Session Initiation Protocol based Instant Messaging on Mobile Devices

By

Adeola Oluwaseyi Poroye

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

Tuesday, November 11, 2008

Thesis Committee:

Supervisor: ___________________________________
Mr. William Tucker, [terminal degree]
BANG Project Leader and
Senior Lecturer of [Computer Science]

Co-Supervisor: _________________________________
Mr. Michael Norman, [terminal degree]
Senior Lecturer of [Computer Science]

Co-Supervisor: _________________________________
Isabella Venter, [terminal degree]
Head of the Center of Excellence,
Head of Department Computer Science and
Professor of [Computer Science]
KEYWORDS

Voice over Internet Protocol (VoIP)
Instant Messaging (IM)
Session Initiation Protocol (SIP)
Mobile Devices
Internet Engineering Task Force (IETF)
SIP User Agent
Proxy server
Registrar server
SIP Session
SIP Methods
ABSTRACT

The work provided in this honours Thesis is an investigation of how Session Initiation Protocol (SIP) establishes, modifies, and tears down sessions whose participants are connected-directly or via a gateway-to a network (often, but not always, an Internet protocol network).

The SIP user agent server (UAS) - “kiara” with SIP user agent clients (UAC) on personal computer or s60 Smartphone in a SIP network is used to demonstrate this in practical. Thus, features allowing data, audio and/or video features shared.

The project analysis, design, development and implementation are discussed.

The work of this honours Thesis has been focused around communicating with “kiara” using SIP and in so doing a simple session was built to illustrate the concept. This in essence leads to the beginning of developing various other SIP features.
I declare that *SESSION INITIATION PROTOCOL BASED INSTANT MESSAGING ON MOBILE DEVICES* is my own work, and that it has not been submitted for any degree or examination in this or any other university, and that all the sources I have used or quoted have been indicated and acknowledged by complete references.

POROYE ADEOLA OLUWASEYI  
Tuesday, November 11, 2008

Signed .................................
ACKNOWLEDGEMENTS

I wish to express my wholehearted gratitude and reverence to a faithful and covenant-keeping GOD, the creator of Heaven and Earth, thank you.

To my supervisors who provided enormous experience without which I will not have come this far. Thank you, may God bless you with even more wisdom and knowledge.

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. I’m short of words to describe how much you mean to me and how your encouragement led me to this successful ending. A thank you to all of you and love you lots and lots.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
<td>II</td>
</tr>
<tr>
<td>Abstract</td>
<td>III</td>
</tr>
<tr>
<td>Declaration</td>
<td>IV</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>V</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>VI</td>
</tr>
<tr>
<td>List of Figures</td>
<td>VII</td>
</tr>
<tr>
<td>Glossary</td>
<td>VIII</td>
</tr>
<tr>
<td><strong>Chapter I: Introduction</strong></td>
<td>9</td>
</tr>
<tr>
<td>1.1 Statement of Thesis</td>
<td>9</td>
</tr>
<tr>
<td>1.2 Objective and Motivation</td>
<td>9</td>
</tr>
<tr>
<td>1.3 Literature Review</td>
<td>9</td>
</tr>
<tr>
<td>1.4 Project Description</td>
<td>10</td>
</tr>
<tr>
<td>1.5 Approach and Methodology</td>
<td>10</td>
</tr>
<tr>
<td>1.6 Thesis Layout</td>
<td>11</td>
</tr>
<tr>
<td><strong>Chapter II: User Requirements Document (URD)</strong></td>
<td>12</td>
</tr>
<tr>
<td>2.1 User’s View of the Problem</td>
<td>12</td>
</tr>
<tr>
<td>2.2 A Brief Description of Problem</td>
<td>12</td>
</tr>
<tr>
<td>2.3 A Complete Description of the Problem</td>
<td>12</td>
</tr>
<tr>
<td>2.4 What the Software Solution is Capable of Doing</td>
<td>12</td>
</tr>
<tr>
<td>2.5 What the Software Solution is Not Capable of Doing</td>
<td>12</td>
</tr>
<tr>
<td><strong>Chapter III: Requirements Analysis Document (RAD)</strong></td>
<td>13</td>
</tr>
<tr>
<td>3.1 Technical Interpretation of the User’s Problem</td>
<td>13</td>
</tr>
<tr>
<td>3.2 Breakdown of Problem into High Level Constituent Parts</td>
<td>13</td>
</tr>
<tr>
<td>3.3 Deep Analysis of Parts and Identification of Relevant Details</td>
<td>14</td>
</tr>
<tr>
<td>3.4 Identifying Existing Approaches</td>
<td>18</td>
</tr>
<tr>
<td>3.5 Linking Solutions to Problem(s) with Respect to Details</td>
<td>18</td>
</tr>
<tr>
<td>3.6 Ways to Test for Solution</td>
<td>19</td>
</tr>
<tr>
<td><strong>Chapter IV:</strong></td>
<td>21</td>
</tr>
<tr>
<td>A. User Interface Specification (UIS)</td>
<td>21</td>
</tr>
<tr>
<td>4.1 Description of Complete Solution</td>
<td>21</td>
</tr>
<tr>
<td>4.2 Behaviour of the User Interface</td>
<td>22</td>
</tr>
<tr>
<td>4.3 User Interaction with the System</td>
<td>22</td>
</tr>
<tr>
<td>B. Programmatic Interface Specification (PIS)</td>
<td>23</td>
</tr>
<tr>
<td><strong>Chapter V:</strong></td>
<td>24</td>
</tr>
<tr>
<td>Object Oriented Analysis (OOA or High Level Design)</td>
<td>24</td>
</tr>
<tr>
<td>5.1 Data Dictionary of Each Object Representation</td>
<td>24</td>
</tr>
<tr>
<td>5.2 Class Diagrams</td>
<td>25</td>
</tr>
<tr>
<td>5.3 Class Relationships</td>
<td>25</td>
</tr>
</tbody>
</table>
5.4 A set of class diagrams and data dictionary for application domain

