IP-BASED PUSH TO TALK ON A MOBILE PHONE

by

Hlabishi Isaac Kobo

A thesis submitted in partial fulfillment of
the requirements for the degree of

Baccalaureous Scientiae (Honours)

University of the Western Cape

2009

Date: November 5, 2009

University of the Western Cape

Abstract

IP-BASED PUSH TO TALK ON MOBILE HANDSET

By

Hlabishi Kobo

Supervisory Committee:
Supervisor: W.D Tucker
Co-Supervisor: M.J Norman
Department of Computer Science

Push-to-talk (PTT) is a new approach to voice communication which emulates walkie-talkie system. The main purpose of this project is to implement PTT on a cellular mobile phone (PoC). PoC is an instant messaging service like a voice SMS. Instead of dialing, you "push" a button and speak. When you release the button the message is sent. Communication is bidirectional but not simultaneous (half-duplex).

The project was implemented on two platforms, PC and mobile phone. The PC PTT was implemented through client server approach whilst the mobile PTT (PoC) through peer-to-peer approach. Several software engineering testing strategies were used during testing. 6 users participated in the test and the results gathered through questionnaires.

The results showed that, half-duplex communication is efficient and yet very economical due it’s less usage of system resources. The PC application used approximately 2% of CPU and about 0.09% of bandwidth.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Firstly I would to thank God almighty for giving the strength and wisdom to study.

I would also like to thank my supervisor Dr Bill Tucker and my co-supervisors Prof. Isabel Venter and Mr. Michael Norman for their guidance. I would also like to extend my sincere gratitude to Mr. Long Yee and Mr. Francois Daniels for their expertise.

To all my colleagues, the long night hours in the laboratory has finally paid off. Thanks for your support and fruitful cooperation. Special thanks to Ryno Hoorn as well as Charles Atuchukwu for his caring advice. Thanks also to my friends Matshipu Manale and Mmuthi Kokhutsa for supporting me throughout the year.

Lastly I would to thank my family for their support.

Hlabishi Isaac Kobo

GLOSSARY

PTT – Push-to-Talk is way of communicating on half-duplex lines using a dedicated button to control the session e.g. Walkie Talkie.

IP-Based PTT – is Push-to-Talk using internet protocol as a communication media.

PoC - Push-to-talk over a cellular is the implementation of walkie-talkie concept of communication on a mobile phone.

RTP - Real Time Protocols is a synchronous multimedia data transfer protocol over the internet.

RTCP - RTP Control Protocol is used to periodically to send quality of service (QOS) feedback and control information.

SIP - Session Initiation Protocol that is used to initiate and terminate unicast and multicast multimedia sessions

Half-duplex – provides a bidirectional communication but not simultaneous (one way).

Full-duplex – allows simultaneous bidirectional communication.

Mxit – is a free instant messaging software application developed in South Africa that runs on GPRS/3G mobile phones and on PCs.

VoIP – voice over internet protocol is family transmission technology that allows voice communication over the internet.

SDK – set of development tools that a developer uses to create applications.

PoC server – relays the voice streams from source to destination.

PoC client – is the end user software used to perform PoC functionalities.

SMS – short message service is a text based communication service on mobile phones.

C h a p t e r 1

INTRODUCTION

INTRODUCTION

IP-based Push to talk over cell phone (PoC) is an IP-based voice communication on a mobile phone. Push to talk is a new instant messaging type of communication. PoC is the implementation of walkie-talkie concept of communication on a mobile phone. PTT is different from the old walkie-talkie system because it operates within the internet network in Voice over Internet Protocol (VoIP) instead of radio frequencies. PoC use half-duplex communication where only one

Transmission of voice messages occurs through the press of a button. Instead of dialing a person “you press a button and speak”. When you release the button the voice message is sent. PoC supports 2.5 and 3 generations of cellular networks as well as future generations of cellular networks. The reason being that PoC is based on internet protocol (IP-based). The transmission of data occurs through an IP packet-switching. Thus the main purpose of this project is to implement PoC over Wi-Fi supported cell phones.

The concept of PoC was introduced in 2003. The standardization of PoC started in 2004 and the first version was finalized in 2005. In June 2006 the first the version OMA PoC 1.0 version was released. Open mobile alliance (OMA) is the official standard that oversees all the infrastructures and processes supporting PoC.

PoC is available on many cellular networks around the world, but not in South Africa. In USA Nextel communications and Motorola network providers offers the service while orange and Vodafone does it for UK.

MOTIVATION

Most South Africans have joined the community mobile instant messaging through Mxit. Mxit is an text based IM, so introducing a voice based IM will enhance the flexibility of interest in the Mobile IM's. IP-based Push to Talk on a mobile phone is cost effective due to its half-duplex nature of communication. This kind of communication makes it to consume less bandwidth. The application among other things is fast to process making it suitable to be used in variety of situations such as emergencies, work place and etc.

C h a p t e r 2

USER REQUIREMENTS DOCUMENT

INTRODUCTION

In this chapter User Requirement Document (URD) is discussed. We look at the user's view of the problem, brief description of the problem domain as well as a complete description of the problem domain. The expectations of the software capabilities are discussed, together with what the software is not expected to do. The URD was obtained by conducting a survey. More than ten people from different faculties at the University of the Western Cape were given questionnaires.

USERS VIEW OF THE PROBLEM

All most all the instant messaging services offered on a mobile phone are text-based. According to the survey carried out, many users are still using short messaging services (sms). However users want a convenient way of exchanging messages. According to most of them, sms are not convenient in urgent situations due to the amount of time it takes to process. Another outcry of the users is the fact that current services are not economically friendly. A text IM can only be sent to one user, group chat are not supported.

BRIEF DESCRIPTION OF THE PROBLEM DOMAIN

Most users, youth in particular want to engage in social interactive communications with friends, family, colleagues etc. Users want voice instant messaging in place of text IM's and the service has to be costless. The system must support real time communication without any difficulties. Majority of people have seen push to talk application on their mobile phones. About 70% of the people from the carried survey have the application integrated on their mobile phones but they cannot use it because network service providers do not support it. However Vodacom offers the application for only corporate customers. Some of the areas supported include construction, transport, security, distribution, manufacturing, and surface mining, as well as companies operating in the catering, hospitality and courier industries (Pieter Uys, Vodacom Chief Operating Officer, December 2005.)

COMPLETE DESCRIPTION OF THE PROBLEM

Majority of South Africans are using text-based instant messaging which takes long to process. Users encounter difficulties to apply text instant messaging to in urgent situation. Users often make use of "Language compressing" to enhance the speed of the process as well as reducing the amount of data to be sent. This type of vocabulary turns to lose the contextual meaning of the message. This is mainly because people have different understanding of the language. However this language is effective when used between English literate people. In contrary the English literacy of people in our society is very low. Thus for this reason most people deduce that text based instant messaging is not multilingually friendly. This leads to people not enjoying the optimum courtesy of the service. Among other technical problems caused by the frequent use of text IM's is the key pad.

Text based IM can also pose danger to the society [2] because of the depersonification. "A 35 year old allegedly abducted a 16 year old girl whom he met in the mxit chart room, due to exchanging personal information such as home address"[2].

WHAT THE SOFTWARE SOLUTION IS CAPABLE OF DOING

The software will enable people to engage in instant messaging in the form of voice instead of text. The software is expected to offer availability of service throughout the network domain. The following features are also expected from the software:

Instant one to one communication

Instant group communication

Efficient contact list management e.g. ability to add, delete contacts

Good alert notification

WHAT THE SOFTWARE SOLUTION IS NOT CAPABLE OF DOING

Users should not expect to transmit videos from the final product. The software is presence orientated, so voice mail messages cannot be expected. The transmission of photos is also not supported. The software would only work on IP-based network phones.

The following table 1 shows some of the questions carried out in the survey as well as the results.

Question	Response (%)		
	YES	NO	UNANSWERED
Do you use Instant messaging applications (MSN Messenger, Google Talk, Mxit etc.) and VoIP applications (Skype etc.)?	90	10	
Have you used any PTT system e.g Walkie-Talkie?	10	90	
Have you heard about push to talk application on mobile phone before?	70	30	

Do you know anyone who uses PTT on a cellular phone?	10	60	30
Does your mobile phone have push-to-talk application?	50	20	30
Have you used push-to-talk before?	0	70	30
How often do you use Instant Messaging applications?	10		Do not use it
	20		Once a week
	70		Daily
In which kinds of situations do you think that PoC-service is most useful?	Emergency services		
	Quick messaging		
	Occasions		
	Construction companies		
In what kinds of situations do you think you would be likely to use PoC?	In place of an sms		
	Overcrowded situations e.g. parties, stadium, forest		
	Emergency features again		
	Saving money		
	Students on-campus communication		

Table 1 : The responses of the interviewees are summarized.

C h a p t e r   3

REQUIREMENTS ANALYSIS DOCUMENT

INTRODUCTION

This section analyzes the user requirements from the previous section. This is the designer's view of the problem. The user requirements are interpreted systematically from the designer's perspective. The problem is broken down into high level constituent parts. These parts are deeply analyzed and all relevant details are identified. We identify existing solutions as well as alternative technical solutions. Among the existing solutions, we identify the suitable one to solve our problem. We finally look at ways in which the solution can be tested.

DESIGNERS INTERPRETATION OF THE PROBLEM

The system is Voice over Internet Protocol application. It introduces the transition of text to voice in instant messaging. The project will be implemented using a client-server approach. Mobile phones will carry the PoC-client and the server would act as an interface of communication.

