VoIP call (or session).

A well-defined mobility management framework or scheme should deal with all three types of mobility, i.e. roaming, macro-mobility and micro-mobility, especially seeking to reduce disruption in handoff. Roaming users are interested in staying connected with the network while moving from one network to another network with multiple network interfaces. Micro-mobility protocols aim to handle local movement (e.g. within a domain) of mobile hosts without interaction with the Mobile IP enabled Internet [42]. A macro-mobility protocol based on Mobile IP manages mobility between domains [42].

To provide seamless VoIP service in a wireless/mobile communication environment, delay or disruption in dealing with macro- and micro-mobility must be minimized because noticeable disruption during a voice conversation will make VoIP service users unhappy. In testing [41] it was found that the disruption for handoff of the Mobile IP approach is smaller than that of the SIP approach in most situations, however, SIP shows shorter disruption when the mobile node (MN) and correspondent node (CN) are close. While the SIP-based approach offers several advantages over a corresponding MIP-based solution for typical UDP based VoIP streams, it continues to suffer from several drawbacks. The most significant of which is the absence of a mobility management for long-term TCP connection. Second, it can cause call disruption if the new SIP session is not created completely while the mobile terminal is in the overlapped area. As opposed to a mobile node using Mobile IP, a mobile node using SIP-mobility always needs to acquire an IP address via DHCP, which depending on implementation, can be a major part of the overall handoff delay.

Three factors were considered [40] that affect the handoff delay and packet loss over wireless networks:

1. D_{mTOc}: The one-way delay from the mobile node to correspondent node.
2. D_{hTOn}: The delay in sending a message between the New Foreign Agent and Home Agent.
3. $D_{overtime}$: The time difference between H_{max} and T_{DHCP}. H_{max} is the maximum time for seamless handoff time difference between move detection and the escaping time from old cell. T_{DHCP} is DHCP IP address renewal time.

Tests were performed by Jung et al [40] and the results reflected Tables 6,7 and 8:

D_{hTOn}	**Disruption Time (msec)**		
	MIP	**SIP**	**MIP-SIP**
10	42	100	41
20	70	99	71
30	90	107	95
40	110	95	95
50	131	100	98
60	151	100	100
70	170	92	91
80	191	93	92
90	210	93	92
100	230	94	94

Table 6: Disruption Time vs. D_{mTOc} [40]

Table 6 illustrates the handoff disruption time as the delay D_{hTOn} increases. Here the D_{mTOc} are set at 30ms.

D_{MTOC}	**Disruption Time (msec)**		
	MIP	**SIP**	**MIP-SIP**
10	90	49	48
20	90	70	59
30	90	90	91
40	90	120	91
50	90	125	91
60	90	143	91
70	90	180	91
80	90	200	90
90	90	205	90
100	90	230	90

Table 7: Disruption Time vs D_{mTOc} [40]

Table 7 shows the disruption time as the delay D_{mTOc} increases. Here the D_{oTOn} is assumed to be 30ms. As SIP mobility only depends on the distance between the MN and CN, the disruption time of SIP-mobility increases according to the delay D_{mTOc}.

$D_{overtime}$	Disruption Time (msec)		
	MIP	**SIP**	**MIP-SIP**
0	0.1	0.1	0.21
0.1	0.2	0.2	0.21
0.2	0.21	0.3	0.21
0.3	0.21	0.4	0.21
0.4	0.21	0.5	0.21
0.5	0.21	0.6	0.21
0.6	0.21	0.7	0.21
0.7	0.21	0.8	0.21
0.8	0.21	0.9	0.21
0.9	0.21	1	0.21
1	0.21	1.1	0.21

Table 8: Disruption Time vs $D_{overtime}$ [40]

In Table 6, D_{mTOc} and D_{hTOn}, respectively are set at 30ms and 100ms. Also the H_{max} is fixed at 500ms. Because the handoff in SIP over Mobile IP does not need IP renewal time, its disruption time is the same as that in Disruption Time vs. D_{hTOn}. However as the SIP-based approach always needs a new IP address its disruption time increase proportional to the rate of increase for $D_{overtime}$.

The simulation results show that the proposed approach out-performs the existing approach in most cases. This shows that VoIP can be supported over wireless Internet.

3.5 Fax over IP

For new Voice over IP (VoIP) networks to succeed, they will have to support legacy fax equipment installed at user premises worldwide. This fax requirement is attached to the PSTN, which is very robust and reliable. However, the reach and the popularity of the Internet combined with the fact that its use is low in costs, is a major driver for VoIP networks and for the transfer to it of all applications that exist on the PSTN. The most major of these applications is fax transmission.
Fax transmission has special requirements, the first being that while the loss of a packet during a human conversation is not likely to affect a voice call a great deal, it can easily affect a fax call. This is because fax transmission requires far more signalling and handshaking than a regular telephone call. This includes negotiating details such as speed, paper size, and delivery confirmation. Apart from the signalling in a fax call, the sending and receiving of fax documents are mostly done and interpreted by automated fax machines. Therefore, any error in either the signalling or the actual transmission is likely to result in a lengthy recovery.

For three network models, Choudhary et al [37] measured the link utilization of the inter-proxy-server link (IPSL), the link utilization of the auxiliary link (AUXL), the average end-to-end delay of the SIP signalling packets, the average end-to-end delay of fax data packets, the average SIP call setup time, and the average fax call setup time.

Network Models:

1. Model 1 calls are generated from the T.38 gateway, and all messages are sent to the SIP proxy server on its network. The path between the T.38 gateway and the SIP proxy server has an IP router on it. This is not necessary, but it is done to maintain compatibility with the next two network models where IP routers play an important role. The SIP proxy server of the originator’s network communicates with that of the recipient’s network and initially sends SIP messages to set up a call. Once that is done, the T.38 gateway starts fax transmission. The originating T.38

gateway starts sending fax packets to the SIP proxy server. That is, the SIP proxy server routes and interprets SIP messages. But it also routes IFP packets without actually interpreting them. Such a scenario is likely if the SIP proxy server is implemented on a router within the network. In that case, it interprets SIP messages, possibly translating them, and maintaining state for them. But all other messages are not interpreted by it – they are just routed. This model is shown in Figure 4.

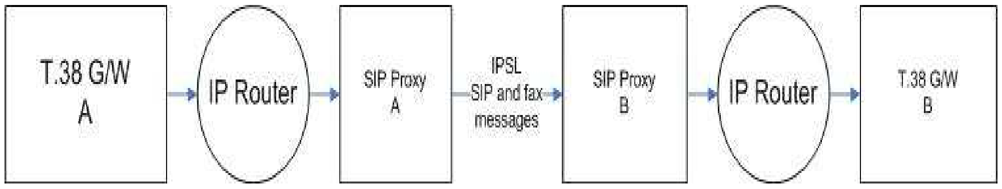


Figure 4: SIP Proxies routing SIP and fax PACKETS [37]

2. Model 2 has the same network as model 1 but a different setup. It includes the components and links of the first model, but also has a direct link between the IP routers. That is, all the SIP signalling is carried out on the path that traverses the two SIP proxy servers. However, once the call is setup, all the fax data packet transmission is done through IP routers and links only. This is more likely to be a network where signalling travels on separate links, and data is sent across another set of links. This is possibly because data does not need to go through SIP proxy servers or other network entities. This frees up resources at such entities and segregates signalling from data transmission, much like PSTN networks where the Signalling System #7 (SS7) links are distinct from trunks. Figure 5 gives a view of this model.

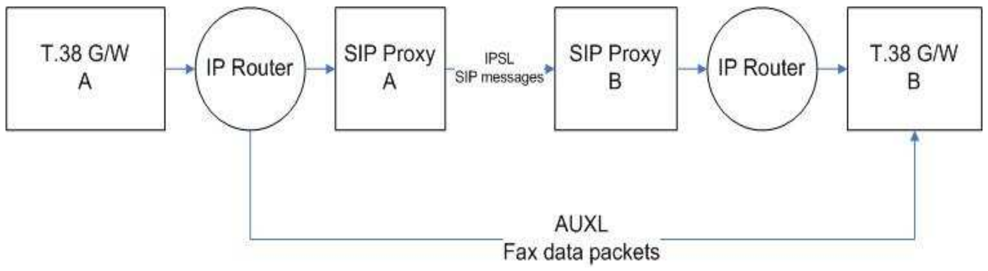


Figure 5: Separate paths for SIP messages and fax images [37]

3. Model 3 uses the same network as network model 2. However, not all SIP messages travel between the SIP proxy servers in this case. In general, all SIP terminals are configured to know their network's SIP proxy server. Hence, to set up a SIP call, they contact their SIP proxy server first not having to know the entire route to a receiver themselves. They only need to know how to route SIP calls they initiate to their designated SIP proxy server. The SIP proxy server then handles all the signalling for the rest of the call. For fax data packets, the AUXL link is used as is done in network model 2.

The measurements obtained by Choudhary et al [37] are reflected in Table 9.

	Model 1	**Model 2**	**Model 3**
Average link utilization of IPSL (Mbps)	77.50	9.223	0.9546
Average link utilization of AUXL (Mbps)		69.03	75.96
Average end-to-end fax data packets delay (sec)	0.2023	0.2017	0.2017
Average end-to-end signalling delay (sec)	0.2014	0.2015	0.2009
Average SIP call setup times (sec)	3.4023	3.4042	3.4020
Average fax call setup time (sec)	4.2086	4.2106	4.2060

Table 9: Experiment Results [37]

In terms of just link utilization values on IPSL, network model 3 is the most suitable. In terms of just link utilization values on AUXL, network model 2 is the most suitable, but it is only marginally better. In terms of end-to-end fax data packet delay, network models 2 and 3 are equally good. In terms of just packet end-to-end delay, network model 3 is the best. In terms of just average SIP call setup time and average fax call setup, network model 3 is the best.

SIP appears to be a powerful and useful signalling protocol supporting mobility for wireless IP networks but it has inherent weaknesses and dangers. These weaknesses and dangers were discussed in Section 2.4.

3.6 Evaluation Methodology of Processing Cost and Experimental Results

Starting from the SIP-based telephony service scenario, eight procedures/scenarios, reflected in Table 11, have been considered in order to compare their processing cost [39]. As reported in Table 5 and Table 6, the basic procedure/scenario (1) in Table 11 is a SIP call setup with no authentication, where the proxy server is call stateless and always uses UDP communication. Procedure/scenario (2) from Table 11 includes authentication and corresponds exactly to the call flow. Scenario (3), seen in Table 11, has been considered in order to see the difference between the use of UDP or TCP between the proxy and tester UAS. The motivation for considering TCP-based SIP communication is to have an incremental analysis toward a TLS-based SIP communication scenario. Procedures/scenarios (5 to 8) replicate (1 to 4), represented in Table 11, as far as authentication and UDP/TCP/TLS are concerned, by considering a call stateful proxy server.
Several measurements were run to define and validate the test methodology Table 10.

Active threads	**1**	**2**	**3**	**4**	**5**	**6**
Single thread throughput (s^{-1})	27.8	15.9	10.6	7.9	6.4	5.3
Total throughput (s^{-1})	27.8	31.7	31.9	31.7	31.8	31.7

Table 10: Experimental Results (threads and throughput) [39]

Table 10 reports the call throughput for the procedure/scenario (1) in Table 11 for different numbers of active threads. Starting from two threads in parallel, the capacity of the server is saturated: each of the N threads takes 1/N of the server capacity, so its throughput is 1/N of the maximum throughput to be evaluated.

	Procedure/Scenario	Total throughput (s^{-1})	Relative processing cost	
1	No authentication, stateless server, UDP	34.8	100	
2	Authentication, stateless server, UDP	19.6	177	
3	Authentication, stateless server, TCP	19.4	180	
4	Authentication, stateless server, TLS	19.2	182	
5	No authentication, call stateful server, UDP	21.0	166	100
6	Authentication, call stateful server, UDP	14.6	239	144
7	Authentication, call stateful server, TCP	13.8	253	152
8	Authentication, call stateful server, TLS	13.6	256	154

Table 11: Experimental Results [39]

Table 11 shows the results of the evaluation by Salsano et al [39] are reported. The third column reports the theoretical maximum throughput in terms of calls per second the proxy server can handle. This includes all the processing that the proxy performs for a call from its setup to its termination. The two rightmost columns are the most important ones and report the throughput value converted into a relative processing cost. In the first one the reference value of 100 has been assigned to procedure/scenario (5).

The results show that the introduction of SIP security accounts for nearly 80% of the processing cost of a stateless server and 45% of a call stateful server. This increase can be explained with the increase in the number of exchanged SIP messages and with the actual processing cost of security. Salsano et al [39] estimated that 70% of the additional cost identified was for message processing and 30% for actual security mechanisms.