CHAPTER VI: A set of class diagrams and data dictionary for application domain

A. OBJECT ORIENTED DESIGN (OOD OR LOW LEVEL DESIGN)

6.1 INNER DETAILS OF CLASS ATTRIBUTE AND FUNCTIONS

6.2 DETAILED OBJECT DIAGRAM FOR OOD

6.3 STATE DIAGRAMS

Case #1: Basic Call

PSEUDO-CODE

Using PJSIP Replaces Support

6.4 ALGORITHMIC DESCRIPTIONS

Initialization

UAC Behavior: Sending a Replaces Header

UAS Behavior: Receiving a Replaces Header

Function Documentation

References

CONCLUSION

IN CONCLUSION, THE REQUIREMENT OF THIS THESIS WAS TO COME UP WITH A

REFERENCES

LIST OF FIGURES

Figure 1: A Simple Message Flow: depicting a request message

Figure 2: Call flow diagram (partial)

Figure 3: PROVIDES a specialized user interface (UI) to capture this extra information

Figure 4: Blocked call

Figure 5: connected call the SIP messages as they appear in the original call flow (Figure 6).

Figure 6: ACTION DETAILS IN A STATE

Figure 7: SIP network

Figure 8: SIP message exchange for VoIP SIP calls

Figure 9: SIP message examples

Figure 10: command line sip uac

Figure 11: during execution

Figure 12: during execution

SIP-Based IM on Mobile Devices
GLOSSARY

SIP: Session Initiation Protocol
IETF: Internet Engineering Task Forces
IP: Internet Protocol
RFC: Request For Comment
UAS: User Agent Server
UAC: User Agent Client
IM: Instant Messaging
IMS: IP Multimedia Subsystem
CHAPTER I: INTRODUCTION

1.1 STATEMENT OF THESIS

This research question that the thesis tries to answer is, “Can a SIP User Agent Client (UAC) be built to create session with another user agent client through a SIP server such as the “kiara”?”

1.2 OBJECTIVE AND MOTIVATION

Objective:

The aim is to build a SIP User Agent Client (UAC) on a mobile device that creates session with a SIP User Agent Server called “kiara”. The benefits includes; platform independence, data, audio and video functionality of Desktop personal computer (PC) onto mobile devices. Thus, providing some aspects of SIP and/or Media features such as multiple lines or identities (account registrations).

Motivation:

In 2001, SIP was an upstart in the voice over internet protocol (VoIP) and multimedia communications industry. Today, SIP is seen to be the future of call signaling and telephony. It has been widely deployed by service providers (SP) and enterprises and is used casually every day by users of the dominant (PC) operating system (OS). The full range of possibilities is just now been glimpsed, and many more possibilities are yet to come [1].

Aside the call signaling capabilities, it has an incredibly powerful call control protocol allowing intelligent end points to implement the entire suite of telephony, Private Branch Exchange (PBX), Class and Centrex services without a SP, and without a controller or switch.

The biggest driver for SIP on the internet, however, is due to the extensions of SIP. They consist of a set of powerful “rendezvous” protocol leveraging mobility and presence that allows users communicate using different modes, devices and services anywhere they are connected to the internet. SIP applications provide support for presence-the ability to obtain status or location of a user without setting up a session [1].