There are basic factors that have to be considered during the implementation. The voice quality is very critical and thus has to be constantly monitored. Real Time Protocols (RTP) used to carry the audio streams. In addition, RTP control protocol would be used to monitor the quality of the voice as well as keeping track of all transmissions.

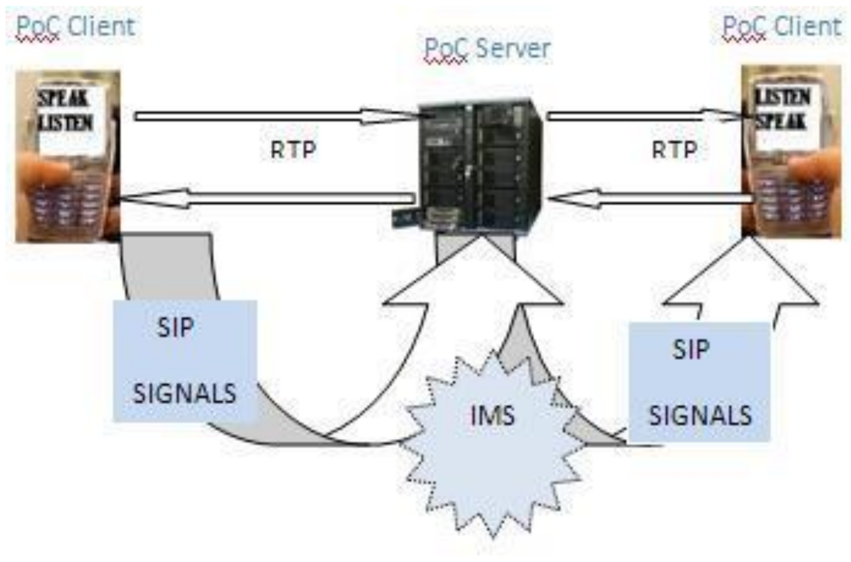


Figure 1: Push to talk concept

The system is IP-based, implemented in a half-duplex communication to ensure the efficiency of the bandwidth. Our push to talk would be using Session Initiation Protocol (SIP) for signaling. User interfaces are required to ensure easy to use system. Contact list interface will be implemented. This will display presence and status information of other users. The presence feature will be managed by SIMPLE protocol. The status information will ensure there are no interruptions.

BREAKDOWN OF PROBLEM INTO HIGH LEVEL CONSTITUENT PARTS

PoC server would be responsible for PTT session setup using SIP, floor control, VoIP steaming (voice distribution) and signal control.

PoC client feature will be integrated on the mobile phone. This feature will consist of user interfaces.

Presence information

Contact list management

Protocols to be deployed

Real Time Protocols (RTP)

RTP Control Protocol (RTCP)

Session Initiation Protocol (SIP)

SIMPLE Protocol

Internet Protocol Multimedia Subsystem (IMS)

SIP application server

DEEP ANALYSIS OF PARTS AND IDENTIFICATION OF RELEVANT DETAILS

PoC server

The PoC server relays the calls from one client (e.g. sender) to the other (e.g. reciever). It controls both the call control and the floor control.

Session setup

Session Initiation (SIP) protocol is an application layer protocol for creating, modifying and terminating sessions with one or more participants [3]. During the session setup, the transmission has to be evaluated to check its validity. Thus SIP includes authentication and authorization to ensuring security. SIP uses SIMPLE protocol to manage the presence information.

Floor control

This feature mediates the PTT calls and decides whether to grant a call or not. One user makes a request to send someone a PTT message. This feature uses RTCP to give permissions and identify the intended receiver.

Floor request – request permission to engage in a PTT session.

Floor release – The results of the request (permission rights).

Floor grant – user granted the floor.

Floor idle indication – floor is idle.

Floor deny - floor request denied.

Floor taken – floor has been granted to the indicated user (occur to involved participants).

Floor revoke – after the session the floor is withdrawn.

Internet Protocol Multimedia Subsystem (IMS)

IMS is a platform for Internet Protocol multimedia services. This platform contains a SIP application server on which mobile data services like PoC are based. The servers handle session and group control, VoIP streaming, stream control, provisioning and management of users and groups [3].

PoC client

This is embedded in the mobile phone. The phone must support IP-based media like WI-FI. The phone must support SIP protocols as well as VoIP features.

Contact list management - allows the user to add and delete contacts. This contains the contact list which shows every contact that the user is acquainted to.

Presence information - Shows online people in your contact list.

EXISTING SOLUTIONS

Push to talk over a cell phone is available in many countries around the world. This service is being offered by mobile service providers. Most of the current PoC solutions are practically similar. The only difference is that they are being offered by different vendors. This uses the architecture of 3GPP (3rd Generation Partnership Project) IMS. Current PoC solutions are available on native OS, Symbian OS, PocketPC operating systems and supports GPRS, EDGE and UMTS mediums. They support 2.5 and 3 generations of cellular network. Table 1 shows some of the PoC service providers.

Service Providers	**Country**
Nextel Communications	USA
Sprint	USA
Motorola	USA
T-Mobile	Germany
Orange	UK

Vodafone	UK
Celtius	Finland
Saunalahti	Finland
M I Mobile	Singapore
India	Hutch

Table 2: Existing PTT solution.

LINKING THE SOLUTION TO THE PROBLEM

All the solution present exactly what the users want. In South Africa none of the three major service providers are offering it. The solutions offers voice instant messaging which is the basic functionality in our project. All the solutions are implemented on half-duplex communication, so bandwidth is efficiently used. The only additional feature is the fact that we will implement our solution on Wi-Fi network.

Best solution

Although the implementation of this project will be slightly different from the existing solutions, the underlying architecture is similar. The best solution in consideration is Celtius PoC. This solution covers most of the user requirements. Since the project has to implement a PoC server, Celtius present common solution.

Testing

The solution will first be tested on an emulator and then move to the actual phone.

Chapter 4

USER INTERFACE SPECIFICATION

INTRODUCTION

In the previous chapter; the analysis of the user requirements mentioned chapter 2 were discussed. The user requirements which are based on the user's perspective are were analyzed from the designer's perspective. Possible solutions together with possible methods of implementation were discussed. This chapter introduces the User Interface Specification (UIS). The description of the UIS includes the snap shots of the interfaces (screen) that the user interacts with. This chapter describes exactly what the user interface is going to do, what it looks like, and how the user interacts with the program.

DESCRIPTION OF THE COMPLETE USER INTERFACE

The UIS consists of one main graphical user interface (GUI), which consists with different operations enlisted in the options. The UIS shows the snap shot of the Symbian s60 emulator. The main window shows two routes options and exit. The options on the menu are:

Play – play the audio file

Record – used to record the audio

Stop – stops every operation (e.g. record, play)

Send - sends the audio file to another user via File Transfer protocol (FTP)

Save – saves the audio file in the emulator's memory

Loads - loads the audio file from the emulator

Exit - closes the application

WHAT THE USER INTERFACE LOOKS LIKE TO THE USER

Figure 2 shows the complete UIS. The main screen (background) shows the progress of each operation as well as the feedback after operation. This enables the user to see the status of the operation carried out. The left part of Figure 2 shows the whole emulator while the other one on the right only shows the screen.

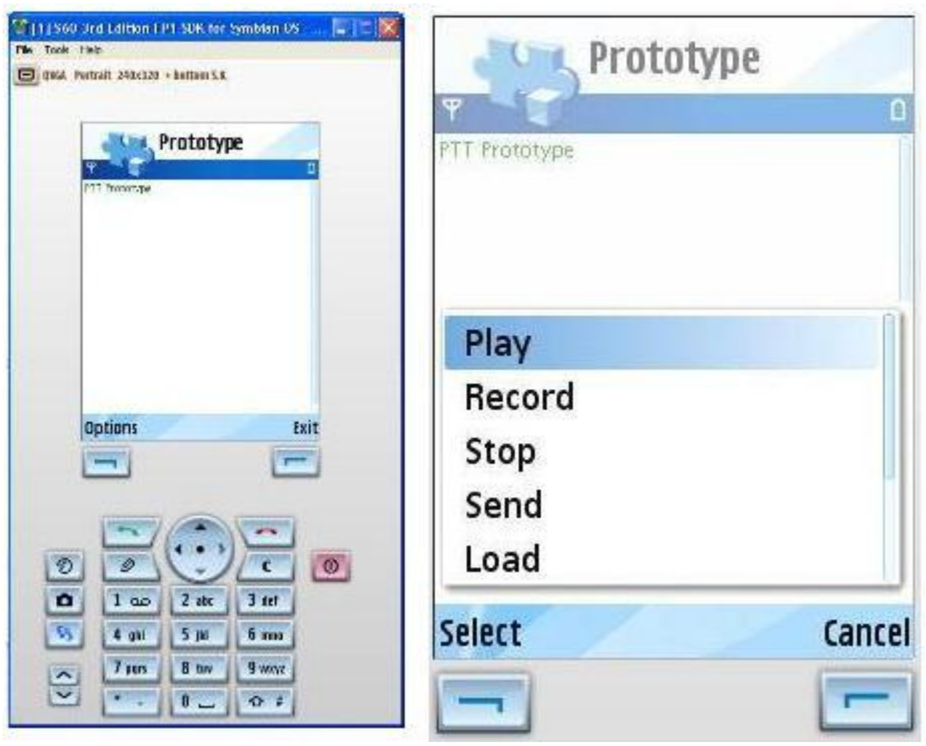
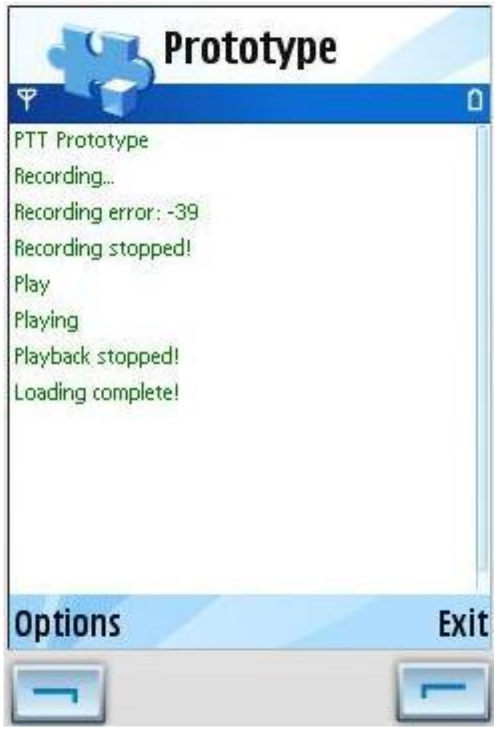