3.7 Summary

In this chapter we discussed the major cost of SIP and SIP signalling in wireless networks and services, Security, Delay, and Disruption Time. The security mechanisms in SIP are discussed, as well as the different attacks in SIP and the limitations that still exist, while accommodating security mechanisms in SIP. The different delay types within SIP are discussed. Outlined were also some simulation results, the testing of the security and the delay level of SIP. In this chapter we also discussed some solutions to SIP costs.

Knowing this it can now be determined to what extends the ratio of cost to performance can be tolerated, and how the planning and design is needed so as to meet the requirements of SIP networks and SIP signalling. The next chapter illustrates how to develop low cost applications for a mobile phone, using the knowledge of this chapter.

Chapter 4: System Design and Implementation

4.1 Introduction

In Chapter 3 and the publications by Lakay and Agbinya [48, 49] from this thesis signalling problems associated with SIP have been studied and outlined. Delay and security was identified as the most severe and need to be adhered to for SIP applications to match PSTN (SS7). Some solutions where shown in Chapter 3. In this chapter the knowledge from Chapter 3 is used to create wireless SIP services for small handsets with delay constraint.

The tools and classes used to develop the wireless SIP service are described in Sections 4.2 and 4.3, which is the basic high-level design. Section 4.4 is the low-level design. The system implementation is described in Section 4.5.

4.2 Java 2 Platform Micro Edition (J2ME)

There are many sets of tools and libraries that developers can use to create rich applications for smart mobile devices like PDAs and handsets. In this thesis our focus is on the J2ME approach for developing SIP applications for mobile phones.

J2ME is used for building embedded applications for smart mobile devices [50]. J2ME configurations specify the minimum requirements for memory, Java language features, virtual machine (VM) support and run-time libraries and do not include any optional components. It is available in two main configurations that incorporate a VM and core API's:

1. Connect Limited Device Configuration (CLDC) for low-end, resource-constrained devices with limited connectivity. There are two options, called profiles [53].
   - The Mobile Information Device Profile (CLDC-MIDP) is widely used in hundreds of millions of phones today [50].
   - The Personal Digital Assistant Profile (CLDC-PDAP) is intended for future low-end PDAs that function primarily as PIMs [50].

2. Connected Device Configuration (CDC) which is relatively new. It is intended for new, more sophisticated devices, including PDA devices [50]. There are three profiles that build on each other and relate primarily to the increasing capabilities of the user interface. The most sophisticated of these is Personal Profile (CDC-PP) which equates to the capabilities in J2SE. It is also the natural competitor to the .NET Compact Framework.

There are also a whole set of optional packages that extend these profiles; they include Wireless Messaging API, Mobile Media API, J2ME RMI Optional Package, and the JDBC Optional Package for CDC Foundation Profile, as well as others still in the specification process, such as J2ME Web Services.

In this thesis use was made of the CLDC-MIDP to develop mobile phone applications. CLDC is discussed in more detail in Section 4.2.1 and MIDP is discussed in more detail in Section 4.2.2, since this profile is used to develop mobile phone applications.

4.2.1 Connected Limited Device Configuration Specification (CLDC)

The main goal of the CLDC Specification is to standardize a highly portable, minimum footprint Java^{TM} application development platform for resource-constrained, connected devices [50].

Devices that might be supported by this specification are:

- Cell phones
- Two-way pagers
- Personal Digital Assistants (PDAs)
- Organizers
- Home Appliances

In this study the cell phone formed the core focus. The J2ME configuration specification defines the minimum required complement of Java technology components and libraries for small connected devices [55]. Java language and virtual machine features, core libraries, security, input/output and networking are the primary topics addressed by this specification.

CLDC is core technology that will be used as the basis for one or more profiles [55]. For the purpose of this study, we used the MIDP profile, described in Section 4.2.2. The J2ME profiles define additional libraries that feature device category or industry for a particular vertical market.

4.2.2 Mobile Information Device Profile Specification (MIDP)

The goal of this specification is to define an enhanced architecture and the associated APIs required enabling an open, third-party, application development environment for mobile information devices (MIDs). MIDs span a potentially wide set of capabilities [52]. For the purpose of this thesis, use was made of MIDP2.0.

The MIDP2.0 [52] specification is based on the MIDP1.0 [54] specification and provides backward compatibility with MIDP1.0 so that MIDlets written for MIDP1.0 can be executed in MIDP2.0 environments. The MIDP is designed to operate on top of the connected limited configuration (CLDC) [53] which is described in Section 4.2.1.

4.2.3 Messaging

There are 3 types of messaging in wireless environments [50]:

1. **Instant Messaging (IM):** In wireless environments, instant messaging usually means SMS messaging, because SMS bearer systems are typically used to implement the IM service. SMS bearers provide instant messaging to the extent that messages are sent “immediately” [50].

2. **Electronic Mail (e-mail):** E-mail is the transmission of arbitrary-length messages using a store-and-forward model [50]. The exact capabilities of e-mail systems in wireless networks depend on implementation.

3. **Unified Messaging:** Unified messages is the concept of a single mechanism that unifies other messaging systems and gives users a single access point for all messaging services [50]. A unified message system might have a user interface for both Web-based access and wireless device access.

The IM type of messaging was selected for wireless SIP application development. In wireless environment, IM usually means SMS messaging, because SMS bearer systems are typically used to implement the IM service [50]. SMS bearers provide instant messaging to the extent that messages are sent immediately. This does not mean that the message gets delivered immediately. Delivery time depends on system congestion, flow control, bandwidth constraints, and so forth, which can result in delays that are greater than those experienced in fixed network instant messaging technologies.

4.2.4 J2ME Wireless Toolkit

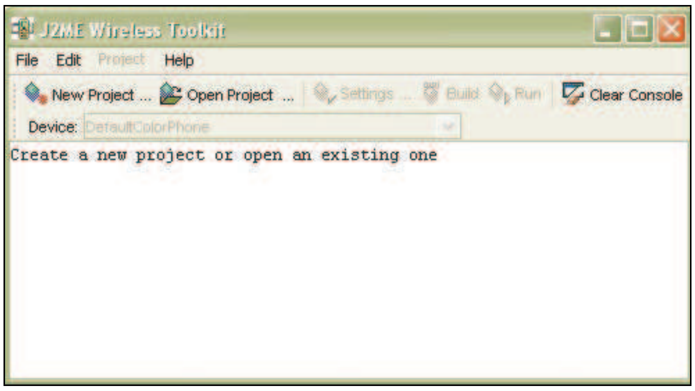
The J2ME Wireless Toolkit (WTK) is a set of tools that makes it possible to create applications for mobile phones and other wireless devices [56]. Although it is based on the Mobile Information Device Profile (MIDP) 2.0, the J2ME WTK also supports a handful of optional packages, making it a widely capable development toolkit.

4.2.4.1 The Tools in the Toolkit

The J2ME WTK has three main components:

- **KToolbar,** see Figure 6 which automates many of the tasks involved in creating MIDP applications.

- The **emulator** is a simulated mobile phone. It is useful for testing MIDP applications. Since it does not support SIP features, this emulator is not used in this study.
- A collection of **utilities** provides other useful functionality including a text messaging console and cryptographic utilities.



```
Create a new project or open an existing one
```

Figure 6: J2ME Wireless Toolkit [56]

KToolbar is the centre of the toolkit. You can use it to build applications, launch the emulator, and start the utilities. Alternately, the emulator and utilities can be run by themselves, which is useful in this situation. If you want to demonstrate MIDP applications, for example, it's useful to run the emulator by itself.
The only additional tool you need is a text editor for source code editing.

4.2.4.2 Toolkit Features

The J2ME WTK supports the creation of MIDP applications with the following features:

- Building and packaging
- Running and monitoring
- MIDlet suite signing

4.2.5 Security Features

To send and receive messages (request/responses) using this API, applications must be granted a permission to perform the requested operations. The mechanisms for granting permission are implementation dependent [50].

4.2.5.1 Permission for Opening Connections

The MIDP2.0 specification defines a mechanism for granting permissions to use privileged features. This mechanism is based on a policy mechanism enforced in the platform implementation. The following permissions are defined for the JSR180 functionality, when deployed with a JSR118 MIDP2.0 implementation.

To open a connection, a MIDlet suite requires an appropriate permission to access the SIPConnection implementation. If permission is not granted, then Connection.open methods must throw a SecurityException.

To create SIP applications using J2ME, use was made of the SIP API for J2ME, which is discussed in the following section.

4.3 SIP API for J2ME

SIP API for J2ME is a J2ME Optional Package that enables resource limited devices to send and receive SIP messages [51]. The API is designed to be a compact and generic SIP API, which provides SIP functionality at transaction level. The API is integrated into the Generic Connection Framework defined in Connected Limited Device Configuration (CLDC).

SIP API for J2ME can be used with many J2ME profiles. The minimum platform required for this API is the J2ME CLDC v1.0. The API can also be used with the J2ME Connected Device Configuration (CDC). Figure 7 shows the simplified class diagram of the API, relations between classes, inheritance from javax.microedition.Connection and the relation to javax.microedition.Connector.

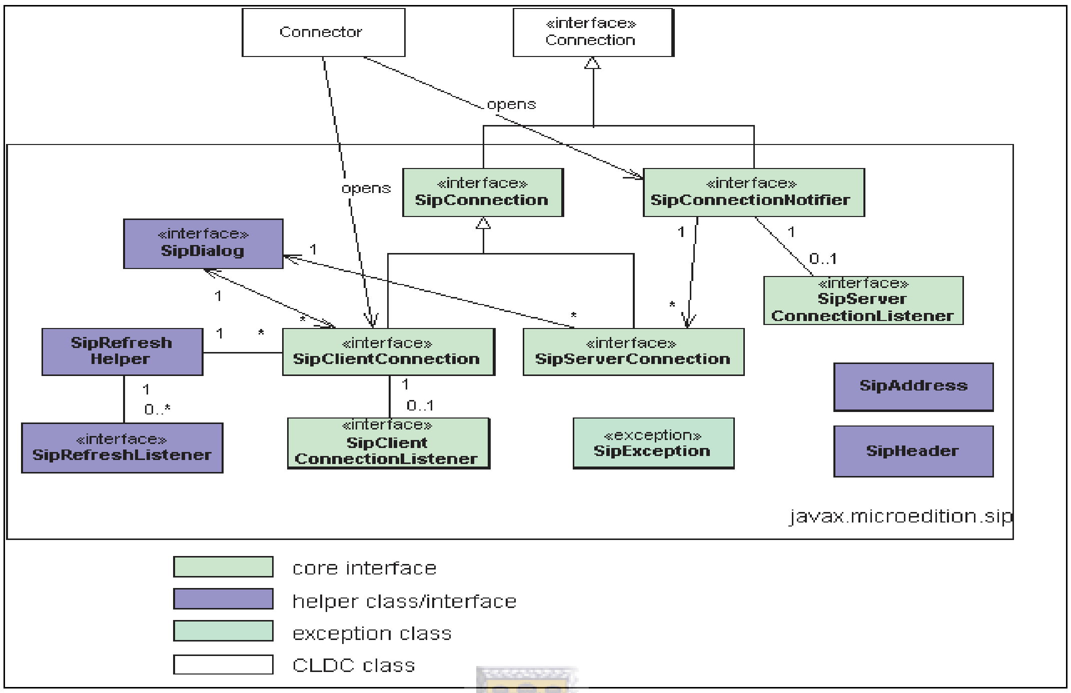

Connector
«interface» Connection
opens
opens
«interface» SipConnection
«interface» SipConnectionNotifier
«interface» SipDialog
«interface» SipServer ConnectionListener
SipRefresh Helper
«interface» SipClientConnection
«interface» SipServerConnection
SipAddress
«interface» SipRefreshListener
«interface» SipClient ConnectionListener
«exception» SipException
SipHeader
javax.microedition.sip
core interface
helper class/interface
exception class
CLDC class

Figure 7: Simplified class diagram of the SIP API for J2ME [51]

The SIP API is used by applications to implement SIP User Agents (UA) functionality. Figure 8 illustrates this scenario.