Furthermore, the adoption of wireless SIP to enable multimedia IP communications and the impressive growth in terms of the specification itself induces for exploring SIP-based applications. Hence, the idea of SIP UACs forming sessions with SIP kiara server developed.

1.3 LITERATURE REVIEW

SIP-Based IM on Mobile Devices
Background

In order to assimilate telephony services with the ubiquitous technology of IP, a signaling protocol is required to set up and tear down connections. The Internet community wanted to introduce innovative services based on enhanced web-authoring tools like The Extensible Markup Language (XML) and more open, peer-to-peer protocols and call models. The Internet Engineering Task Force (IETF) offered SIP.

The SIP approach exemplifies classic Internet-style innovation: build only what you need, to address only what is lacking in existing mechanisms. Because the SIP approach is modular and free from underlying protocol or architectural constraints, and because the protocols themselves are simple, SIP has caught on as an alternative to H.323 and to vendor-proprietary mechanisms for transporting SS7 protocols over IP.

1.4 PROJECT DESCRIPTION

Description

A softphone UAC on a PC will attempt to start session with a kiara client running on a PC and vice versa and similarly, another UAC running on an s60 smartphone creates a session with kiara. To extend this concept multimedia sessions between UACs is created. Hence, demonstrating the various usage and features of SIP based IM applications on mobile devices.

Tools

The following are required for a proper set-up to take place; Symbian E60 smartphone with cable connection to PC, wireless and wireline Network connection to kiara server, a PC.

In addition, the following software are required an operating system, Carbide.c++1.2/1.3, Microsoft Visual Studio 2005 and it’s SDK , PJSIP - Open Source SIP Stack or any other SIP stack will do and Symbian S60 3rd Edition Device as SDK.

1.5 APPROACH AND METHODOLOGY

Approach

The requirement gathering involves carrying out semi-structured interviews with users, especially developers in various software development companies e.g. Blue Chip Institute of Technology (BCIT) and Information Communication Section of the University of The Western Cape.

This process is followed by conducting analysis on information gathered, constructing a high level design (HLD), a low level design (LLD) and the actual implementation.
Some of the work products include; class, state and activity diagrams, prototype-simulation of actual product, and a working implementation.

**Methodology**

This involves developing a SIP-based UAC/S application on PC using “c/c++” language and another UAC/S on the mobile device using a variant of “c/c++” called “Symbian.c++” language. After configuring a SIP session via kiara, porting data, voice and video capabilities to and fro the desktop PC and mobile device becomes possible.

1.6 **THESIS LAYOUT**

The User Requirement Document (URD) on the entire system and breakdown is discussed. In addition, it identifies the actual details of the problem unknown to the users.

Chapter III describes exactly what the user interface is going to do, its Look’n feel, and how the user is going to interact with the program.

Chapter IV provides that Object Oriented Analysis, as it applies an Object-Oriented (OO) view of the problem. It uses C++ and OO design techniques.

In chapter V does the following: moves the OOA into pseudo-code level, identifying the objects, state and event diagrams, and providing algorithms.

Last, chapter VI offers a work schedule for accomplishing the rest of the project for the duration as stated in this course.

SIP-Based IM on Mobile Devices
CHAPTER II: USER REQUIREMENTS DOCUMENT (URD)

2.1 USER’S VIEW OF THE PROBLEM

Users are not able to originate, receive incoming calls and carry multimedia session on various phones without the thought of a service provider.

Users want to be able to integrate multiple types of media.

2.2 A BRIEF DESCRIPTION OF PROBLEM

Users want data, audio and/or video sessions developed with kiara on mobile phones and personal computer desktop and with little or no cost involved.

2.3 A COMPLETE DESCRIPTION OF THE PROBLEM

For example, users had the difficulty of receiving an IM on their phone screen that they would like to respond to verbally. Once they start to press the button to call the IM sender, they see that the sender has set his status to "Don't call me," so they refrain.

Again, they are not able to have simultaneous calls or multiple sessions i.e. three or more other people, as one can guess that one of them will have his phone off-hook.

A list of probes obtained during interviews conducted provides more of the problems faced by users. See table 1 for a list of probes and their response.

2.4 WHAT THE SOFTWARE SOLUTION IS CAPABLE OF DOING

It is able to provide data and voice session among UACs and simultaneously. It is capable of creating session among more than two UACs, hence more than two participants.