Figure 2: The main interface of the solution

HOW THE USER INTERFACE BEHAVES

The user interface enables the user to carry out operations that are listed. For each operation, a progress /feedback notification is displayed on the background screen. The system can be ported on a Symbian supported cell phone. The UI enables the user to record, play, send, load, stop and save the audio file. The send option first connects to the FTP server and uploads the audio file on the server. On the other hand, the Load option checks the server periodically for new messages (the audio file) and downloads it. The following figure 3 shows the feedback notifications n the main screen of the UI.



```
PTT Prototype
Recording...
Recording error: -39
Recording stopped!
Play
Playing
Playback stopped!
Loading complete!
```

Figure 3: Feedback of every operation of the option soft key

HOW THE USER INTERACTS WITH THE SYSTEM

The user must first record an audio through the Record option to be able to use the other functions. Microphone can be used to record the audio. Figure 4 shows the record function while Figure 5 shows the Play function.

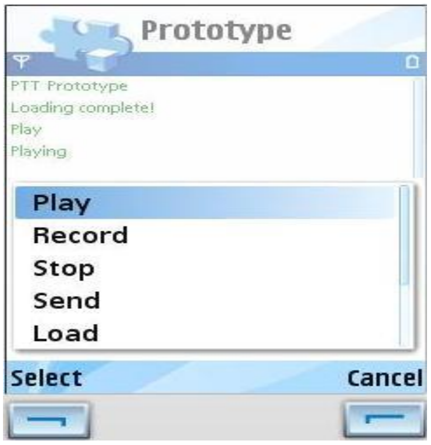

PTT Prototype
Loading complete!
Play
Playing

Figure 4: The play function

After recording the audio can be sent to another user. The other operations are applicable only the emulator.

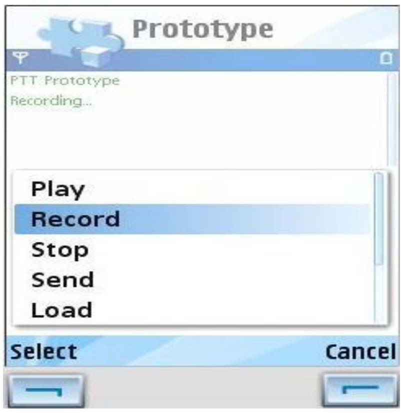

PTT Prototype
Recording...

Figure 5: The record function

SUMMARY

In this chapter User Interface were analyzed and observed. The various screens involved on the cell phone where analyzed. The various options or a menu that enables the user to interact with the application were analyzed. In the next chapter, the High Level Design is analyzed. The object oriented analysis of the solution is discussed.

Chapter 5

HIGH LEVEL DESIGN (OBJECT ORIENTED ANALYSIS)

INTRODUCTION

In the previous User Interface Specification were discussed, together with the various functions or options applicable. This chapter offers the analysis of the high level design. This is the object oriented view of the problem. This chapter describes various objects that are part of the solution at hand. The detailed description of each object is analyzed and relationship between the object is established.

DATA DICTIONARY OF EACH OBJECT REPRESENTATION

Object	Description
PoC client	This is the cell phone application. The PoC client uses SIP to transmit the sessions request and floor request to the PoC server. The client consist of user interface, call control module, floor control module, SIP module, contacts management, RTP module and alert management. User interface - consist of login page, registration page and the main PoC interface. Call control module – initiates calls through SIP VoIP Floor control module uses the SIP to exchange the floor control signals with PoC server. SIP module – enables other modules to access SIP communication services. Alert – offers the multimedia services to the PoC client.

	Contacts management - used to retrieve the user's information from group and list management server.
PoC server	The PoC server is an interface of communication between two PoC clients. It handles the sessions (through SIP) and relays/ directs them to their destination. It handles the floor control which decides who to grant the floor (permits to talk) at a certain time. The PoC server works handy with the SIP server. It also stores the information of all the registered clients.
SIP server	This server consists of Session Initiation Protocol. SIP is used for initiation, termination of PoC sessions. It is used for call control.
Group and List management server	Stores the details of the groups and the list. It also stores the presence information as well as the status information. The presence uses the SIP sub protocol called SIMPLE.
RTP module	Real time protocol that enables real time communication between the PoC Client and the PoC server.

Table 3 : This table describes the various object of the solution.

USER INTERACTION DESIGN (USE CASE DIAGRAM)

This is the high level design of the from the users perspective. It shows the functional activities that the user can perform. Figure 6 shows the Use Case Diagram.

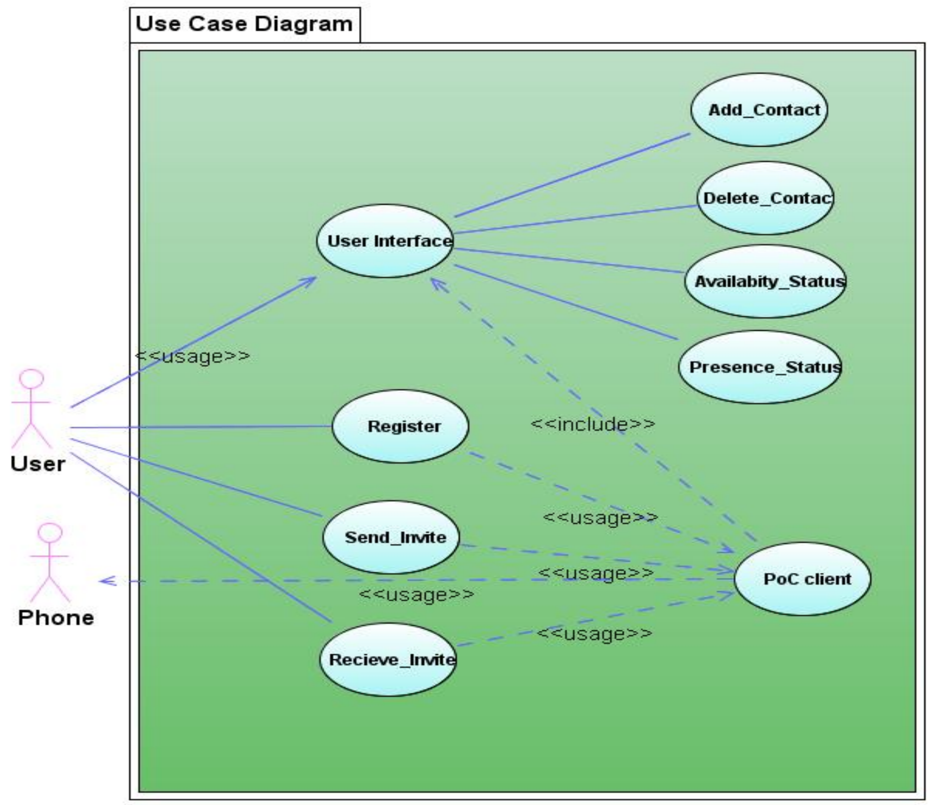


Figure 6: Use case diagram of the solution.

The use case diagram has two actors, the user and the mobile phone. The user uses the mobile phone where the PoC application is installed. The user interacts with the system through the interface. The basic high level functions are – registration of new user, sending invites, receiving invites. From the main interface the user can add new contacts, delete contacts, see who is online (presence) amongst your buddies and see the availability information.

CLASS DIAGRAMS SHOWING THE NAME, ATTRIBUTES, AND METHODS OF EACH CLASS

Figure 7 shows the floor control class. Its main responsibility is to control the allocation of the floor. To obtain the floor the client sends the request to the

PoC server and waits for the feedback. The status of the floor is broadcasted to all the parties by the PoC server. Figure 8 shows the detailed description of the class diagrams.

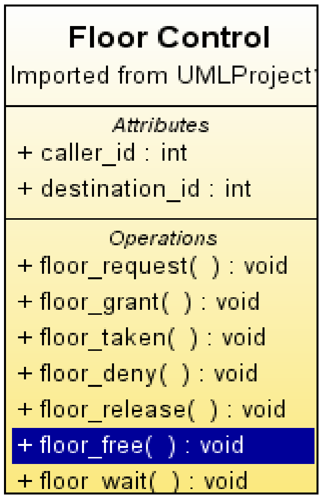


Figure 7: Floor control class.