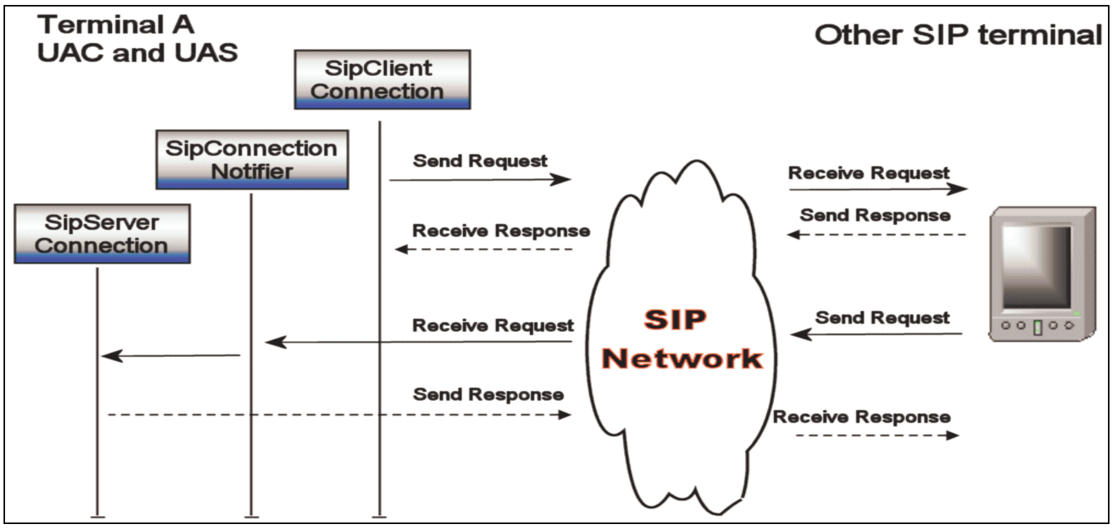


Figure 8: SIP UA functionality, consisting of both client and server. [51]

4.3.1 Javax.microedition.sip

This specification defines the JSR180, SIP API for J2ME. This is the main package used to implement this project's application [51]. This package is a combination of SIP and the java language named J2ME, discussed in Section 4.2. In this section a discussion is given on the classes and interface used.

4.3.1.1 Class Hierarchy

```
Java.lang.Object
        Javax.microedition.sip.SipAddress
        Javax.microedition.sip.SipHeader
        Javax.microedition.sip.SipRefreshHelper
        Java.lang.Throwable
                Java.lang.Exception
                        Java.io.IOException
                                Javax.microedition.sip.SipException
```

Figure 7 is a diagram of the class hierarchy. Next is an explanation on the classes used for this project:

- **SipAddress**

  SipAddress provides a generic SIP address parser [51]. This class can be used to parse either full SIP name addresses or SIP/SIPS URIs. Correspondingly, valid SIP addresses can be constructed with this class.

- **SipHeader**

  SipHeader provides a generic SIP header parser helper [51]. This class can be used to parse bare String header values that are read from the SIP message. SipHeader is a separate helper class and is not mandatory for creating SIP connections.

- **SipException**

This is an exception class for SIP specific errors [51]. The exception includes free format textual error message and error code to categorize the error.

4.3.1.2 Interface Hierarchy

```
Javax.microedition.io.Connection
        Javax.microedition.sip.SipConnection
                Javax.microedition.sip.SipClientConnection
                Javax.microedition.sip.SipServerConnection
        Javax.microedition.sip.SipConnectionNotifier
Javax.microedition.sip.SipClientConnectionListener
Javax.microedition.sip.SipDialog
Javax.microedition.sip.SipRefreshListener
Javax.microedition.sip.SipServerConnectionLister
```

Next are explanations in the interface classes used for this project:

- **SipConnection**

SipConnection is the base interface for SIP connections [51]. SipConnection holds the common properties and methods for sub-interfaces SipClientConnection and SipServerConnection.

A new SIP connection is initiated by a call to Connector.open(). An application should call close() when it is finished with the connection. It should be noticed that the Connector.open() returns a client mode connection (SipClientConnection) or server mode connection (SipConnectionNotifier) depending on the string (SIP URI) passed to the connector.open().

The general form of the SIP URI is:
{scheme}:[{target}][{params}]
where:

- o Scheme: Is the SIP scheme supported by the system sip or sips.
- o Target: Is user network address in form of {user_name}@{target_host}[:{port}] or {telephone_number}.

- o Params: Stands for additional SIP URI parameters.

If the target host is included a SipClientConnection will be returned. The server mode is detected by a missing target host in the SIP URI. The server connection can be opened in two modes, shared and dedicated. This specification defines a special URI format for opening the SIP server mode connections:
{scheme}[:{port}][{params}]
where:

- o Scheme: Is SIP scheme supported by the system sip or sips.
- o Port: Optional port number or asterisk *. Asterisk * is used to detect shared mode server connections.
- o Params: Additional parameters.

- **SipClientConnection**

SipClientConnection represents SIP client transaction [51]. Application can create a new SipClientConnection with Connector or SipDialog object. Figure 9 represents the flow and states of SIP client connections.

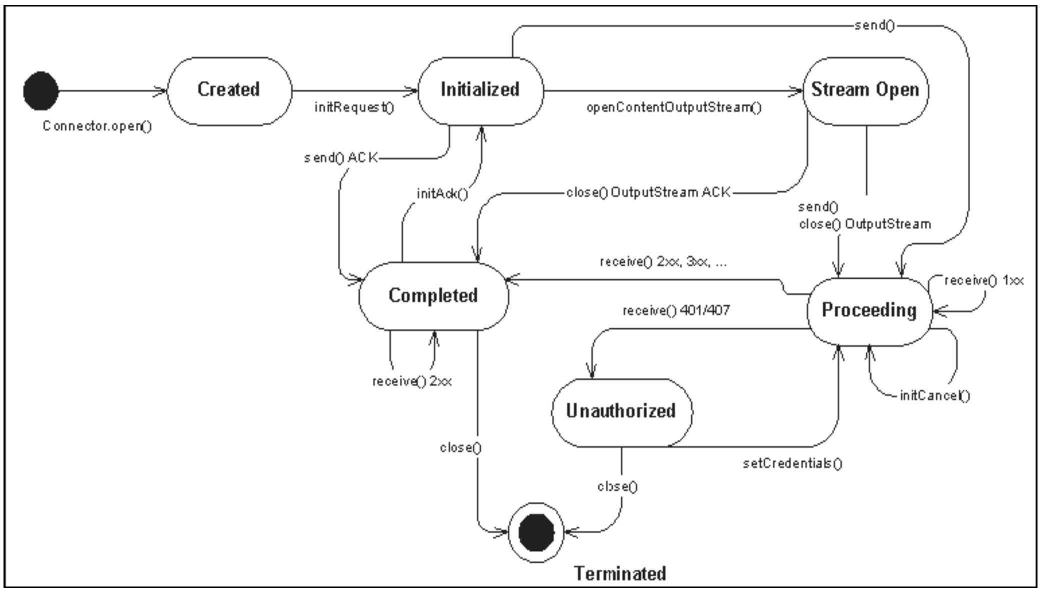


Figure 9: SipClientConnection State Diagram [51]

In what follows is an explanation of Figure 9:

- *Created* – SipClientConnection is created using Connector.

- *Initialized* – Requests are initialised using initRequest() or initAck() or initCancel() or SipDialog.getNewClientConnection().

- *Stream Open* – OutputStream opened with openContentStream(). Opening InputStream for received response does not trigger state transition.

- *Proceeding* – request has been sent, waiting for the response or provisional 1xx response received. initCancel() can be called which will spawn a new SipClientConnection which is in Initialized state.

- *Completed* – transaction completed with final response (2xx, 3xx, 4xx, 5xx, 6xx) in state the ack can be initialised. Multiple 200 OK responses can be received.

- *Unauthorized* – transaction completed with response 401 (Unauthorised) or 407 (Proxy Authentication Required). The application can re-originate the request with proper credentials by calling setCredentials() method. After this the SipClientConnection is back in Proceeding state.

- *Terminated* – the final state in which the SIP connection has been terminated by error or closed.

- **SipServerConnection**

SipServerConnection represents SIP server transaction [51]. SipServerConnection is created by the SipConnectionNotifier when a new request is received. Figure 10 represents the states that a SIP server connection can go through.

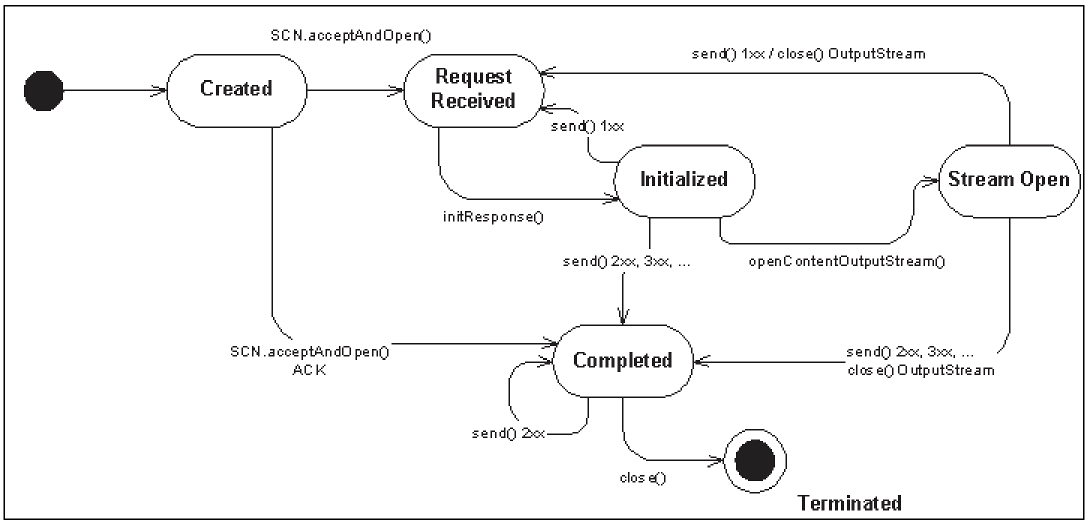


Figure 10: SipServer State Diagram [4]

The following explains of each of the states that the SIP server connection can go through:

- *Created* – SipServerConnection created.

- *Request Received* – SipServerConnection returned from SipConnection (not ack) or provisional response(s) (1xx) sent.

- *Initialized* – Response initialized calling initResponse().

- *Stream Open* – OutputStream opened with openContentOutputStream(). Opening InputStream for received request does not trigger state transition.

- *Completed* – transaction completed with sending final response (2xx, 3xx, 4xx, 5xx, 6xx) or resent 2xx or SipServerConnection for ack returned from SipConnectionNotifier.

- *Terminated* – the final state in which the SIP connection has been terminated by error or closed.

- **SipConnectionNotifier**

This interface defines a SIP server connection notifier [51]. The SIP server connection is opened with Connector.open() using a SIP URI string with the host and user omitted. SipConnectionNotifier can also be opened with sips. SIPS protocol scheme indicates that this server connection accepts only requests over secure transport. SipConnectionNotifier is queuing received messages. In order to receive incoming requests, application calls the acceptAndOpen() method, which returns a SipServerConnection instance. If there are no messages in the queue, acceptAndOpen() will block until a new request is received.

SipServerConnection holds the incoming SIP request message. SipServerConnection is used to initialise and send responses. A SIP server connection can be used to dynamically select an available port by omitting both the host and the port parameters in the connection SIP URI string. The string sip defines an inbound SIP server connection on a port which is allocated by the system. To discover the assigned port number use the getLocalPort() method.

The SipConnectionNotifier also offers an asynchronous call-back interface to wait for incoming requests. The interface is defined in SipServerConnectionListener which the user has to implement in order to receive notifications about incoming requests.

- **SipClientConnectionListener**

  Listener interface for incoming SIP responses [51].

- **SipServerConnectionListener**

  Listener interface for incoming SIP requests [51]

4.4 System Design

In this section an explanation is provided on how the classes and methods explained in Sections 4.1 to 4.3 were used. Apart from the classes used above, a Java servlet in Tomcat Webserver which processes the following requests from the application was designed.

- login
    - CheckUser: Boolean
    - changeUserStatus: void

- logout
    - changeUserStatus: void
    - deleteIP: void

- register
    - checkExistingUser: Boolean
    - insertUser: void

- retrieveBuddies
    - getBuddies: List

- addBuddies
    - checkExistingBuddy: Boolean
    - insertBuddy: void

- deleteBuddy
    - deleteBuddy: void

- storeIP
    - addIP: void

- getIP
    - getIP: string

- getUsername
    - getUsername: string

A database IM was designed with tables' users and buddies. The users table has the following columns:

- username: char
- password: char
- status: tinyint
- IP Address: char

The buddies table has the following columns:

- username: char
- buddy name: char
- buddy status: tinyint

The design of the system is discussed in sub Section 4.4.1 to 4.4.3

4.4.1 Register User

When the user registers with the system, the username given is stored in the MySQL database. By doing this, a request is sent to the Java servlet, stored on Tomcat Webserver, which processes the request. After processing, the Tomcat register query is sent to the database. A search is done in the users table, to check if the username exists or not. If the username exists, a response will be sent to the servlet, which will be processed and the processed response will be in sent to the MIDlet. If the user does not exist the username and password is stored in the users table in the database. The Login screen will be displayed to the user.