2.5 WHAT THE SOFTWARE SOLUTION IS NOT CAPABLE DOING

It will not do or provide the following functionalities;

1. Adding capabilities like print, fax or videoconference to kiara.
2. Modifying or extending kiara beyond its minimal basic application features or even mobile.
CHAPTER III: REQUIREMENTS ANALYSIS DOCUMENT (RAD)

3.1 TECHNICAL INTERPRETATION OF THE USER’S PROBLEM

From a VoIP point of view the following problem arises

a. User location and registration
b. User availability
c. User capabilities
d. Session setup
e. Session management

3.2 BREAKDOWN OF PROBLEM INTO HIGH LEVEL CONSTITUENT PARTS

- Who participates in the call?
- Decision as who “answers” a call.
- An agreed choice of codec?
- How do we know when the phone is ringing and also, how to agree on session attributes used?
- How to transfer calls, change call parameters in the mid-session?

The Figures below illustrates the concept- modeling SIP message and SIP Methods involved.

**SIMPLE**

**Session Mode Messaging**

![Diagram](image)

**FIGURE 1: A SIMPLE MESSAGE FLOW: DEPICTING A REQUEST MESSAGE**

SIP-Based IM on Mobile Devices
This example illustrates the use of request messages INVITE, ACK, and BYE, as well as the 200 OK response messages. See pics1 for a typical SIP call framework [1].

### 3.3 Deep Analysis of Parts and Identification of Relevant Details

**Call flow modeling**

One of the most common ways to communicate a specific scenario of activity between user agents and SIP services is with a call flow diagram. These diagrams are, for the most part, UML sequence diagrams, as shown in Figure 2. The SIP User Agent (UA) typically initiates the call flow with an INVITE message to a SIP-aware service. This service usually forwards the messages on to other SIP services, or translates the messages and invokes other types of equipment (like the call control elements in the control plane).

![Session Establishment Through Two Proxies](image)

**FIGURE 2: CALL FLOW DIAGRAM (PARTIAL)**

Most of the information exchanged between the participants is in the header content.
Call flow merging into state machines

In the case of multiple media sessions occurring simultaneously, one can model this concept with a workflow as done in Figures 1-3 above.

Take as an example the two following call flow diagrams (Figure 4). In the first scenario an attempted call is blocked, and an instant message is sent to the intended caller. In the second the call flow the call is proxied. Some of the messages in the call flows are optional.

By tagging a lifeline with a common keyword in each diagram, we define the focus of the combined behavior. With the common lifelines tagged we can generate the emergent state machine. This machine (Figure 5) represents the combined behavior of all the call flow diagrams into a single machine.
FIGURE 4: BLOCKED CALL

FIGURE 5: CONNECTED CALL

FIGURE 6: MERGED STATE MACHINE
Figure 5 can be examined in detail. Selecting the triggers on the events that cause state transitions shows the details of the SIP messages as they appear in the original call flow (Figure 6).

The entry actions inside the states (Figure 7) provide a summary of the outgoing messages that the service may instigate (depending upon the specific scenario). This state machine is a key artifact of understanding in the design process of developing the service.
3.4 IDENTIFYING EXISTING APPROACHES

Windows Messenger: The Microsoft Approach

a) SIP-based
b) Runs on Windows XP systems (sorry no Macs)

c) Core Features:
   • Presence and contact list management
   • Instant Messaging
   • Voice and Video
   • Data collaboration and File transfer
   • PC to Phone
   • Administrative Policies to enable/disable

3.5 LINKING SOLUTIONS TO PROBLEM(S) WITH RESPECT TO DETAILS

- UACs should notify SIP proxies of their location and SIP determines which one will participate in a call.
- UACs should use SIP to decide whether they will “answer” a call.
• UACs should use SIP to negotiate media capabilities, such as agreeing on a mutually supported voice codec.
• SIP should know when the phone should be “ringing;” and UACs should use SIP to find out how to agree on session attributes used by the calling and called party.
• UACs should use SIP to transfer calls, terminate calls, and change call parameters in mid-session (such as adding a 3-way conference).

### 3.6 WAYS TO TEST FOR SOLUTION

By examining the content of the SIP User Agents clients when calls are made or messages sent. An example will be

![SIP Network Diagram](image)

**FIGURE 7: SIP NETWORK**

We have the following SIP network setup and the SIP messages are:

- **REGISTER**: for recording a SIP User Agent to the Registrar Server.
- **INVITE**: for setting a session between 2 SIP User Agents.
- **CANCEL**: for cancelling an INVITE request.
- **ACK**: for acknowledging an INVITE request.
- **BYE**: for closing a SIP session.
- **SUBSCRIBE, NOTIFY**: SIP event subscription and notification.
- **MESSAGE**: instant messaging.
- **Others**: REFER, OPTIONS, INFO.