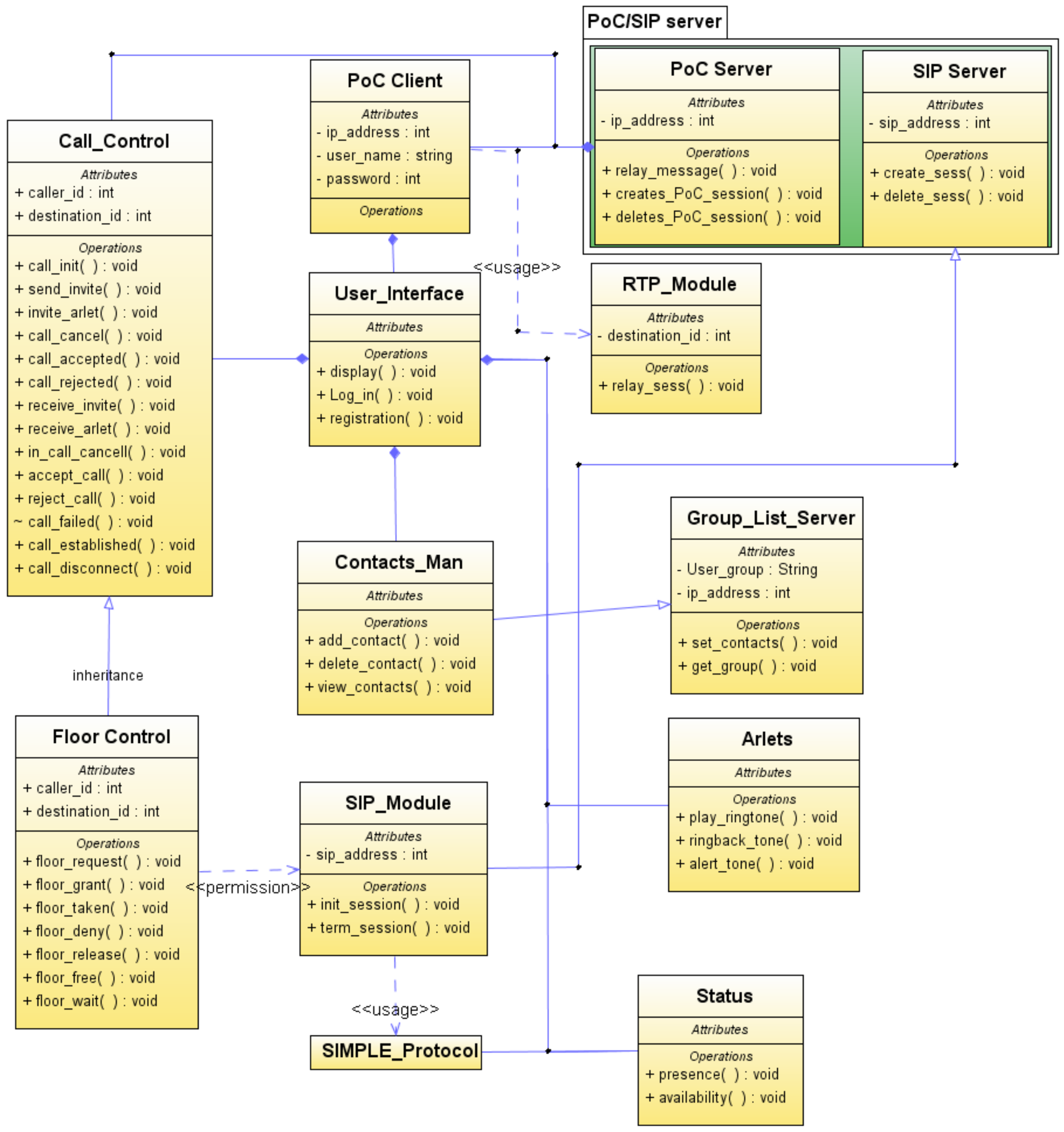


Figure 8: Class diagram depicting various classes of the problem solution.

THE RELATIONSHIP BETWEEN OBJECTS

Figure 9 shows the relationship between some of the object. The M represents "Many" while the 1 represent "one" i.e. 1: M is one-to-many relationship. The

PoC client and the PoC server has M:1 relationship, which means that you can have many PoC clients and only one server. The relationship between the servers is 1:1.

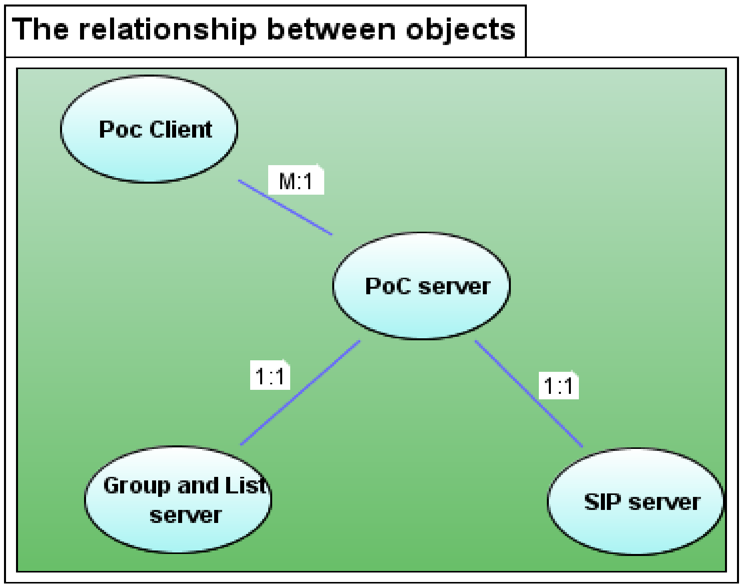


Figure 9: The relationship between classes.

CLASS DIAGRAMS AND DATA DICTIONARY FOR THE INTERFACE DOMAIN

Table 4 and figure 10 shows the data dictionary and class diagrams of the interface domain respectively.

Object	Description
User interface	GUI application of the PoC client
Register	Allow first users to register for the PoC services. The information is stored in the PoC server.

Login	Users log-in in order to use the application. The post log-in invokes the presence status.
Main window	Displays the necessary information on the screen i.e. presence and availability status

Table 4 : Data dictionary of the interface of the application domain.

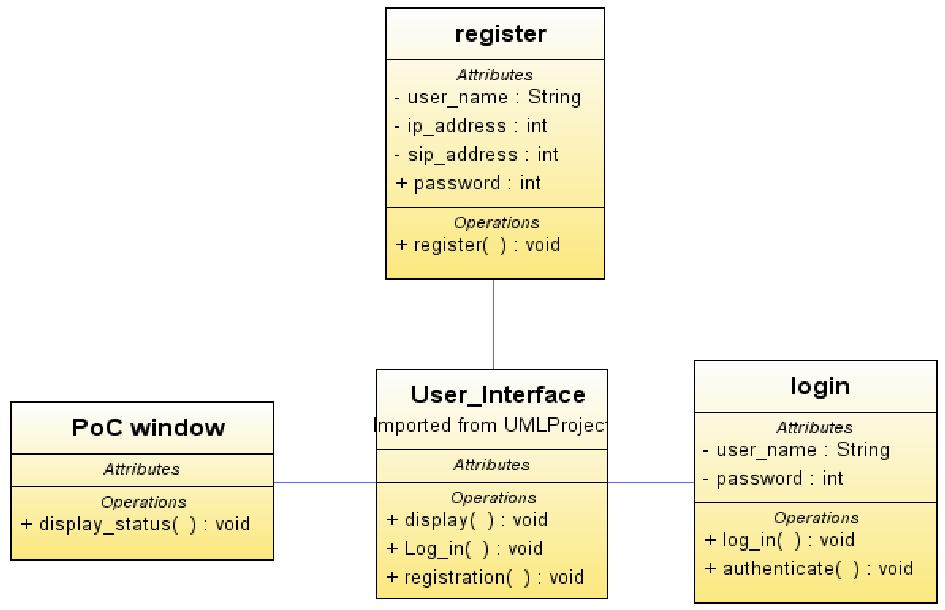


Figure 10: Class diagrams of the interface domain.

Figure 10: Class diagrams of the interface domain.

CLASS DIAGRAMS AND DATA DICTIONARY FOR THE APPLICATION DOMAIN

Figure 11 shows the class diagram for the application domain of the problem. For the data dictionary of this domain refer to table 3.

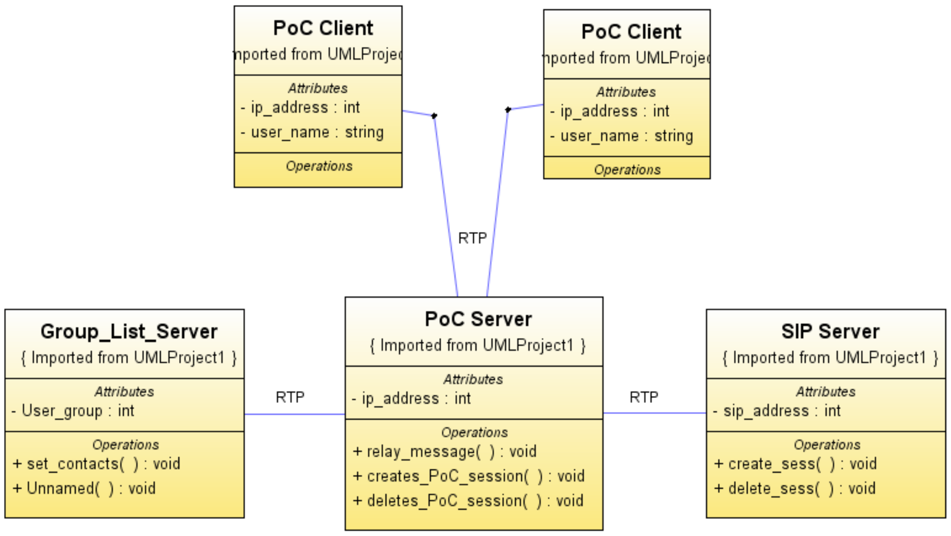


Figure 11: Class diagrams of the solution's application domain.