4.4.2 Login User

When the user logs in, a request is sent to the servlet, which processes the request and sends a query- login- request to the database. A username and password check is done, by searching through the users table for the username and corresponding password. The query response is then sent to the servlet which sends a response to the MIDlet (user). When the username exists and the username and password corresponds, a success login is sent to the MIDlet and the buddy list of the logged in user is displayed on the MIDlet. A SipConnectionNotifier is then initialised to listen for incoming chat requests from other logged in users. A connection is then opened to listen on port 6000 for incoming requests, using Connector.open().

The buddies of each user are stored in the database under buddies table. To retrieve the buddies, the buddies' table request is sent to the servlet, which will process the request and send a query-retrieveBuddies, to the database.

The IP address of the device is stored in the users table by sending a storeIP query to the servlet.

On an unsuccessful login, a fail message is displayed on the MIDlet.

4.4.3 Buddies List

In the buddies list all the buddies of the specific logged in user are displayed. The user has the option to delete a buddy, add a buddy or chat to a buddy.

- **Delete Buddy**

  When a user chooses to delete a buddy from the menu, a deleteBuddy query is sent to the database. In this query the selected buddy in the buddy list is retrieved, followed by a search in the buddies table for the corresponding username and buddy name. When found, the buddy is deleted from the buddies table and the users' buddy list.

- **Add Buddy**

  To add a buddy, the user has to insert the buddy name. An addBuddy query is sent to the database via the servlet. In doing so, a search is done in the users table to see if the given buddy's name exists as a user of the system. If the buddy exists as a user to the system, a search is done in the buddies table under the username, to check if the given buddy name already exist as a buddy for the logged in user. If not, the buddy is added to the user's buddy list. Otherwise an error is sent to the MIDlet. If the given buddy's name is not a registered user, an error will be sent to the MIDlet.

- **Chat to Buddy**

When choosing to chat to a selected buddy, a query is sent to the database to check if the user is logged on. If the user is logged on, a query - getIP of the selected buddy - is sent to retrieve the buddies IP address. The format of the address of a buddy should be as follows:

sip:username@ipaddress:port

where:

- Username is the buddy name the user chooses to chat to
- IP address is the IP retrieved from the database of the buddy
- Port is the port number the user chooses to chat to, in all cases 6000.

The user does not know the details of the sip address. This is done for the user. After formatting the sip address, a connection is made to the buddy, using Connector.open(). A listener is initialised to listen for messages from the buddy. After the connection is established, the user can choose to send text messages or send picture messages to the buddy he/she chose to chat to.

- **Sending Text Message**

  Firstly, a request is initialised as "MESSAGE". The "From", "Content-Type" and "Content-Length" header is set. After that, the OutputStream is initialised. Convert the text to bytes by using the getBytes method. The message is then written to the buddy, by the OutputStream class, including the bytes. The message is then sent to the buddy following the above-mentioned steps. After that, the OutputStream is closed.

- **Sending Picture Message**

  The request for sending a picture message is initialised as “IMAGE”. The actual image is not sent over, since we could not find a way of converting an image to an array of bytes at the time. The message is sent as a string. This string is handled the same as sending a text message.

4.5 System Implementation

This section gives a simulation of a fast SIP application for small handsets with delay constraints. In implementing the system, the tools described in Sections 4.2 and 4.3 were used. In Section 4.4.1 a real scenario of the system that is implemented was provided. Section 4.4.2 discussed the software and hardware environment while Section 4.4.3 gave the simulation of this project.

4.5.1 Real Scenario

In Figure 11 the SIP Mobile phones contains the implemented application.

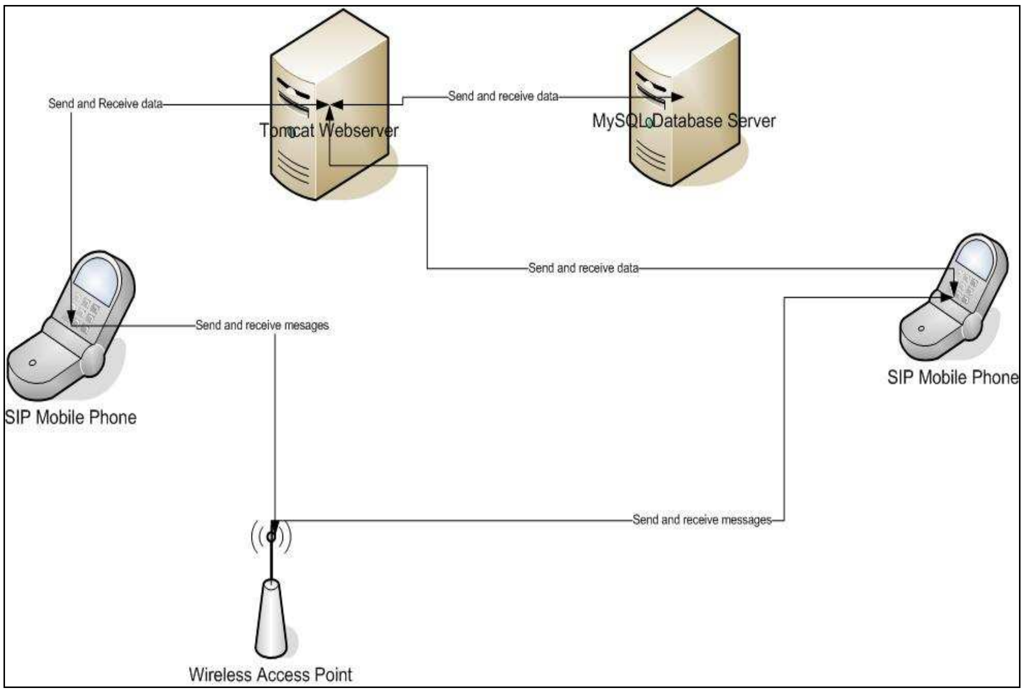


Figure 11: System Real Scenario

For record purposes a Java servlet on Tomcat web server was created which sends and receives data to and from the MySQL database, where the username, password and the status of the registered user is kept for later use.

4.5.2 System Simulation

The simplest application for the mobile phones, the Chat, was tackled followed by the instant messaging application. The scenario in Figure 7 was used with both server and client to create both Chat and IM services for the mobile phones application. Since IM is an extension of the Chat application we will only give a discussion of the IM application development.

4.5.2.1 Creating the System

A project was created in the WTK KToolbar, named InstantMessaging. In this project, a folder, InstantMessaging is created in the WTK application directory:

- C:\WTK22\apps\InstantMessaging

The WTK was installed on the C drive. Within the InstantMessaging folder, the following folders are created automatically:

- Bin: Where the JAR and JAD files of the application are stored.
- Classes: Where the class files of the compiled application are stored.
- Src; Where the source code is stored.

After the project was created, the source code was edited in Netbeans editor and stored in the src folder. This java source code is then compiled and preverified, by building it with the WTK. The compiled files, class files, are stored in the classes' folder

Next the project was packaged, where WTK compiles and preverifies the source files and bundles the java files and resource files into the MIDlet suite JAR files and MIDlet suite descriptor. These MIDlet suites are stored in the bin folder. The MIDlet suite is now ready to be installed onto a real device or emulator.

In this case we used the Nokia emulator, provided with the SIP API for J2ME, because the J2ME WTK emulator does not support SIP features.

In order to use SIP features in the WTK, the WTK must be modified to support compilation of MIDlet using JSR180. The approach used was:

- Open C:\WTK22\lib\midpapi20.jar in WinZip
- Open C:\sip1_0_1-ri-bin\lib\sipa1_0_1.jar in another WinZip
- Drag and drop the classes from sip1_0_1.jar to the midpapi20.jar

To use the application, it needs to be installed on J2ME SIP enabled devices, in this case mobile phones. The application can also be run from the command line, specifying the emulator directory plus the class directory of the application, e.g.

- C:\sipa1_0_1-ri-bin\midp\bin\sipa-midp–classpath /wtk22/apps/InstantMessaging/bin/InstantMessaging.jar IM

whereby:

- C:\sip1_0_1-ri-bin\midp\bisipa-midp is the directory of the emulator
- /wtk22/apps/InstantMessaging/bin/InstantMessaging.jar is the directory of the application you want to run on the emulator
- IM is the classfile of the application.

To install the application on a J2ME SIP enabled device or emulator, the JAR and JAD files need to be placed on a web server, where the MIME type for a JAD file is *text/vnd.sun.j2me.app-descriptor* and the MIME type for JAR file is *application/java-archive*.

4.5.2.2 Using the System

Figure 12 represents the system flow when using the system.

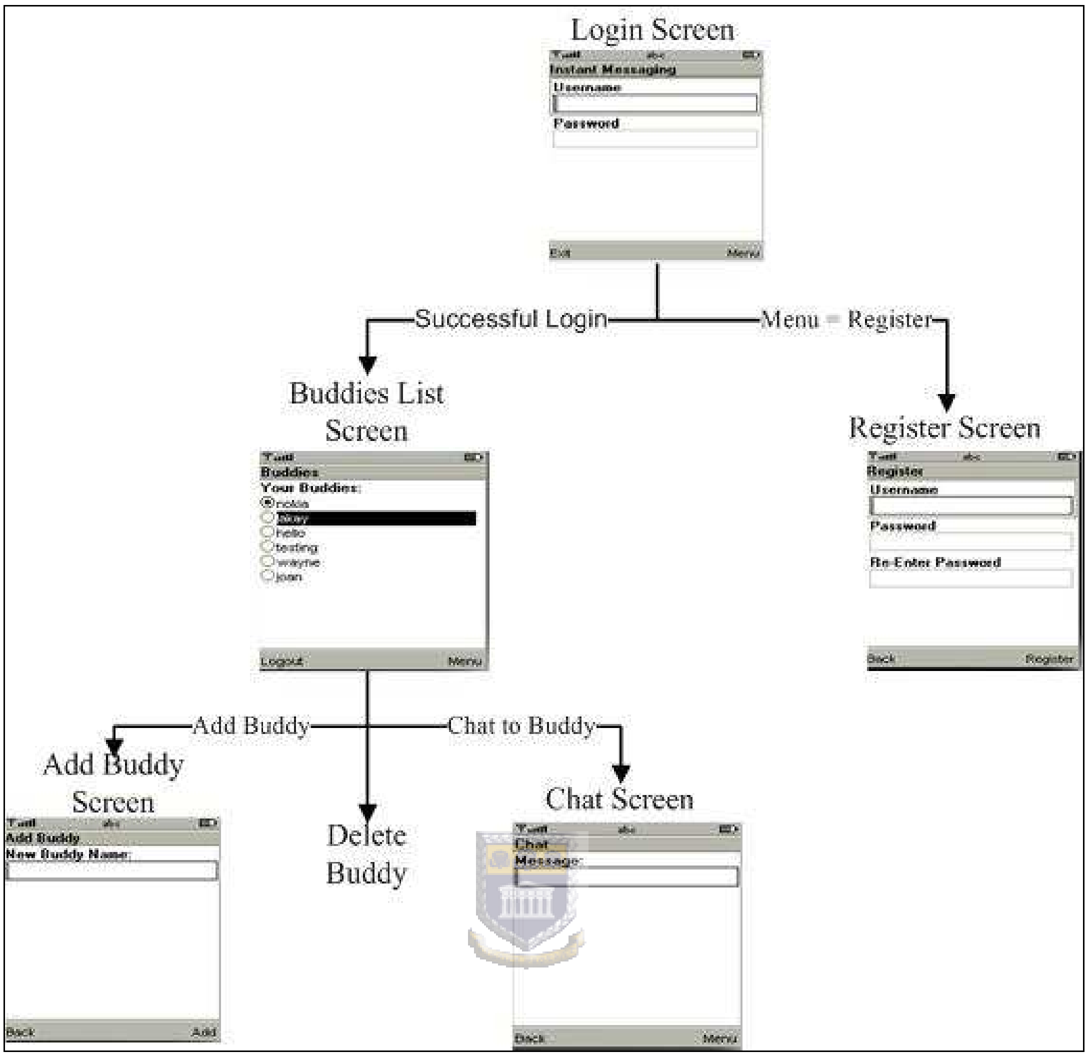


Figure 12: System Basic Flow Chart

The first screen that appears is the login screen, as seen in Figure 12, where the client must provide a login and password. If the user is not registered, he/she can register. The registered username and password is saved in a database in table of users. This is purely for record purposes. After successful registration the user is free to login. When logging in, a SipServerConnection is made with a set port number 6000. A SipConnectionNotifierListener is then initialised, to listen for incoming requests.

After a successful login, the buddy list of the logged in user is displayed. The user's buddies are also stored in a database in table buddies. In this state of the system, the user can:

- Add a buddy – This allows the user to add other registered user as a buddy.
- Delete buddy – This allows the user to delete an unwanted buddy from the list.
- Chat to a chosen buddy – This allows the user to chat to one of the logged in buddies. In choosing this option, a SipClientConnection is made. The SipConnectionNotifierListener will notify the buddy of the connection made from the user. The buddy can then choose to accept or reject the connection. On rejecting the connection the user is notified as such. In accepting the connection, the Chat Screen will be displayed to both the user and the buddy. The user and buddy are now free to send messaging to and fro.