The use case will for the SIP message calls will be as shown in Figure 8 below
Figure 8: SIP Message Exchange for VoIP SIP Calls

In a text format, Figure 9 better shows the exchange taking place:

- **SIP INVITE request:**
  
  ```
  INVITE sip:laura@home.com SIP/2.0
  Via: SIP/2.0/UDP pc11.work.com
  Max-Forwards: 70
  To: Laura <sip:laura@home.com>
  From: Bob <sip:bo@work.com>;tag=1928301774
  Call-ID: a84b4c76e66710
  CSeq: 314159 INVITE
  Contact: <sip:bo@pc11.work.com>
  Content-Type: application/sdp
  Content-Length: 131
  
  o=Bob 289123451 289123451 IN IP4 111.22.33.44
  s=Let us talk for a while
  c=IN IP4 111.22.33.44
  t=0 0
  m=audio 200002 RTP/AVP 0
  a=rtpmap:0 PCMU/8000
  ```

- **SIP Response:**
  
  ```
  SIP/2.0 200 OK
  To: Laura <sip:laura@home.com>;tag=a6c85cf
  From: Bob <sip:bo@work.com>;tag=1928301774
  Call-ID: a84b4c76e66710
  CSeq: 314159 INVITE
  Contact: <sip:laura@222.33.44.55>
  Content-Type: application/sdp
  Content-Length: 131
  
  v=0
  o=Laura 289123444 289123444 IN IP4 222.33.44.55
  s=Let us talk for a while
  c=IN IP4 222.33.44.55
  t=0 0
  m=audio 41002 RTP/AVP 0
  a=rtpmap:0 PCMU/8000
  ```

Figure 9: SIP Message Examples
The content of Figure 9 describes in detail Figure 7 and 8, and in human readable text allowing one to validate and verify results of SIP methods and message calls.

CHAPTER IV:

A. USER INTERFACE SPECIFICATION (UIS)

4.1 DESCRIPTION OF COMPLETE SOLUTION

USER INTERFACE (UI)

Simply put, the UI will provide graphical menu, submenus, navigational path, options and functionalities for traversing the application. The application on the other hand carries out the porting of kiara onto mobile, thus, making the user feel that the UI does the actual action of making a mobile kiara.

WHAT THE UI WILL LOOK LIKE

Start the PC UAC by compiling and executing, once done your screen should look like figure 10 below.

FIGURE 10: COMMAND LINE SIP UAC

Similarly, start mobile UAC and confirm that the kiara application (see Figure 11) is running and that there is connection.

SIP-Based IM on Mobile Devices
Figure 11 shows registration on kiara.

HOW THE USER WILL INTERACT WITH THE UI

Steps;

1. Register an account on kiara example will be sip:101@172.16.39.100
2. Choose m on PC UAC to make calls,
3. press Enter and type sip:101@172.16.39.100
4. Enter

Similarly, carry out steps 2 to 4 on mobile UAC which will both create session with kiara.

4.2 BEHAVIOUR OF THE USER INTERFACE

The confirmation will be data and voice over IP (as in ringing and sound) that can be seen and heard on the end point that kiara is installed on.

4.3 USER INTERACTION WITH THE SYSTEM

There will be accept and reject button shown, once a click to accept is made, the accept button changes to a hang up button and both endpoints can begin their session. To end simple click the hang up to end calls.

In addition, the IP address and tag number of opposing endpoint’s machine is indicated on each other graphical user interfaces. The content will look very similar to SIP message responses as in figure 9 above and very importantly messages are tagged.
B. PROGRAMMATIC INTERFACE SPECIFICATION (PIS)

Figure 11 shows the audio rate during execution.

This time current call identification, the SIP uniform resource locator (url) and indication showing the state, an SIP invite method, user datagram protocol is used, the port and the branch, source, destination, contact, call-id and various SIP methods enabled and more information are clearly detailed in Figure 12 during the execution.

SIP-Based IM on Mobile Devices
CHAPTER V:

OBJECT ORIENTED ANALYSIS (OOA OR HIGH LEVEL DESIGN)

5.1 DATA DICTIONARY OF EACH OBJECT REPRESENTATION

The Data dictionary produce high level entities when broken down, called SIP Entities.