SUMMARY

In this chapter an analysis of the High Level Design was obtained. The various classes involved and how they interact with each other were described as well as the relationship between them. The problem solution was analyzed from an object oriented view. In the next chapter, the classes discussed in this chapter are analyzed further; pseudo codes of the solution are extracted from the classes.

Chapter 6

LOW LEVEL DESIGN (OBJECT ORIENTED DESIGN)

INTRODUCTION

In the previous chapter, object oriented analysis otherwise known as the high level design of the solution was described in details. The possible classes of the solution were modeled through class diagrams and the relationship amongst them. This chapter takes the classes further by creating pseudo codes.

INNER DETAILS OF CLASS ATTRIBUTE (DATA TYPES)

The following table 5 shows the inner details of the classes with their attributes. It is the description of the attributes with regards to the type of data that they represent.

Class	Attributes
PoC client	**int ip_address** – stores the IP address of the client.
	String user_name – stores the user name of the person using the PoC client
PoC server	**int ip_address** – stores the IP address of the server.
SIP server	**Int sip_address** – stores the address of the SIP server.
Group and List management server	**String user_group** - stores the names of the groups currently registered.
	Int ip_address - the IP address of the server
RTP module	**int destination _id** – The IP address of the intended receiver.

Call control/floor control	**int caller_id** – The IP-address of the sender
	int destination _id – The IP address of the intended receiver.

Table 5 : A description of methods of each class.

INNER DETAILS OF CLASS METHODS (FUNCTIONS)

Class	functions
PoC client	**public poc_client()** – initializes all attributes.
PoC server	**public poc_server()** – initializes all attributes. **public void relay_message ()** – this method relays the audio message to the intended receiver.
SIP server	**public sip_server()** – initializes all the necessary attributes. **public void create_session()** – this initiates the PoC sessions. **public void delete_session()** – terminates the PoC sessons.
Group and List management server	**public glms()** – initializes all attributes.
	public String set_group() – create a PoC buddy group
User interface	**public user_inteface()** – attribute initialization.
	public void display() – this method displays all the necessary information on the screen i.e. presence, availability
	public void log_in() – this method allows the user to

	log in, only registered users can log in.
	public void register() – enables the first time users to register.
Contact management	**public contact_man()** – initialization of attributes.
	public void add_contact() – this method allows the user to add new contacts on the buddy list.
	public void delete_contact() - this method allows the user to delete contacts from the buddy list.
	public void view_contact() - this method enables the user to view all contacts in the buddy list.
Alerts/multimedia services	**public alert_notification()** – initializes all attributes.
	public void ringtone() – this method plays a ringing tone to alert the user (reciever) of incoming call.
	public void ringback() – this method plays a tone that alert the sender if the call is going through
Call control	**public call_control()** - initializes all attributes.
	public void init() – initializes a PoC session.
	public void send_invite() – this method enables the user to sent a invite to another PoC user.

	public void invite_alert() – returns notification feedback on an send invite.
	public void call_cancelled() – cancels a call that is in process (by the sender).
	public void call_accepted() –alerts the sender if the call is accepted call.
	public void call_rejected() – alerts the sender of a if a call is rejected.
	public void reicieve_invite() – alert the user of the incoming invite.
	public void receive_alert() – plays a ringing tone to notify a user of an
	public void in_call_cancelled() – the call has been cancelled by the receiver.
	public void accept_call() – the call has been accepted by the receiver.
	public void reject_call() – call reject
	public void call_failed() – returns a notification if a call failed to establish.
	public void call_established() – the call established successful, talk can take place.
	public void disconnect() – connection lost.
Floor control	**public floor_control() –** initializes all attributes.
	public void floor_request() – this method request the floor in order to make talk session.
	public void floor_grant() – this method grants the

	floor to the requested user.
	public void floor_taken() – alerts the user that the floor is in use by someone.
	public void floor_deny() – returns an alert that the floor request has been turned away (denied).
	Public void floor_release () – this method releases the floor after use
	public void floor_free() – alerts the user that the floor is free.
	public void floor_wait() – waiting for the floor to be free.

Table 6 : An inner details of class methods (functions).

STATE DIAGRAM OF THE FLOOR CONTROL

Table 6 and figure 11 shows the data dictionary and the state diagram of a floor request control.

State	**Description**
Initial state/ activation	A PoC call is activated; After the activation a floor signal is send from the PoC server.
Request_pending	A floor request has been sent to the PoC server. User is waiting for the response.
Floor_free	The floor is free and ready to be used.

Owner/Receiver	The user is granted the floor
Reserved/ in_use	The floor is been used by another user. Not available
Release_Pending	floor request release has been sent to the PoC server. User waiting for the response.

Table 7 : Data dictionary of the control.

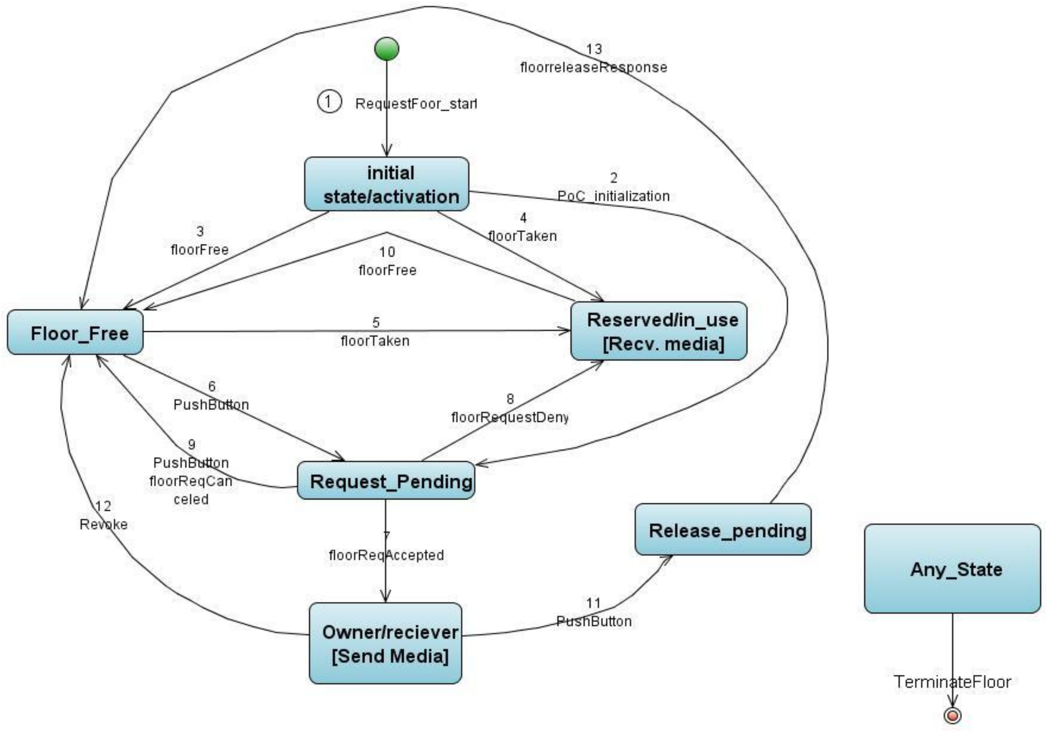


Figure 12: State diagram of the floor control.

PSEUDO-CODE

```
PoC server
Class poc_server{
        String user_name;
        int ip_address;
        String password;
      method poc_server() {
            initialize user_name, password = NULL;
            ip_address = 0;
            while (true)
                    get destination_address;
                    relay_message(destination_address,message)
           method relay_message(int d_address, msg){

           }
         method void setUserName(String us){
               username = us;
          }

         method void setIPaddress(int ip){
         ip_address = ip;
         }
         method void setIPassword(int pas){
         password = pas;
         }
         method String getUserName(String us){
               return username;
          }

         method int getIPaddress(int ip){
               return ip_address;
         }

          method int getIPassword(int pas){
              return password;
         }

      }
}
PoC client
class poc_client extents poc_server (){
   super.poc_class();
   method poc_server(){
         initialize user_name, password = NULL;
         ip_address = 0;
         while (true)
               sip_server();
               call_control();
                floor_control();
  }
```

```
}
SIP server
class sip_server extends poc_server(){
    super.poc_server();
    method sip_server(){
    while (true)
              create_session();
     }
  method void create_session(){
        invoke SIP session initiation;
}
  method void terminate_session(){
        invoke SIP session termination;
    }
}
Call_control
class call_control{
  int caller_id;
  int destination_id;

  method call_control{
    while(true){
    switch(call){
    case 1 : initialize call
      call init();
    case 2 : send invite
      call send_invite();
    case 3 : invite alert
      call invite_alert(); //play the alerrt tone
    case 4 : call accepeted
      call call_accepted();
    case 5 : call rejected
      call call_reject();
    case 6 : receiving invites
      call ireceive_invite();
    case 7 : receive invites alert
      call receive_alert();
    case 8 : cancel incoming
      call in_call_cancel();
    case 9 : accepting call invites
      call accept_call();
    case 10 : rejecting call invites
      call reject_call();
    case 11 : failed call
      call call_failed();
    case 12 : establishement
      call call_established();
    case 13 : diconnection
      call call_disconnect();
    }

      method void call_init() {
        call sip;
      }
      method void send_invite(){
        call sip module to isssue a SIP invite;
      }
      method void invite_alert(){
        if call established
          play rinngback tone;
      }
      method void call_cancelled(){
        if cancel button is pressed
          call not established;
      }
      method void call_accepted(){
        if call is established and answered
          beging to transmit/ to talk;
      }
      method void call_rejected(){
        if user is not available for PTT sessions e.g. busy
          send reject alert to the caller;
      }
      method void receive_invite(){
        if you are available and some is callling you
          call recieve_alert; //play ringtone
      }
      method void receive_alert(){
        if there is an invite
          play ringtone;

      }
      method void in_call_cancelled(){
        if an invite is recieved and cancell is pressed
```

```
        call reject_call;
      }
      method void accept_call(){
        if ringing
          connection is made;
      }
      method void reject_call(){
        if busy
          call in_call_cancell;

      }
      method void call_failed(){
        if call is not established
          return call failed;
      }
      method void call_established(){
        if there is a sent alert
          waiting for answer
      }
      method void disconnect(){
        if there is disconnection
          return there is a disconnection

      }
    }
  }
}
```