4.6 Summary

In this chapter, descriptions are given of the tools and classes used to develop the system. J2ME is the main tool used, where CLDC and MIDP standards were used for developing the handset application. J2ME is used for building embedded applications for smart mobile devices. To use J2ME with SIP, the SIP API for J2ME had to be used, using the javax.microedition.sip package.

After describing the tools and classes, the system implementation is given and described.

In the next chapter, Chapter 5, testing and results are given on the system implemented in this chapter.

Chapter 5: System Experiments and Results

5.1 Introduction

In Chapter 4 and the publications Lakay and Agbinya [57, 58] of this thesis, a detailed description is given on the tools used to develop the instant messaging (IM) client with SIP as the protocol in Java, with delay constraints. The system design and implementation was specified and described in Chapter 4.

In Chapter 5, the system experiments and results are described and given, based on the design described in Chapter 4. First, the software and hardware used for experiments are outlined. The different networks and scenarios in which the IM clients are tested are described, including the relevant results obtained in each network for each scenario. Test was done from the simplest scenario - same network (LAN) and same lab – to the more complex scenario – WAN to WAN, with poor signal strength.

There are two important ports that were used - port 6000 which SIP runs on and port 8084 which the Tomcat Web server servlet runs on. The networks in which the tests are done are LAN and WLAN. The testing scenarios in the different networks are explained later in the chapter.

5.2 Hardware and Software Environments

5.2.1 Hardware Environment

The hardware environment of the system is outlined and described on Table 12.

Name	Description
PC	Intel Pentium 4 CPU 3.00GHz (2 CPUs), 494M RAM, 80G Hard drive
	Intel Pentium 4 CPU 3.20GHz (2 CPUs), 1022M RAM, 70G Hard drive
Laptop	Intel Pentium M processor 1500MHz, 504MB RAM, 40G Hard drive
Wireless Access Point	Cisco Aironet 1200 Series Wireless Access Point
Ethernet	100M bps, support TCP/IP 100M network adapter
Network Connection	Intel Pro/100 VE Network Connection

Table 12: Hardware Environment

5.2.2 Software Environment

The software environment of the system testing is outlined and described in Table 13.

Name	Description
PC	Windows XP Professional, Service Pack 2.
J2ME Wireless Toolkit 2.2	The J2ME Wireless Toolkit (WTK) is a set of tools that makes it possible to create applications for mobile phones and other wireless devices [56]
SIP API for J2ME 1.0.1	SIP API for J2ME is a J2ME Optional Package that enables resource limited devices to send and receive SIP messages [51]
Nokia Emulator	Provided with the SIP API for J2ME package in [51]
Tomcat 5.5	Database server
MySQL Server 4.1	A fast software deliverer, multi-threaded, multi-user and robust SQL (Structured Query Language) database server. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedded into mass-deployed software.

Table 13: Software Environment

5.3 Simulation Environment

For testing the Nokia emulator provided with the SIP API, the J2ME package was used. The total source code size of the IM client is 25.4KB. The total size of the JAR file is 83.0KB. The JAR file is downloaded onto a mobile phone, so that the IM client can be used on the device.

The images used are in png format. A total of ten images are sent for testing purposes, which ranging size from size 995 bytes to 1.05KB. The actual image is not sent, but the image is converted into its text format and sent. A normal SMS' total numbers of characters are 160; the text maximum number of characters was set as such. For testing, the numbers of characters sent to compare delay in different scenarios, where 5, 10, 20, 40, 80 and 160. The average time interval between sending messages is about 100 milliseconds.

Tests are done on text messages, with the different numbers of characters. For testing, the average delay time is calculated from when a message is sent during peak and off-peak times and the total average delay time during peak and off-peak times is calculated. All the test scenarios are done during peak time and then during off-peak time. The results of the peak and off-peak times are then compared, to analyse in which scenarios the IM application can or cannot be used more efficiently. In Section 5.3.1 to 5.3.3 the testing scenarios are explained for each network environment.

5.3.1 Local Area Network Testing Environment

Figure 13 shows the local area network testing environment.

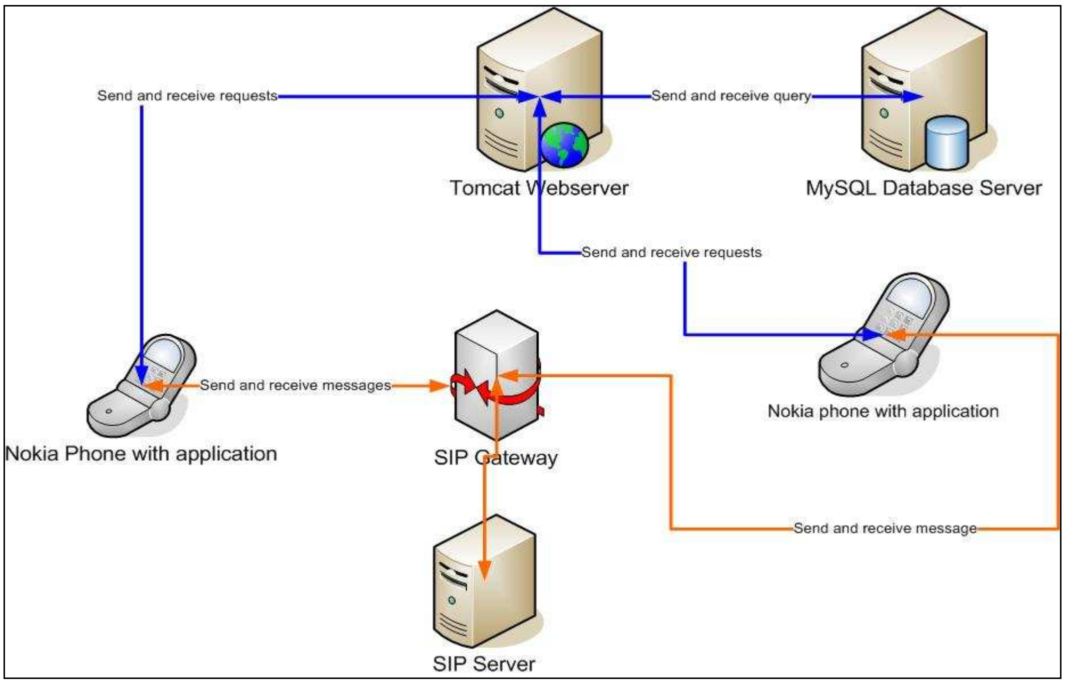

Send and receive requests
Send and receive query
Tomcat Webserver
MySQL Database Server
Send and receive requests
Send and receive messages
Nokia Phone with application
SIP Gateway
Nokia phone with application
Send and receive message
SIP Server

Figure 13: LAN Testing Environment

In this scenario the user can send and receive requests to the Tomcat webserver, which then sends and receives queries to and from the MySQL database. Next the different scenarios in which the testing is done are explained:

- LAN Scenario 1: The IM client is deployed in the same lab, same network. The distance between the terminals is about 1meter.

- LAN Scenario 2: The IM client is deployed in different labs, same network. The distance between the labs is about 15 meters.

- LAN Scenario 3: The IM client is deployed in different labs. The distance between the labs is about 50 meters.

5.3.2 Wireless Network Testing Environment

Figure 14 presents the wireless network testing environment.

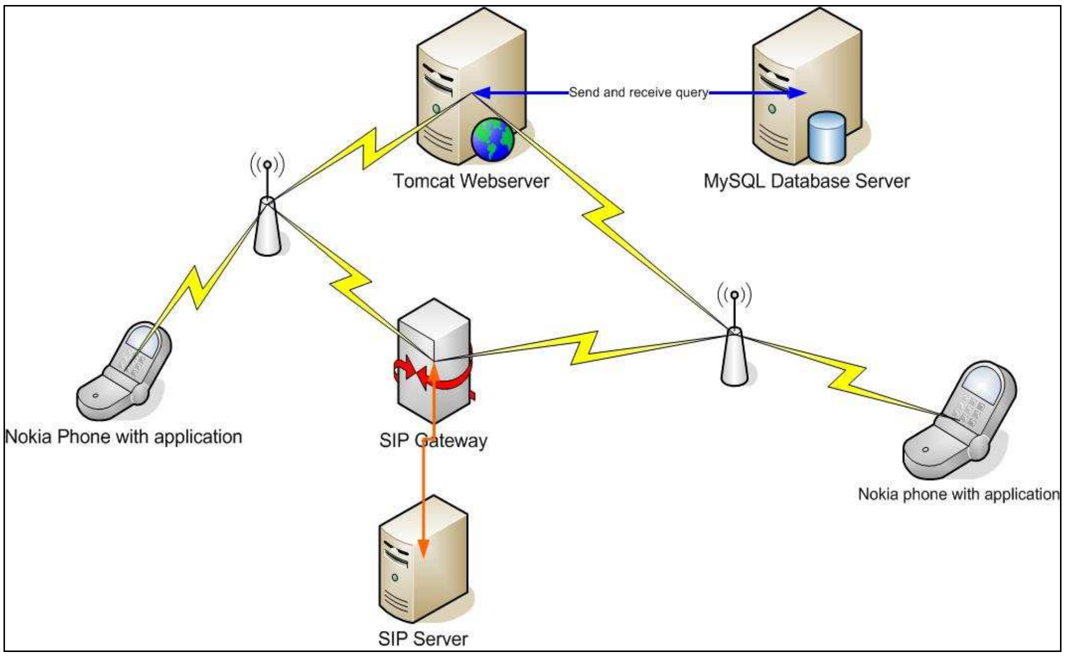


Figure 14: Wireless Network Testing Environment

Two access points (AP) set up in different locations. Next the scenarios are discussed:

- WLAN Scenario 1: The distance between the devices and the AP are ±0.5 meter, with signal strength of excellent.
- WLAN Scenario 2: The distance between the devices and the AP is ±2 meters, with signal strength of good.
- WLAN Scenario 3: The distance between the device and the AP is ±5 meters, with signal strength of low.
- WLAN Scenario 4: The signal strength of the device is poor.
- WLAN Scenario 5: One device is connected to one AP and another to a different AP. The distance between the two APs is about 10 meters. The signal strength on both sides is good.

5.4 *Simulation Results*

In this section the test results for each test scenario, as explained above are given. The delay is calculated by taking the timestamp when a message is sent and the timestamp when a delivery response is received and subtracting the two to get the delay time called message delay time (MDT). The time interval between message sending is 100 milliseconds. This is done for each scenario and scenario case. For this testing, peak hours used are from 8H30 AM to 13H00 PM and 14H00 PM to 17H00 PM, and the off-peak hours are from 13H00 PM to 14H00 PM and 17H00 PM to 8H30 AM Mondays to Fridays, South African times.

5.4.1 LAN Testing Results

To obtain the results in the LAN, the environment described in Figure 13 was used. Figures 15, 6 and 17 graphically present the results obtained.

5.4.1.1 LAN Scenario 1 Testing Results

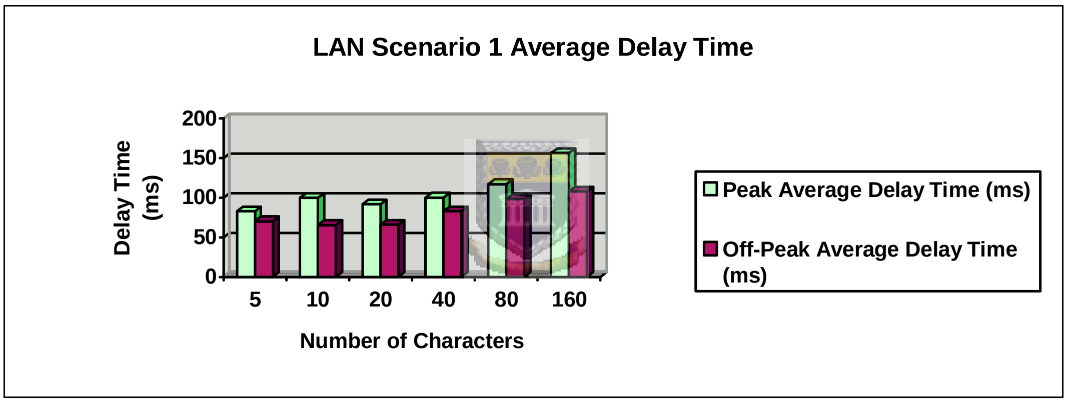


Figure 15: LAN Scenario 1 Testing Results

The sets of tests are performed by sending different numbers of characters. Each time a message is sent, the delay time is calculated. The average delay time is then calculated for each set of character messages.