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile Kiara – SIPmobileIM-uac</td>
<td>UAC is a SIP-enabled end-device that initiates a session. It maintains call states, which it initiates or participates in. Even if a session has been terminated, call state must be maintained by the UAC for at least 32 seconds to accept responses in case of lost messages in the call termination.</td>
</tr>
<tr>
<td>Kiara Server - SIPmobileIM-uas</td>
<td>UAS is a SIP-enabled server application that contacts a user when a request is received and responds on behalf of the user. It can accept, reject or redirect a request. UAS too maintains the call state as mentioned for the UAC.</td>
</tr>
<tr>
<td>Proxy Server (PS)-SIPmobileIM</td>
<td>It is an intermediary entity that acts as both client and server. It can make requests on behalf of other clients and service a received request internally or by passing them to other proxies possibly after translation. A proxy server has no media capabilities. It can either be a stateless server or a stateful server. A stateless server never stores any information about a message once it has been parsed, processed or responded. All the processing is done sheer based on the message contents. It never uses any timers nor retransmits a message. A stateful proxy server, on the contrary, maintains information of all the requests and responses and uses it for future processing. It is capable of retransmitting messages and maintains timers. A proxy is used only to help the caller user agent to locate the called party (if they don’t know each other apriori) and once a connection is established then proxy does not participate in the call. The two parties can communicate directly.</td>
</tr>
<tr>
<td>Redirect Server (RS)- SIPmobileIM-simple</td>
<td>It receives a SIP request and returns zero or more possible locations for the called party. No message forwarding is done. It neither initiates a request unlike the UAC nor accepts a call unlike the UAS.</td>
</tr>
<tr>
<td>Registrar Server (RGS)- SIPmobileIM-lib</td>
<td>A registrar server accepts SIP REGISTER requests. It can be co-located with a proxy server or a redirect server. It provides the information about the registered SIP user agents to other SIP servers within the same administrative domain. A user agent must register with the registrar if it intends to receive calls. Registration is not necessary for making outgoing calls.</td>
</tr>
</tbody>
</table>

TABLE 11: THE FOLLOWING PRIMARY ENTITIES HAVE BEEN DEFINED [5]
5.2 CLASS DIAGRAMS

The class diagram shows the SIPmobileIM is a generalization of i SIPmobileIM-uas, SIPmobileIM-uac, SIPmobileIM-simple while SIPmobileIM-test has an association with SIPmobileIM-uas and SIPmobileIM-lib have association with SIPmobileIM (see figure 25 below).

5.3 CLASS RELATIONSHIPS

We filter the Data Dictionary (DD) to obtain those important to our SIP entities include; UAC, UAS, PROXY (SIPmobileIM), REDIRECT (SIPmobileIM-simple), and REGISTRAR (SIPmobileIM-lib).

SIP-Based IM on Mobile Devices
SIPmobileIM-lib associates with SIPmobileIM

SIPmobileIM-test associates with SIPmobileIM-uas

All other class inherit from SIPmobileIM

Additional information

UAC:
- Creates a new request, and then uses the client transaction state machinery to send it.
- The role of the UAC lasts only for the duration of that transaction

UAS:
- generates a response to a SIP request
- This role lasts only for the duration of that transaction.

Server: uses response to indicate progress

SIP Server: deals with setup of all calls

Registrar Server: Provides registration services for UACs for their current locations.

Registrar: Allowing a user to tell the server how to map an incoming address into an outgoing address that will reach the user

See table 12 above for a more explicit information.

5.4 A SET OF CLASS DIAGRAMS AND DATA DICTIONARY FOR APPLICATION DOMAIN

- It is an intermediary entity that acts as both client (mobile phone) and server (kiara/SIP server). It can make requests on behalf of other clients
- It can service a received request internally or by passing them to other proxies possibly after translation.

A proxy server has no media capabilities. It can either be a stateless server or a stateful server.

1. A stateful proxy server, on the contrary, maintains information of all the requests and responses and uses it for future processing. It is capable of retransmitting messages and maintains timers.
2. A stateless server never stores any information about a message once it has been parsed, processed or responded. All the processing is done sheer based on the message contents. It never uses any timers nor retransmits a message.

A proxy is used only to help the caller user agent to locate the called party (if they don’t know each other apriori) and once a connection is established then proxy does not participate in the call. The two parties can communicate directly (see figure below).
The figures above; (collaborative and use case diagram) help provide a different views of looking how the server is related in and out with respect to functions such as requesting and responding to request.
A. OBJECT ORIENTED DESIGN (OOD OR LOW LEVEL DESIGN)

6.1 INNER DETAILS OF CLASS ATTRIBUTE AND FUNCTIONS

SIP has the capabilities to implement reliability in transporting SIP messages hence it can use unreliable UDP.

SIP is a lightweight, text-based, application layer signaling protocol for multimedia call control. It is used to create, modify and terminate sessions with one or more participants. These sessions include, but not limited to, Internet multimedia conferences, IP telephony etc. Signaling is achieved based on entities like SIP clients and servers, well-defined SIP request messages, SIP response messages, included SIP headers, related protocols like SDP, RTP etc.