Floor control

```
class floor_control extends call_control{
  int caller_id;
  int destination_id;

  method call_control{
    while(true){
    switch(call){
    case 1 : requesting a floor
      call floor_request();
    case 2 : the user granted the floor
      call floor_grant();
    case 3 : floor is in use
      call floor_taken();
    case 4 : floor request rejected
      call floor_deny();
    case 5 : floor is released, now available
      call floor_release();
    case 6 : floor is not in use
      call floor_free();
    case 7 : user waits for the floor which is inuse
      call floor_wait();

    }
    }
  }
    method void floor_request() {
      sends floor request to the PoC server through SIP
    }
    method void floor_grant() {
      invoke RTP module;
      user interface -> show user busy;
      multimedia->record
    }
    method void floor_taken() {
      call floordeny;
    }
    method void floor_deny() {
      PoC sends resevered message to the client;

    }
    method void floor_release(){
      PoC client sends floorRelease message to the PoC server via SIP
    }
    method void floor_free() {
      PoC server sends a floorRequest response to the PoC releasing client

    }
    method void floor_wait() {
      call_control(); // instructs the UI module show the status of the floor

    }
}
```

Class SIP Module

```
class sip_module extends sip_server{
  super.sip_server()
  method sip_module(){
    sip_address = 0;

  }

  method void init_session(){
    invokes SIP to initialize the connection;
  }
  method void term_session(){
    invokes the SIP to terminates a connecton;
  }

}
Status
class status extends SIMPLE{

  method status(){

  }

  method void presence(){
    invokes the SIMPLE
  }

  method void availability(){
    dislpays the availability status;
  }

}
RTP module
class rtp_module extends RTP(){

  method rtp_module(){}

  method void ringtone(){
    play ringtone;
  }

  method void ringback(){
    play ringback; }
  }
```

C h a p t e r   5

IMPLEMENTATON

INTRODUCTION

The previous chapter gave an overview of how the actual implementation of the project looks like. Various classes together with their pseudo codes were discussed. The previous chapter holds the introductory keys to this chapter, which are now detailed. In this chapter two methods of implementation are discussed which are the cell phone implementation of Push-to-Talk (PoC) and PC Push-to-Talk.

SOFTWARE'S DEPLOYED

SIP server

Both implementations use the same SIP servers throughout. SIP server is the server that relays the Real Time Protocol (RTP) voice sessions from one end to the other. This project used two SIP severs for experimentation purpose in order to test the feasibility of the system as well as the degree of compliance between the server and either of the two approaches. We used Openser and Asterisk SIP servers.

Openser was the ideal one due to its support for various communication media. Apart from SIP, Jabber and other common mediums amongst the SIP servers, it also supports instant messaging. The version improvement from Openser to Kamailio also brought a lot of improvements. The advantage of using Openeser/Kamailio is the fact that registration is on the client's side. However OPenser/Kamailio runs in the background which makes it difficult to trace down bugs. On the other hand Asterisk operates much differently. The registrations are configured on the server. During RTP session relay, IP addresses of the participants are shown as well as the session process, so it very vital in terms of debugging. The disadvantage of using it however is the fact that during real time communication, the registration might take a little bit longer as it will have to make use of scripts. Since the project is still on the development stage we use Asterisk as our SIP server.

Cell phone Implementation

Symbian SDK – This is a software development kit for Symbian S60 applications. The SDK enables developers to build applications using Symbian C++, Open C/C++, Java development and Web runtime [4]. It includes a Symbian mobile phone emulator which is used for testing of applications before being imported to the real mobile phone device.

Carbide C++ – This is the software development tool for developing Symbian C++ applications. It is an integrated development environment (IDE) which is based on Eclipse IDE. It enables developers to quickly and efficiently create, Code, Test and deploy software for the Symbian operating system [5].

Nokia PC suite – It is an application programming interface (API) that create an interface of connectivity between a Symbian phone and a PC.

PJSIP Symbian – Is an open source SIP stack protocol. It is very small as compared to other SIP stacks but yet flexible with a high performance rate [6]. PJSIP is the only SIP stack based on Symbian platform. We implement Push-to-Talk on the mobile phone using Symbian PJSIP as the basis for our development. The reason being, PJSIP offers a lot of functionalities such as presence, RTP sessions etc.

PC Push-to-Talk

Netbeans – This is a multi-lingual platform IDE for development.

SIP-communicator – This is an open source SIP stack developed in java. It was used as a basis for the PC implementation of the project. It was basically used because of its simplicity and object oriented style. This allows us to implement our Push-to-talk using some of their explicitly documented classes and methods.

HARDWARE'S DEPLOYED

Microphone – used for record the voice for the PC Push-to-Talk.

Speakers – used for sound output, also for PC Push-to-Talk

Mobile phone device (Nokia) – required for the cell phone application

CHALLENGES

Every project has to go through rough patches and ours was no exceptions. There were a lot of technical obstacles encounter throughout the development of the project. We had technical challenges through some configurations of SIP server to gauge most suitable one for our project. However those were sort of challenges were overcome.

The most daunting challenge which left us with no choice but to divert a bit from the original architecture was the feasibility of the emulators. The project is real time and as such the emulators do not support real time audio streaming. Since the real mobile devices were not available at that time, the project was then moved to PC. The other reason being that it is not easy to debug on the mobile phone, if the application cannot be tested on the emulator before ported on the mobile phone always raises doubting views as far as efficiency and time is concerned. Nonetheless the project will be continued on the mobile phone after the completion of the PC application.