During peak time, it is observed that the more characters sent, the larger the delay. The reason why this is so, is simply that the more characters sent, the longer the message will be and thus the bigger the size of the message. The bigger the size of a message sent, the more work that needs to be done and more bandwidth over the network is used.

During off-peak time, it is observed that as the number of characters in the message grows so does the average delay time. This is expected, since the more characters that are sent, the more bandwidth that is used resulting in a greater delay time.

In all the test cases, the off-peak average delay time was less than the average delay time during peak hours. This was however expected, since less people are on the university's networks during off-peak hours. Peak and off-peak results are acceptable for real time usage. The overall average delay time during peak in this scenario is 107.97 ms, which is very good to be used in real-time scenarios. The overall average delay time during off-peak hours is 81.8 ms, which is very good. The off-peak total average delay time shows that it's best to use this scenario during off-peak hours.

5.4.1.2 LAN Scenario 2 Testing Results

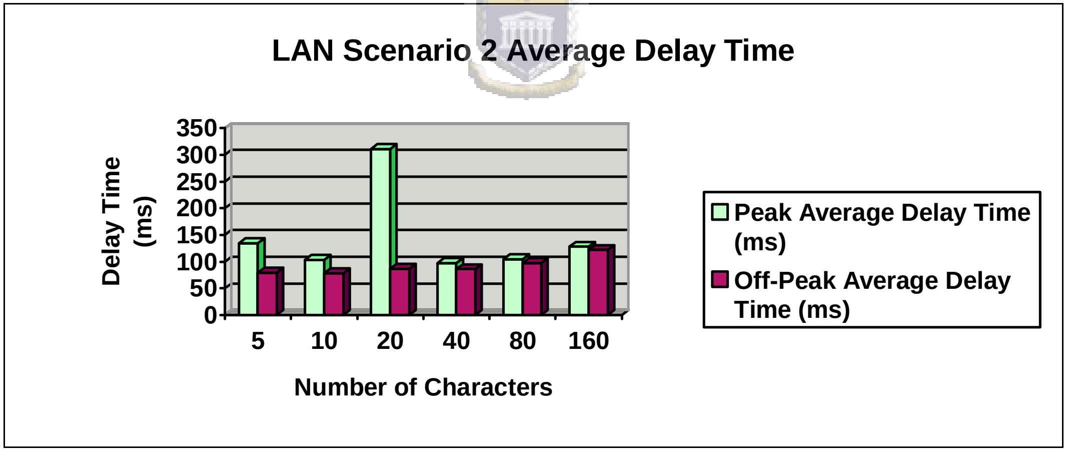


Figure 16: LAN Scenario 2 Testing Results

Figure 16 represents the LAN Scenario 2 testing results during peak and off-peak hours.

During peak, it is found that as the number of characters in the message increases, so does the average delay time. The average delay time reaches its peak for twenty characters. This, however, does not mean that the delay times are always higher for twenty characters than for less than twenty characters. This is because of the network congestion.

The off-peak average delay increases as the number of characters increases. This is expected. It is also found that the average delay times during peak are always greater than the average delay times during off-peak. This was also expected, since less people are on the network at university during off-peak hours than during peak hours times.

The overall average delay time during peak and off-peak hours are 146.4 ms and 91.53 ms respectively. These results show that it is better to use the system during off-peak times in this scenario.

5.4.1.3 LAN Scenario 3 Testing Results

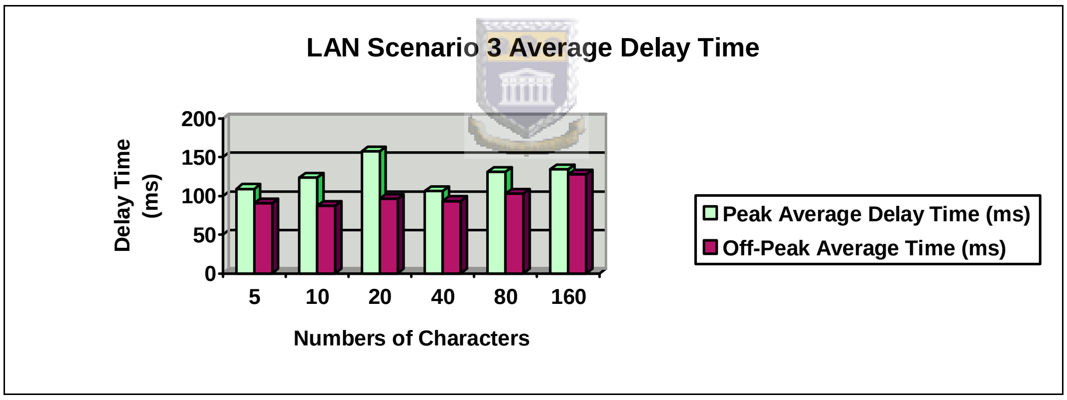


Figure 17: LAN Scenario 3 Testing Results

In the above figure the LAN Scenario 3 testing results are graphically presented. From Figure 17 it can be seen that the average delay during peak is greater than the off-peak average delay in all instances. This shows that the SIP IM client is best for use during peak hours in this scenario. This was expected, since more people are connected to the LAN during peak, than during off-peak, thus more bandwidth space is available and makes throughput of messages much smoother and easier.

The system is better for use during off-peak times, since the overall average delay time during off-peak time is less than the overall average delay time during peak time. The total average delay time during peak is 127.05 ms and during off-peak 99.97 ms.

5.4.2 WLAN Testing Results

This section constitutes the most important part of the testing results, since this project focuses on wireless networks. This section will show and discuss the results obtained from each WLAN scenario discussed in Section 5.3.2. All the WLAN scenarios were tested in Figure 14’s environment. Figures 18 and 19 are a graphical representation of the WLAN average delay results for different numbers of characters.

5.4.2.1 WLAN Scenario 1 Testing Results

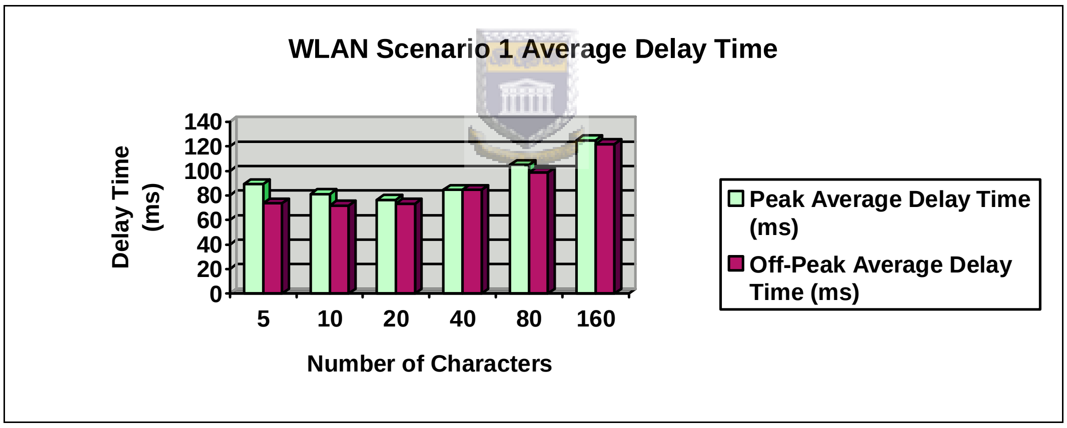


Figure 18: WLAN Scenario 1 Testing Results

Figure 18 graphically presents the average delay times of WLAN Scenario1 during peak and off-peak times.

The average delay time, during peak, decreases as the number of characters in the message increases until twenty characters are reached. Thereafter the average delay time increases as the number on characters increases further. It can be seen that the average delay time, when sending twenty characters, is the lowest. This was not expected since the more characters you have the more bandwidth that is required to send the message and thus the message was supposed to take longer than when sending five or ten characters. The average delay time still falls within the real time criteria and thus the delay is acceptable.

Test results during off-peak hours do not really vary much from those of peak hours. The possible reason for this is because of the strong signal. The signal strength is excellent. As the number of characters increases, the average delay increases. This is expected, since more bandwidth is used when sending bigger messages.

The total average delay time during peak and off-peak hours are 93.48 ms and 87.23 ms respectively. The best time to use the SIP IM client in this scenario is during off-peak hours.

5.4.2.2 WLAN Scenario 2 Testing Results

In this scenario the distance between the terminals and AP are about 2 meters. Figure 19 graphically presents the average results obtained during peak and off-peak time for the various numbers of characters.

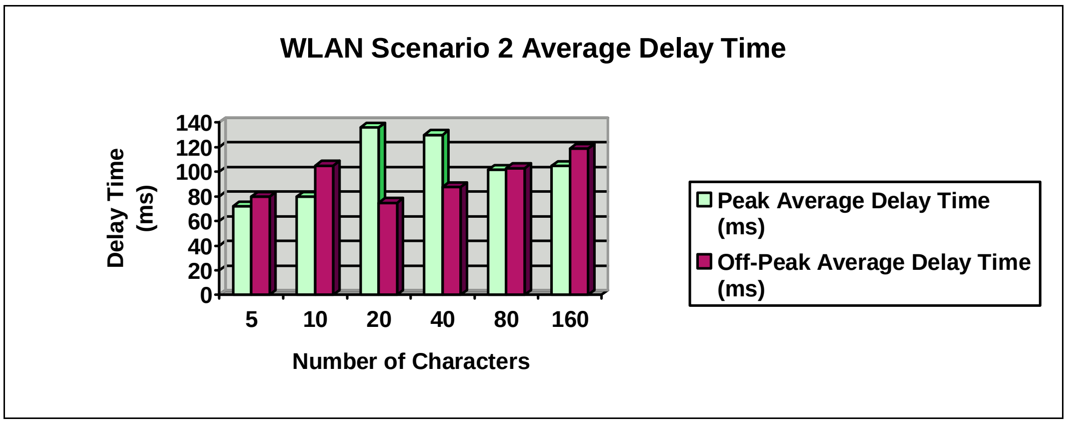


Figure 19: WLAN Scenario 2 Testing Results

The signal strength in this scenario was good. This means that since the signal strength in WLAN Scenario 1 was excellent, the average delay time will be more for WLAN Scenario 2. This, however, was not completely the case.

As expected, the average delay time increased, as the number of characters increased, during peak testing, but only until twenty characters where the average delay time reached its peak. After twenty characters the average delay time decreases as the number of characters increases to 160 characters. It is noted that the opposite happens in WLAN Scenario 1.

Off-peak test results show that between five characters and twenty characters and again between forty characters and hundred and sixty characters, the off-peak average delay is greater than the peak average delay. Between twenty characters and forty characters, the peak average delay is significantly greater than the off-peak average delay. The latter case is normal, but in the former it cannot be explained, since it was expected to perform as in the latter case.

The overall average delay time during peak and off-peak hours is 103.97 ms and 94.78 ms respectively. The best time to use the SIP IM client for this scenario is during off-peak hours.

5.4.2.3 WLAN Scenario 3 Testing Results

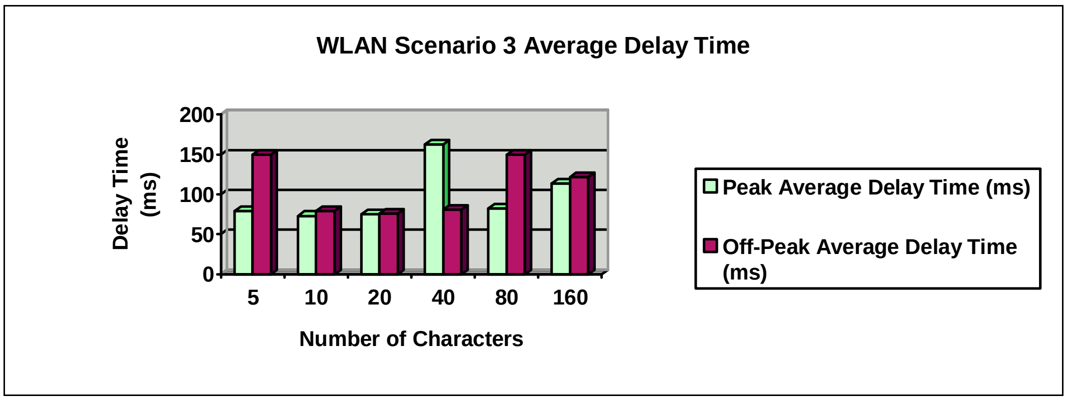


Figure 20: WLAN Scenario 3 Testing Results

The signal strength was low while the tests were being performed. This means that the average delay time must be greater than for the previous two WLAN scenarios. The results actually show the reverse of what was expected. It was found that the delay is at its peak for forty characters. The average delay time decreases as the number of characters increases until the number of characters reaches forty. After forty characters, the average delay time also increases as the number of characters increases.

The peak average delay times are smaller than the off-peak average delay times in all cases, except for forty characters. The total average delay times during peak and off-peak hours are 97.9 ms and 109.83 ms respectively, which show that the SIP IM client is best for use during peak times.