The SIP entities can be mapped to figure 26 based on their operations and attributes. In addition, there are six defined methods SIP Requests: SIP entities defined [5] and well are listed and described in table 1, above.

There are six SIP requests defined methods:

**INVITE:** is used to set up multimedia sessions between two parties (mobile device and kiara on PC)

**REGISTER:** is used by a user agent to register it with the registrar notifying it of its contact address where it can receive calls.

**BYE:** is used to terminate an established session and is always initiated by the participating user agents.

**ACK:** is used to acknowledge the final responses to INVITE methods. Other final responses are never acknowledged.

**OPTIONS:** is used to find out the capabilities and availability of a UA or a server. It is always initiated by a UA and not by a proxy.

**CANCEL:** is used to “tear down” a pending INVITE

6.2 DETAILED OBJECT DIAGRAM FOR OOD

Sub-protocol

The following sub-protocol has been mentioned in the [5]:

Registration (2 Phase Protocol): To register a UA (i.e. kiara and mobile devices) with a registrar server. The UA will be able to register to a local SIP server by sending a request to a multicast address “sip.multicast.net” (224.0.1.75). Same UA can register from different locations. Third party registration is also allowed. The requests are processed in the order they are received.
Session Termination (2 Phase Protocol): To terminate an established session by sending a BYE message. Either caller or the callee can initiate this request. The UA agent receiving a BYE request must stop transmitting data.

Invite Request Cancellation (2 Phase Protocol): To cancel a pending request other than ACK and CANCEL e.g., REGISTER, INVITE, OPTION but typically it is used to cancel a pending INVITE request.

Session invitation and modification are not really used here as they are for 3 Phase Protocols beyond this project.

This is the basic SIP call model with the most simple offer-answer exchange.

A(uac-kiara)      B(uas-mobile device)
|<-----INVITE (offer) ----> |
|< -- 180 Ringing ------|
|<------200(answer) ------|
|---------------ACK---------->

Figure 27: a simple kiara to mobile device

PSEUDO-CODE

Name: Poroye Adeola O.
Student Number: 2561906
Term II: Project
Title: Algorithm/Pseudo-code for the SIP User Agent (SIP UAC [for either 1 or 2]) when using PJSIP Replaces Support
Supervisor: Bill Tucker

SIP-Based IM on Mobile Devices
Please do note that all references in this section have been obtained from PJSIP reference manuals [5].
The pseudocode below illustrates how an application can process the incoming INVITE if it wants to support the Replaces extension:

```c
// Incoming INVITE request handler
invite(data) {
    Dialog *dlg, *replaced_dlg;
    Inv_session *inv;
    Tx_data *response;
    pj_status_t status;

    // Check whether Replaces header is present in the request and process accordingly.
    //
    status = verify_request (data, &replaced_dlg, PJ_FALSE, &response);
    if (status != PJ_SUCCESS) {
        // Something wrong with Replaces request.
        //
        if (response) {
            send_response (endpt, data, response, NULL, NULL);
        } else {
            // Respond with 500 (Internal Server Error)
            respond_stateless (endpt, data, 500, NULL, NULL, NULL);
        }
    }

    // Create UAS Invite session as usual.
    //
    status = create_uas (.., data, .., &dlg);
    ...
    status = create_uas (dlg, .., &inv);

    // Send initial 100 "Trying" to the INVITE request
    //
    status = inv_initial_answer(inv, data, 100, ..., &response);
    if (status == PJ_SUCCESS)
        inv_send_msg (inv, response);

    // This is where processing is different between normal call
    // (without Replaces) and call with Replaces.
    //
    if (replaced_dlg) {
        inv_session *replaced_inv;

        // Always answer the new INVITE with 200, regardless whether
        // the replaced call is in early or confirmed state.
        //
        status = inv_answer(inv, 200, NULL, NULL, &response);
        if (status == PJ_SUCCESS)
            inv_send_msg (inv, response);

        // Get the INVITE session associated with the replaced dialog.
        //
        replaced_inv = dlg_inv_session (replaced_dlg);

        // Disconnect the "replaced" INVITE session.
        //
        session.
    }
}
```
6.4 ALGORITHMIC DESCRIPTIONS

INITIALIZATION

Application needs to call replace_initialization() during application initialization stage to register "replaces".

UAC BEHAVIOR: SENDING A REPLACES HEADER

A User Agent that wishes to replace a single existing early or confirmed dialog with a new dialog of its own, MAY send the target User Agent an INVITE request containing a Replaces header field. The User Agent Client (UAC) places the Call-ID, to-tag, and from-tag information for the target dialog in a single Replaces header field and sends the new INVITE to the target.