CODE DOCUMENTATION

The implementation of the project was done through modifying the SIP-communicator source code as well as creating new methods in order to meet the basic requirements. The source code given below show basic functionalities of the project which are half duplex communication, floor control and PTT button control. The rest of the code will be in a CD. We first change the audio streaming from full duplex to half duplex and create a method that will automatically put the streaming on hold. This is to allow the users to control the transmission through a PTT button. All this is done the SIP-Communicator's "OperationSetBasicTelephonySipImp.java" under the method "ProcessAck". The second method is for the PTT button control functionalities. This is done by modifying the SIP-Communicator's "HoldbuttonModel.java", new methods are implemented inside such as session termination. The code modified or created is clearly documented using /**/ or //.

```
/**
 *This method initiates the audio streaming during a session.
 *It first find the call to be streamed, before the streaming we put it on hold, so
 *to be controlled by the PTT button.
 *The streaming then depends on the PTT/hold button
 *Normally the Sip-communicator is full duplex, so this is where we reduce
 * the duplex to half.
 *The audio session is is first initiated and put on hold immediately
 *
 * Updates the session description and sends the state of the corresponding
 * call participant to CONNECTED.
 *
 *@param serverTransaction the transaction that the Ack was received in.
```

```
*@param ackRequest Request
*input – call – to establish the session
*output – audio streaming
*/
  private void processAck(ServerTransaction serverTransaction,
                          Request ackRequest)
  {
     // find the call
     CallParticipantSipImpl participant
        = activeCallsRepository.findCallParticipant(
              serverTransaction.getDialog());

     if (participant == null)
     {
        // this is most probably the ack for a killed call - don't signal it
        logger.debug("didn't find an ack's call, returning");
        return;
     }

     ContentLengthHeader contentLength = ackRequest.getContentLength();
     if ((contentLength != null) && (contentLength.getContentLength() > 0))
     {
        participant.setSdpDescription(
           new String(ackRequest.getRawContent()));
     }

     // change status
     CallParticipantState participantState = participant.getState();
     if (!CallParticipantState.isOnHold(participantState))
     {
        if (CallParticipantState.CONNECTED.equals(participantState))
        {
           try
           {
              ((CallSipImpl) participant.getCall())
                 .getMediaCallSession()
                    .startStreamingAndProcessingMedia();
           }
           catch (MediaException ex)
           {
              logger.error(
                 "Failed to start the streaming"
```

```
                         + " and the processing of the media",
                    ex);
                }
            }
            else
               participant.setState(CallParticipantState.CONNECTED);
        }

/*
 *This is the most important part of the project, this method put the audio
 * streaming which is defined above on hold.
 * The streaming is automatically put on hold immediately after initiation of
 * the session.
 *The status of the call is set to on hold locally.
 * The
 */
        if ((CallSipImpl) participant.getCall() != null) {
            //initiate a new instance of the call session
            OperationSetBasicTelephony telephony =
              (OperationSetBasicTelephony)  ((CallSipImpl)  participant.getCall()).
getProtocolProvider().getOperationSet(OperationSetBasicTelephony.class);
                  //get the involved participants
                  Iterator<CallParticipant>      participants      =      ((CallSipImpl)
participant.getCall()).getCallParticipants();
                    //checks if the call is on i.e the callee has responded
                    while (participants.hasNext()) {
                        //the involved participants
                        //initiate a new call instance between the defined participants
                       CallParticipant callParticipant = participants.next();

                       try {
                         //put the call onhold
                          telephony.putOnHold(callParticipant);
                       } catch (OperationFailedException ex) {
                       // TODO Auto-generated method stub
                       }
                    }
                    //set the status of the call to onhold locally

participant.setState(CallParticipantState.ON_HOLD_LOCALLY);
                  }
                 ///////////////
```

```
    }


/**
*This is where we define the on hold button which in this case serves as our
*PTT button.
* The button controls the streaming that was put onhold.
*Procedure - initially the call is onhold as defined previously
*               - when the button is pressed, the connection is initiated and is
*                half duplex
*               -When the button is pressed for the second time, the half duplex
*                connection closes.
*               -  and hence the connection is terminated.
*Since this is a button, all the execution methods are implemented inside the
*actionPeformed method,
*this method executes the action depending on the action perfomed on the
*mouse. This is done through
*java's MouseListener methods and executed by java's actionlistener.
* Initializes a new <tt>HoldButtonModel</tt> instance to represent
* the state of a specific <tt>CallParticipant</tt> as a toggle
* button.
*
* @param call- this is the input
*          the <tt>Call</tt> whose state is to be
*          represented as a toggle button.
*@output
*/
      public HoldButtonModel(Call call)
      {
        //initiate new button
          this.call = call;
         //passes whatever the mouse does to the actionlistener where the
execution takes place
         addActionListener(this);
      }

      public void actionPerformed(ActionEvent evt)
      {

         if (call != null)
         {
```

```
      //initiate a new instance of the call session
       OperationSetBasicTelephony telephony =
          (OperationSetBasicTelephony) call.getProtocolProvider()
              .getOperationSet(OperationSetBasicTelephony.class);
       //get the involved participants
       Iterator<CallParticipant> participants = call.getCallParticipants();
       //checks if the call is on
       while (participants.hasNext())
       {
         //initiate a new call instance between the defined participants
          CallParticipant callParticipant = participants.next();

          try
          {
              //check i the hold/PTT button is selected
              if (isSelected())
                    //disconnect the call and terminate the session
                    telephony.hangupCallParticipant(callParticipant);
              else
          //put the call off hold to allow the sstreaming and hence the half
duplex connection.
                    telephony.putOffHold(callParticipant);

          }
          catch (OperationFailedException ex)
          {
              // TODO Auto-generated method stub
          }
       }
    }
  }
}
```

SUMMARY

In this chapter two approaches of implementing the project were discussed in detailed together with their source code. The project was first implemented on the mobile phone emulator using Pjsip as a SIP stack; however the emulators do not support RTP audio streaming. This resulted in us implementing it on a PC using SIP-Communicator as a SIP stack. The mobile development will nonetheless be continued using real mobile phone devices. Asterisk SIP server was used which relays the voice streams between two parties. The code documentation given shows two basic functionalities, which are half-duplex communication, PTT button functionalities with the rest of the code to be in a CD.

Chapter 8

USER GUIDE

INTRODUCTION

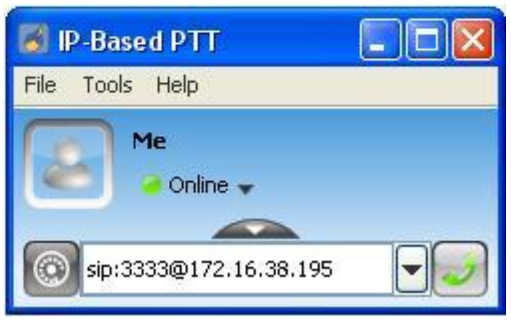
In the previous chapter, we discussed the implementation of the project. We discussed the two approaches of implementing the project which are the PC PTT as well as the mobile PTT and looked at some of the code. We further discussed the challenges encountered as well as the tools used. This chapter discusses the user's guide of the project.

PC

The interface has three sub-menus File, Tools and Help. File helps you to add contacts and create PTT groups while Tools gives you technical options and Help menu gives inside as how to go about the application.

Getting started

The green light denotes that you are online and ready to make a PTT session. If the green light is red, then the SIP server is not running. On the interface's address box, enter the sip address of the callee and press the dial button.



sip:3333@172.16.38.195

Figure 13: PTT before the session.

The following window will appear which allows you to see the state of the session. The remote user will have to answer the call session before you can engage in a PTT session.

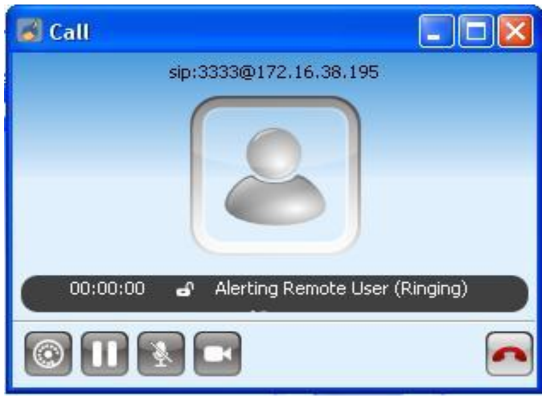


Figure 14: The state of the session call is ringing.

After the confirmation, the connection would have been made with no streaming. The following screen shot shows the state of the session. The caller must now control the session with the PTT button. By clicking the button, the half-duplex voice streaming is established.

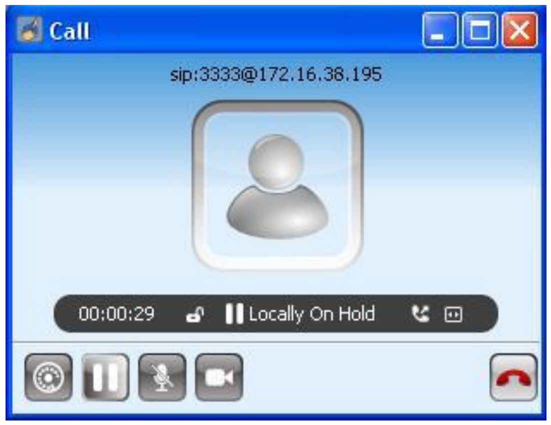


Figure 15: Awaiting for the PTT control.

The following window shot shows the status when the PTT session is on.

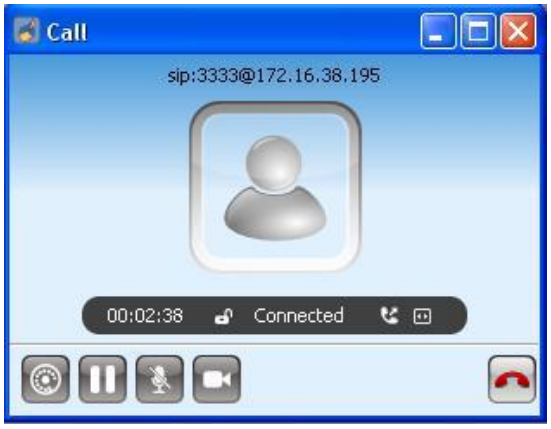


Figure 16: session on.

The session is terminated by clicking the same button again. Ideally it is press->hold talk release->close, but the concept has been changed due to user's requirements. The following window shot shows the end of the session.

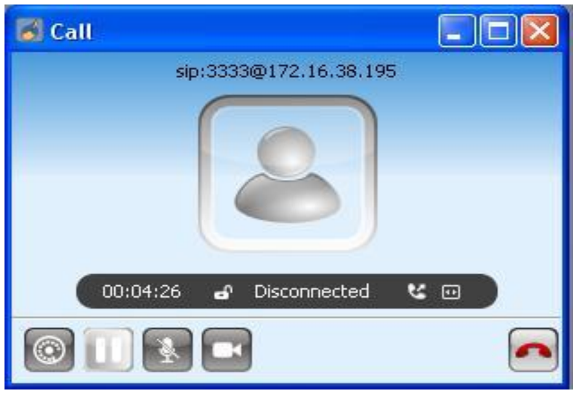


Figure 17: PTT session terminated.

MOBILE PTT

The mobile version of the project was explored using Nokia E 51 and E 71.

Getting started

On the phone's menu go to installations and choose symbian_ua. The command line like interface appears. All of the instructions are displayed on the interface. The following snap shots show the display on the phone.