5.4.2.4 WLAN Scenario 4 Testing Results

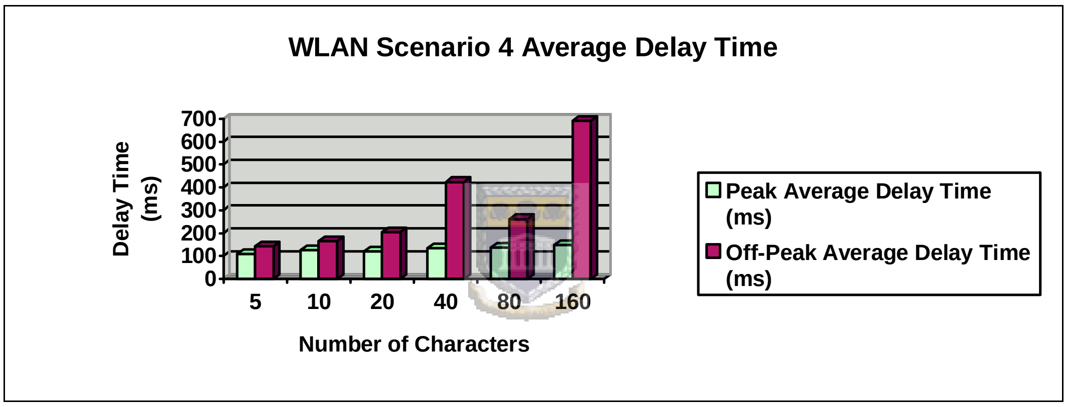


Figure 21: WLAN Scenario 4 Testing Results

Figure 21 graphically presents the testing results in the scenario where the signal strength of the AP is low. The results reflected in Figure 21 show the peak and off-peak average delay time in milliseconds. From Figure 21 we can generally analyse that during peak hours as the number of characters of the message increases the average delay time also increases. This is however expected, since the more characters sent, the more bandwidth is used and thus the message is more prone to delay. The total average delay during peak is 128.93 ms and during off-peak is 314.27 ms.

The average delay time during peak is always less than the average delay time during off-peak. This shows that it is better to use the SIP IM client during peak time rather than off-peak time in this scenario. This scenario has the worst delay of all the scenarios for the SIP IM client. This, however, does not mean that the system is not useable in this scenario.

5.4.2.5 WLAN Scenario 5 Testing Results

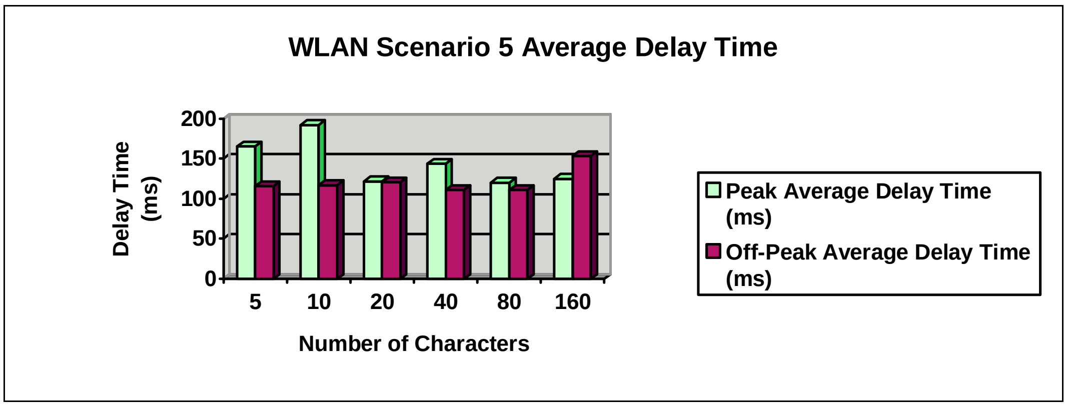


Figure 22: WLAN Scenario 5 Testing Results

Figure 22 graphically presents the average delay time of WLAN Scenario 5, during peak as well as off-peak times. From the above, it can be seen that the average delay time during peak essentially exceeds the average delay time during off-peak, which means that the SIP IM client is better being used during off-peak time rather than peak times.

The total average delay time during peak is 144.78 ms and during off-peak is 121.37 ms. This shows that for this scenario the SIP IM client is better used during off-peak times.

5.4.3 Different Networks Testing Results

In this testing environment, a few IM clients are deployed wirelessly and a few are deployed in a fix network, LAN. Tests are done during peak as well as off-peak. The first set of tests was done when sending messages from the LAN to the WLAN. The second was done by sending messages from the WLAN to the LAN.

5.4.3.1 LAN to WLAN Testing Results

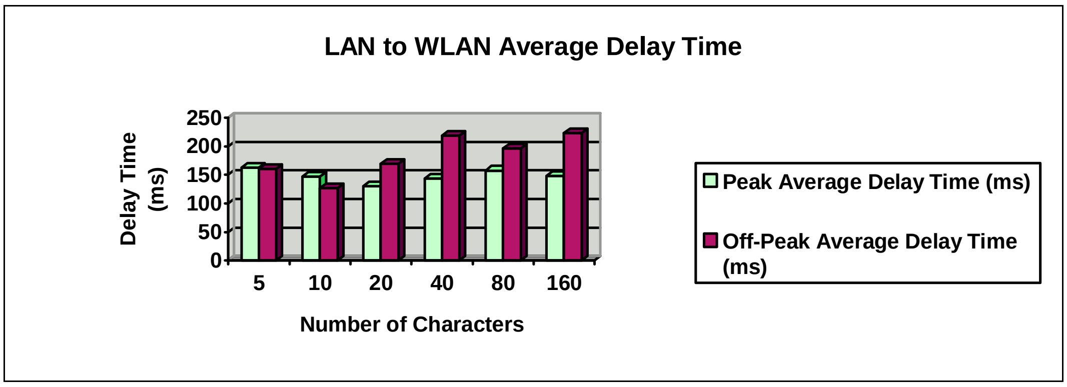


Figure 23: LAN to WLAN Testing Results

Figure 23 graphically presents the average delay time during peak hours and off peak hours, when sending messages from a LAN to a WLAN. From Figure 23 it is seen that the peak average delay time between five characters and twenty characters is less than the off-peak average delay. From twenty characters to hundred and sixty characters the peak average delay is less than the off-peak average delay. It is observed that for this project, delay totally depends on the network.

The total average delay times during peak and off-peak are 148.17 ms and 182.58 ms respectively. This shows that, on average, it is better to use the SIP IM client during peak hours rather than off-peak hours in this scenario.

5.4.3.2 WLAN to LAN Testing Results

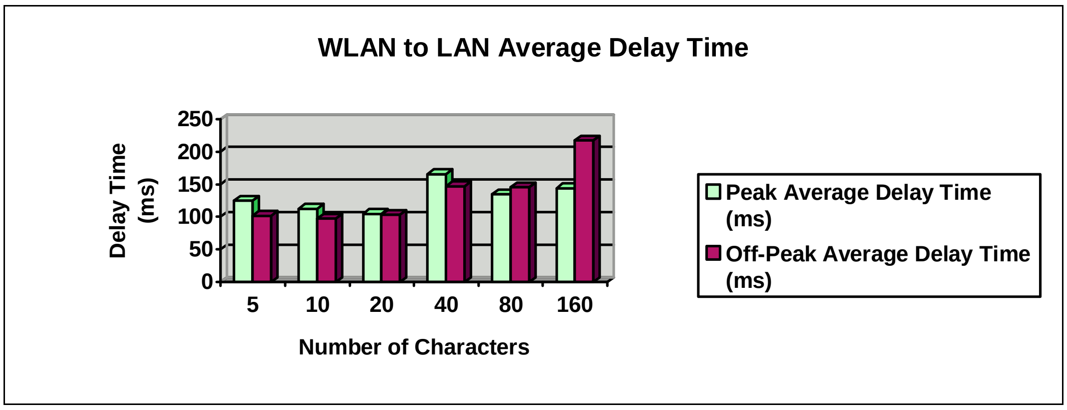


Figure 24: WLAN to LAN Testing Results

Figure 24 graphically presents the average delay time of the peak and off-peak hours, when sending messages from the WLAN to the LAN, using the SIP IM client. From the figure, it is observed that for between five and forty characters, the peak average delay is greater than the off-peak average delay. This case is normal, since less people are on the network during off-peak hours. For eighty to hundred and sixty characters, the peak average delay is less than the off-peak delay. The possible cause for this is that in some cases, when sending a message, the congestion occurs, which causes a bigger delay. The important factor is that the messages are always delivered, even in the case of congestion. A fascinating fact is that when sending messages from a WLAN to a LAN, the average delay time is always less than when sending messages from a LAN to a WLAN. The possible reason for this is that when a message is sent from a fixed network to an over-the-air (OTA) network the exchange delay is greater in this case than when exchanging from an OTA to a fixed network.

The total average delay times during peak and off-peak are 130.98 ms and 135.18 ms respectively, this means that in this scenario it does not really matter when you use the SIP IM client.

5.5 Summary

In this chapter the system simulation environments and testing results are given. The different scenarios were outlined in which the IM client for the phone is tested and the according results, from the simplest scenario to a complex scenario. The results were satisfactory to the extent that it can be stated that the SIP IM client can be used in LAN and WLAN, including different networks.

On average, the SIP IM client is best for use during off-peak hours, since the total average delay, when looking at the bigger picture, is less for almost all the scenarios in which it was tested. As expected, the SIP IM client is best for use in LAN Scenario 1. This was expected because of the small distance the message needs to travel. In the wireless network, the best scenario in which the SIP IM client can be used is during WLAN Scenario 1. This was expected since the signal strength is excellent, thus the distance messages travel is shorter, which results in less delay.

When looking at the overall testing results, it can be concluded that the SIP client can be used in any of the tested networks and scenarios. The delay is feasible, the system is reliable and the system is easy to use which results from the software design criteria in a dedicated WLAN network. This results in the expected software design criteria.

Chapter 6: Conclusion and Future Work

The main aim of this project was to develop SIP services with cost constraints for wireless devices. After the research in Chapter 3, it was found that delay and security were the most severe costs within SIP. Several solutions for the delay and security cost where found and developed by several other people. Thus, it was decided to this knowledge to develop a low cost SIP IM client for wireless devices specifically concentrating on the mobile phone. In Chapter 4 the tools and classes used to develop such a service were described. The J2ME package, which is designed for mobile devices, including two specifications namely CLDC and MIDP specifications, was used.

In Chapter 5, the test results show that the designed system delay constraint is of such a nature that the system can be used in all the tested network scenarios. In Chapter 6 a summary of the research is given, including the solutions to the research questions proposed in Section 1.2, the limitations of the research, and what work can still be done on this topic in the future.

6.1 Research Questions Outcome

In this section the proposed questions in Section 1.2 are revised and answered according to the research, development and testing done in Chapters 2, 3, 4 and 5.

- **What are the most severe SIP security issues?**

  Three security characteristics should be guaranteed, i.e. high service availability (uptime), stable and error-free operations, and protection of the user-to-network and user-to-user network traffic.

- **How can the most severe SIP security issues be overcome?**

  Data authentication is used to authenticate the sender of the message and to ensure that some critical message information is unmodified in transit. Data encryption is used to ensure confidentiality of SIP communications, letting only the intended recipient decrypt and read the data.

- **What are the most severe SIP delay issues?**

There are several delay issues within SIP, which include queuing delay, call setup delay, message transfer delay, session setup delay, and handoff delay. From the five delay types, handoff delay is the most severe. This is because even after modifying the kernels and the delay has been reduced, the delay is still not usable in real-time multimedia communications. Research by Nakajima et al [36] still maintains that application layer mobility is a potential candidate to support real-time applications.

- **How can the most severe SIP delay issues be overcome?**

Most of the delay issues have been overcome, by modifying some of the SIP layers and/or modifying the kernel.

- **Can SIP applications be structured to overcome the delay and security issues?**

Yes. SIP applications can, and in most cases have, overcome the delay and security issues. By just looking at the design, implementation, and testing results of this project, it proved that the costs of SIP have not dominated the power of SIP.

6.2 Limitations of Research

The limitations within the designed application are that the user must be registered to the specific SIP server in order to use the system. The user cannot be registered to another SIP server and use the same account to log into the project system. Users can only send messages to users registered on the system. This means that the SIP application is disclosed only to registered users of the project system.

The system was only tested within the LAN and WLAN and not a real wireless telecommunication network, like Vodacom or MTN. This limits the application largely, because the application can only be used in these networks if all the required tests were done in the networks scenarios. This, however, does not mean that it cannot be done.

Tests were performed only on delay constraints and not security. This limits the research to only half of the identified severe costs.