To initiate outgoing INVITE request with Replaces header, application would create the INVITE request with invite (), then adds replace_header() instance into the request, filling up the call-Ident, To_kiara, and From-kiara properties of the header with the identification of the dialog to be replaced.

The outgoing INVITE request (with Replaces) initiated from an incoming REFER request (as in Attended Call Transfer case) should be done automatically. Upon receiving incoming REFER request, normally these processes will be performed:

- Application finds Refer-To header.
- Application creates outgoing dialog/invite session, specifying the URI in the *Refer-To* header as the initial remote target,
- The URI in the *Refer-To* header may contain header parameters such as *Replaces* and *Require* headers.
- The dialog keeps the header fields in the header parameters of the URI, and the invite session would add these headers into the outgoing INVITE request. Because of this, the outgoing INVITE request will contain the *Replaces* and *Require* headers.

Please see the implementation of replaces() source code.

**UAS BEHAVIOR: RECEIVING A REPLACES HEADER**

The Replaces header contains information used to match an existing SIP dialog (call-id, to-tag, and from-tag). Upon receiving an INVITE with a Replaces header, the User Agent (UA) attempts to match this information with a confirmed or early dialog.

If application wants to process the Replaces header in the incoming INVITE request, it should call verify_request() before creating the INVITE session. The verify_request() function checks and verifies the request to see if Replaces request can be processed. To be more specific, it performs the following verification:

1. It checks that Replaces header is present. If not, the function will return SUCCESS without doing anything.
2. It checks that no duplicate Replaces headers are present, or otherwise it will return 400 "Bad Request" response.
3. It checks for matching dialog and verifies that the invite session has the correct state, and may return 481 "Call/Transaction Does Not Exist", 603 "Declined", or 486 "Busy Here" according to the processing rules specified in [5].
4. If matching dialog with correct state is found, it will give SUCCESS status and return the matching dialog back to the application.
status_t replaces_init_module (endpoint *endpt)

Please do note that initialize Replaces support in PJSIP. This would, among other things, register the header parser for Replaces header.

Parameters:
    *endpt The endpoint instance.

Returns:
    SUCCESS on success.

replaces_hdr* replaces_hdr_create (pool_t *pool)

Create Replaces header.

Parameters:
pool Pool to allocate the header instance from.

Returns:
An empty Replaces header instance.

status_t replaces_verify_request ( rx_data* data,
                                 dialog** p_dlg,
                                 bool_t lock_dlg,
                                 tx_data** p_tdata
                           )

Verify that incoming request with Replaces header can be processed. This function will perform all necessary checks according to [5] Section 3 "User Agent Server Behavior: Receiving a Replaces Header".

Parameters:
  data   The incoming request to be verified.
  dlg    On return, it will be filled with the matching dialog.
  lock_dlg Specifies whether this function should acquire lock to the matching dialog. If yes (and should be yes!), then application will need to release the dialog's lock with dlg_dec_lock() when the function returns SUCCESS and the dlg parameter is filled with the dialog instance.
  tdata  Upon error, it will be filled with the final response to be sent to the request sender.

Returns:
The function returns the following:
If the request doesn't contain Replaces header, the function returns SUCCESS and dlg parameter will be set to NULL.
If the request contains Replaces header and a valid, matching dialog is found, the function returns SUCCESS and dlg parameter will be set to the matching dialog instance.
Upon error condition (as described by RFC 3891), the function returns non- SUCCESS (FAILURE), and data parameter SHOULD be set with a final response message to be sent to the sender of the request.
CONCLUSION

In conclusion, the requirement of this thesis was to come up with a

In conclusion, this thesis has carefully shown in a systematic order a software development life cycle from the stages of requirement gathering, analysis and the implementation process.

It starts by introducing the subject matter; seamlessly explain the requirement gathering stages whilst illustrating the URD, RAD, User and Programmatic IS.

Furthermore, it took time to thoroughly dissect the concept of OO of both HLD and LLD while focusing on answering research question, that the thesis tries to answer is, “Can a SIP User Agent Client (UAC) be built to create session with another user agent client through a SIP server such as the “kiara”?"

The discussion took us through classes and their relationships, functions, dictionary, state diagrams. Also, algorithms, pseudo codes and provided sample section of the actual implementation plus function documentation.

In the end it investigates SIP-based IM applications on mobile devices by discussing the use of SIP UACs on mobile phone and PC while in a network with a second PC that has “kiara” a SIP server installed and configured on it. Thus, providing the much needed argument of a working
REFERENCES