6.3 Future Research

In this research, the focus was on developing low cost SIP applications for the mobile phone using J2ME, but further research can be done by testing the same application on a PDA. More research can be done by using .NET to develop the same SIP application and comparing the test results to see which application performs better on the mobile phone and PDA.

The application must still be tested in a telecommunication network, like Vodacom or MTN, using GPRS or Bluetooth to see if the application is usable within that network.

A wireless SIP micro-server for the SIP services can be developed to accommodate the services. The micro-server can be language independent, meaning both J2ME and .NET applications can be running on the micro-server.

6.4 Summary

In Chapter 6 this dissertation has discussed the research question posed in Section 1.2. The limitations within the focus and implementation were discussed as well as what can still be done within this area of research. From this thesis it can be stated that though there are still some costs within SIP to be taken into account, SIP is still an easy-to-use protocol, a fast protocol and is on its way to standardising PSTN.

REFERENCES

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002

[2] "About SIP", Available: http://www.sipcenter.com/sip.nsf/html, 2000-2005

[3] M. Handley and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998

[4] P. Resnick, "Internet Message Format", RFC 2822, April 2001

[5] T. Berners-Lee, R. Fielding and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998

[6] F. Yergeu, "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1", RFC 2616, June 1999

[8] D. Crocker and P Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997

[9] F. Freed and N, Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996

[10] H. Schulzrinne, S. Casner, R. Frederick and V Jaconson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996

[11] H. Schulzrinne, R. Rao and R Lanphier, “Real Time Streaming Protocol (RTSP)”, RFC 2326, April 1998

[12] F. Cuervo, N. Greene, A. Rayhan, C. Huitema, B. Rosen and J. Segers, “Megaco Protocol Version 1.0”, RFC 3015, November 2000

[13] M. Handleym, H. Schulzrinne, E. Schooler and J. Rosenberg, “SIP: Session Initiation Protocol”, RFC 2543, March 1999

[14] J. Rosenberg, H. Schulzrinne, “Reliability of Provisional Responses in the Session Initiation Protocol (SIP)”, RFC 3262, June 2002

[15] J. Rosenberg and H Schulzrinne, “SIP: Locating SIP Servers”, RFC 3263, June 2002

[16] J. Rosenberg and H. Schulzrinne, “An Offer/Answer Model with SDP”, RFC 3264, June 2002

[17] A.B. Roach, “Session Initiation Protocol (SIP) – Specific Event Notification”, RFC 3265, June 2002

[18] J.B. Postel, “Simple Mail Transport Protocol (SMTP)”, RFC 821, August 1982

[19] “SIP Overview”, Available: http://www.sipcenter.com/sip.nsf/html/WEBB5YNVK8/$FILE/Ubiquity_SIP_Overview.pdf

[20] R. Padya, “Emerging mobile and personal communication systems”, IEEE Communications Magazine, Vol. 33, June 1995, pp.44-52

[21] "SIP for Wired Network", Available: http://www.dynamicsoft.com/innovation/sip4wire.php

[22] D.E. Comer, D.L. Stevens, "Internetworking with TCP/IP Client-Server Programming and Applications Linux/POSIX Sockets Version", Vol. 3, 2001, pp.106

[23] J. Glasmann, W. Kellerer and H. Müller, "Service Development and Deployment in H.323 and SIP", IEEE Communications Surveys & Tutorials, Vol. 5, No. 2, Fourth Quarter 2003, pp.32-47

[24] S. Berger, H. Schulzrinne S. Sidiroglou, X. Wu, "Ubiquitous Computing Using SIP", IEEE Communications Magazine, Vol.41, No.11, November 2003, pp.128-135

[25] H. Schulzrinne, E. Wedlund, "Application-layer mobility using SIP", ACM SIGMOBILE Mobile Computing and Communications, Vol.4, Issue 3, July 2000, pp.47-57

[26] D.C. Sicker, A. Kulkarni, A. Chavali, M. Fajandar, "A Federated Model for Securing Web-Based Videoconferencing", Proceedings of the International Conference on Information Technology: Computers and Communications (ITCC'03), April 2003, pp.396

[27] W. Rao, W. Li, "Design of An Open and Secure Ubiquitous Computing System", Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI'04), September 2004, pp.656-659

[28] N. Banerjee, S.K. Das, A. Acharya, "SIP-based Mobility Architecture for Next Generation Wireless Networks", Proceedings of 3rd IEEE Int'l Conference on Pervasive Computing and Communications (PerCom 2005), March 2005, pp.181-190

[29] H. Tsai, F. Leu, “Mobile Agent Communication Using SIP”, Proceedings of the 2005 Symposium on Application and the Internet (SAINT’05), January 2005, pp.274-279

[30] J.Y. Zhou, Z.Y, Jia and D.X. Chen, “Designing Reliable Communication Protocols for Mobile Agents”, Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW’03), May 2003, pp.484

[31] D. Tosi, “An Advanced Architecture for Push Services”, Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshop (WISEW’03), December 2003, pp.193-200

[32] T. McFadden, K. Hendricksen, J. Indulska, P. Mascaro, “Applying a Disciplined Approach to the Development of a Conext-Aware Communication Application”, Proceedings of the 3rd IEEE Int’l Conference on Pervasive Computing and Communications (PerCom 2005), March 2005, pp.300-306

[33] J.S. Wu and P.Y. Wang, “The Performance Analysis of SIP-T Signalling System in Carrier Class VoIP Network”, 17th International Conference on Advanced Information Networking and Applications (AINA’03), IEEE March 2003, pp.39

[34] I.D.D. Curcio and M. Lundan, “SIP Call Setup Delay in 3G Networks”, Seventh International Symposium on Computers and Communications (ISCC’02), IEEE July 2002, pp.835

[35] V.Y.H Kueh, R. Tafazolli and B. Evans, “Performance Evaluation of SIP-based Session Establishment over Satellite-UMTS”, Vehicular Technology Conference 2003 (VTC 2003-Spring). The 57th IEEE Semi-annual, IEEE 22-25 April 2003, vol. 2, pp.1381-1385

[36] N. Nakajima, A. Dutta, S. Das and H. Schulzinne, "Handoff Delay Analysis and Measurement for SIP based mobility in IPv6", International Conference on Communications (ICC'03), Vol.26, IEEE May 2003, pp.1085-1089

[37] U. Choudhary, E. Perl and D. Sidhu, "Using T.38 and SIP for Real-Time Fax Transmission Over IP Networks", 26th Annual IEEE Conference on Local Computer Networks (LCN'01), IEEE November 2001, pp.74

[38] R.B Cooper, "Queues Served in Cyclic Order: Waiting Times", The Bell System Technical Journal, Vol.49, No.3, March 1970, pp.399-413

[39] S. Salsano, L. Veltri and D. Papalilo, " SIP Security Issues: The SIP Authentication Procedure and its Processing Load", Network IEEE November/December 2002, vol. 16, Issue 6, pp.38-44

[40] J.W Jung, R. Mudumbai, D. Montgomery and H.K. Kahng, "Performance Evaluation of Two Layered Mobility Management using Mobile IP and Session Initiation Protocol", Global Telecommunication Conference 2003 (GLOBECOM'03), IEEE 2003, Vol. 3, pp.1190-1194

[41] T.T. Kwon, M. Gerla, S.K. Das and S. Das, "Mobility Management for VoIP Services: Mobile IP vs SIP", IEEE Wireless Communications, Vol. 9, IEEE October 2002, pp.66-75

[42] A.T. Campbell, J. Gomez, S. Kim, A.G. Valko and C Wan, "Design, Implementation, and Evaluation of Cellular IP", IEEE Personal Communications, vol. 7, no. 4, August 2000, pp. 42-49

[43] J. Rosenberg and H Schulzrinne, "SIP: Locating SIP Servers", RFC 3263, June 2002

[44] P. Chown, “Advanced Encryption Standard (AES) Cipher suites for Transport Layer Security (TLS)”, RFC 3268, June 2002

[45] T. Dierks and C. Allen, “The TLS Protocol Version 1.0”, RFC 2246, January 1999.

[46] S. Kent and R Atkinson, “Security Architecture for the Internet Protocol”, RFC 2401, November 1998

[47] ETSI Telecommunications and Intranet Protocol Harmonization Over Networks (TIPHON), End to End Quality of Service in TIPHON System; Part2: Definition of Quality of Service (QoS) Classes, TS101 329-2 v.1.1.1, July 2000

[48] E.T. Lakay, J.I. Agbinya, “Communication Cost of SIP Signalling in Wireless Networks and Services”, 12th International Conference and Telecommunications (ICT2005), 3-6 May 2005

[49] E.T Lakay, J.I. Agbinya, “Security Issues in SIP Signaling in Wireless Networks and Services”, Proc. IEEE The Fourth International Conference on Mobile Business (ICBM2005), 11-13 July 2005, pp.639-642

[50] V. Piroumian, “Wireless J2ME Platform Programming”, Sun Microsystems Press A Prentice Hall Title, Release 1.0, March 2002, pp.1-349

[51] SIP API for J2ME Specification, JSR180, Version 1.0.1, 17 November 2004

[52] Mobile Information Device Profile Specification, JSR118, Version 2.0, 5 November 2002

[53] J2ME Connected Limited Device Configuration, JSR30, 30 May 2000

[54] Mobile Information Limited Device Profile for the J2ME Platform, JSR37, Version 1.0, 19 September 2000

[55] Connected Limited Device Configuration 1.1, JSR139, Version 2.1, 27 March 2003

[56] User's Guide J2ME Wireless Toolkit, Version 2.2, October 2004

[57] E.T. Lakay and J.I. Agbinya, "Wireless SIP J2ME Application Development for Mobile Devices", South African Telecommunication Networks and Applications Conference 2005 (SATNAC 2005), 11-14 September 2005

[58] E.T. Lakay and J.I. Agbinya, "SIP-Based Content Development for Wireless Mobile Devices", 1st International Conference on Computers, Communications, and Signal Processing with Special Track on Biomedical Engineering 2005 (CCSP'05), 14-16 November 2005

Appendix I Data Sample of LAN and WLAN Testing Scenarios

	5	10	20	40	80	160
Peak Average Delay Time (ms)	82.7	99.9	91.7	100.2	117.1	156.2
Off-Peak Average Delay Time (ms)	70.4	65.5	65.7	83.1	98.4	107.7

Table 14: LAN Scenario 1 Data Sheet

	5	10	20	40	80	160
Peak Average Delay Time (ms)	134.5	103.1	310.8	97	104.9	128.1
Off-Peak Average Delay Time (ms)	79.7	78.2	86.3	86	97	121.9

Table 15: LAN Scenario 2 Data Sheet

	5	10	20	40	80	160
Peak Average Delay Time (ms)	109.3	123.5	157.8	106.2	131.1	134.4
Off-Peak Average Time (ms)	90.6	87.5	96.7	93.6	103.3	128.1

Table 16: LAN Scenario 3 Data Sheet

	5	10	20	40	80	160
Peak Average Delay Time (ms)	89.2	81.2	76.4	84.4	104.8	124.9
Off-Peak Average	73.5	71.7	73.4	84.5	98.5	121.8

Delay Time (ms)						

Table 17: WLAN Scenario 1 Data Sheet

	5	10	20	40	80	160
Peak Average Delay Time (ms)	71.8	79.8	135.9	129.9	101.7	104.7
Off-Peak Average Delay Time (ms)	79.6	104.9	74.9	87.7	102.9	118.7

Table 18: WLAN Scenario 2 Data Sheet

	5	10	20	40	80	160
Peak Average Delay Time (ms)	79.6	73.4	75.3	162.3	82.8	114
Off-Peak Average Delay Time (ms)	150	79.5	76.4	81.1	150	122

Table 19: WLAN Scenario 3 Data Sheet

	5	10	20	40	80	160
Peak Average Delay Time (ms)	109.4	124.9	119	134.3	137.5	148.5
Off-Peak Average Delay Time (ms)	142.2	165.5	203.1	423.5	260.9	690.4

Table 20: WLAN Scenario 4 Data Sheet

	5	10	20	40	80	160
Peak Average Delay Time (ms)	165.6	192.1	121.8	143.9	120.3	125
Off-Peak Average Delay Time (ms)	115.8	117.1	120.4	110.9	110.8	153.2

Table 21: WLAN Scenario 5 Data Sheet

	5	10	20	40	80	160
Peak Average Delay Time (ms)	162.4	146.9	129.8	143.8	157.7	148.4
Off-Peak Average Delay Time (ms)	160.9	126.6	169	218.9	196.8	223.3

Table 22: LAN to WLAN Data Sheet

	5	10	20	40	80	160
Peak Average Delay Time (ms)	124.9	112.3	104.6	165.6	134.6	143.9
Off-Peak Average Delay Time (ms)	101.7	97	103	146.8	145.4	217.2

Table 23: WLAN to LAN Data Sheet