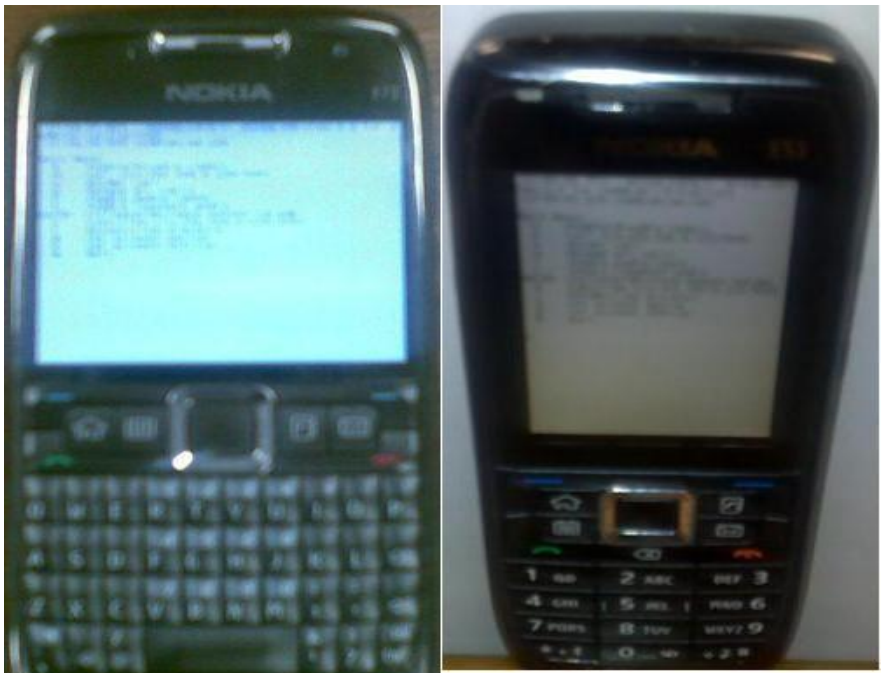


Figure 18: On the phone E 71 left and E 51 right.

C h a p t e r 9

TESTING

INTRODUCTION

In the previous chapter, we discussed the user’s guide of the application. In this chapter we look at various testing strategies and give detailed discussion about the results. The testing was conducted through users.

TEST PROCEDURE

The application was tested on two platforms. On the local area network (LAN) using two PC’s and on the UWC CoE Wi-Fi network using Nokia E51 and E71. The users were given three task scenarios to choose from. The scenarios were social, emergency and corporate (secretariat) situations. The users were then given questionnaire to complete. We had six users testing our application with two users per scenario. Refer to appendix F for questionnaire and the task scenarios.

VALIDATION TESTING

According to Roger S. Pressman, Validation tests if the functionality of the software functions in a manner that can be reasonably expected by the customer [7]. A series of tests were conducted to ensure and demonstrate conformity with user’s requirement. After six test series, the graph below shows the response of the users to following questions.

Question: *Did you manage to complete your task?*

Question: *Are you satisfied with the efficiency of the application?*

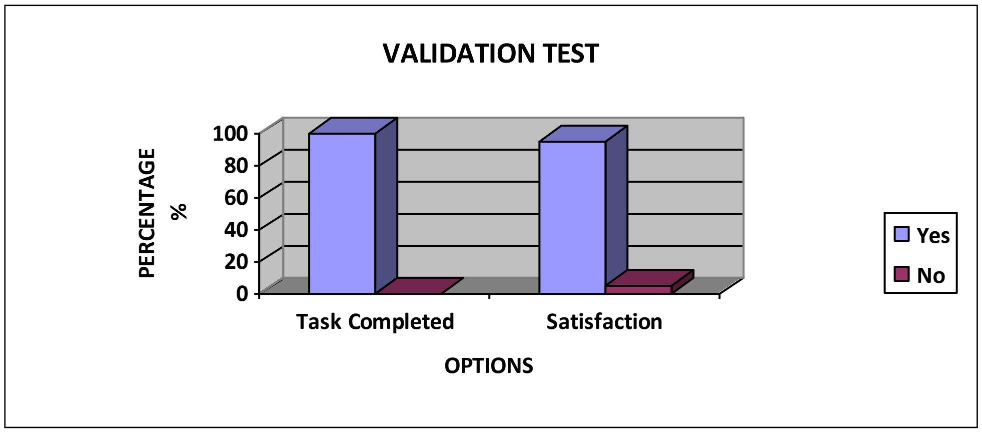


Figure 19: Validation testing results.

According to the results, the validation test was successful as 100% of the users completed the task and about 95% satisfied with the results. However the reason given for the unsatisfactory user was mainly about the voice stream cuts, which was not the case to all users as the testing were conducted at different times. This will be discussed thoroughly in the stress testing. The following figure 20 shows the responses of the users in rating the interface of the application.

Question: *How would you rate the interface?*

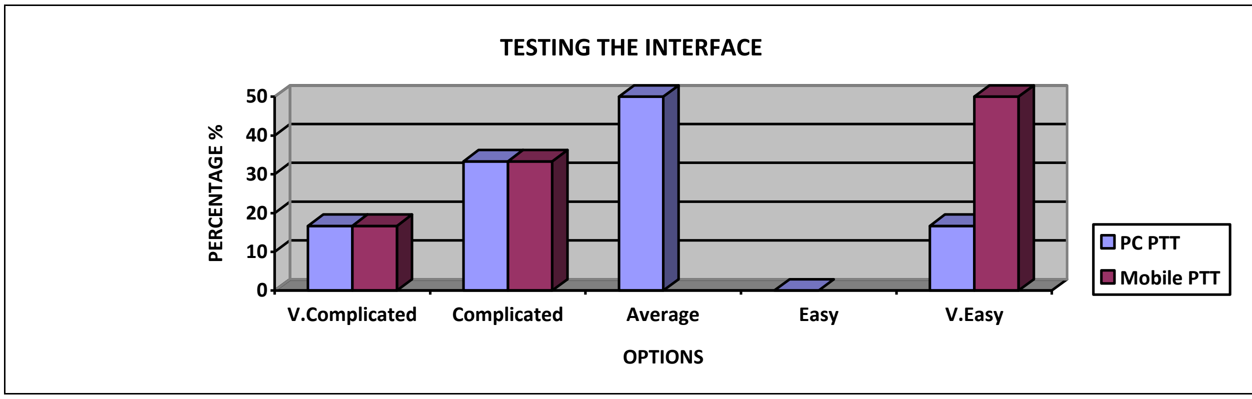


Figure 20: Testing the interface.

Most comments regarding the interface were not positive. The interface was not difficult to use according to the results. However it was just not attractive especially the phone's one. Most people would love to see graphical user interface. This was no surprise because from due to time we could not complete the GUI on time.

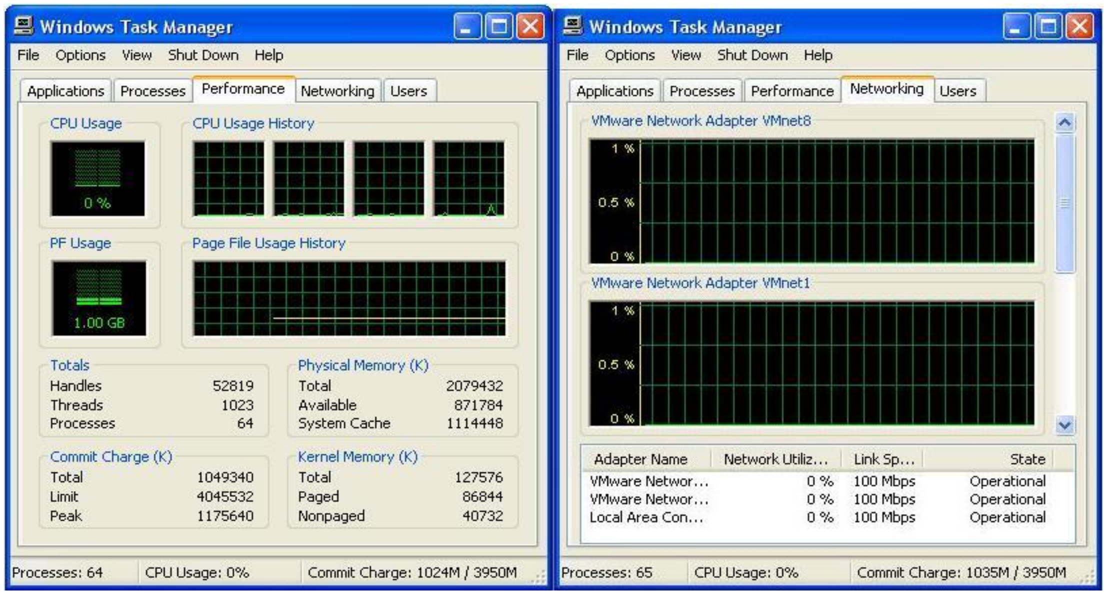
STRESS TESTING

Stress testing was conducted to test the system under immense pressure from different factors. The PC application was tested against the different traffic variations on the network. Thus we tested at mid-night were the traffic was low and also mid-day were the network traffic is very high. Under lot of traffic on the network, the communication tends to suffer against latency (delay of voice stream packets on the network). This was observantly confirmed by the breakdown of the voice streams and the delay in time of the voice to arrive. After careful evaluation and literature survey study, we deduce that the client server concept which was used also had a negative effect towards the application under abnormal circumstances.

On the mobile phone many networks can bear negative effect on the application precisely due to interference. The interference cause lot of unnecessary background noise. Nonetheless the peer-to-peer approach overcomes the latency problems encountered in the PC application because even in the presence of the noise, the voice still arrives on time and clear. This noise can however be eliminated by switching off the loud speaker.

PERFORMANCE TESTING

Software should conform to performance otherwise it will be unacceptable even if it is validated. Performance testing, tests the run-time performance of software within the context of an integrated system. The performance of the application was tested observantly through users and systematically using windows task manager. The following screen shots were taken before the commencement of the application whereby no process was running on the machine.


Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
CPU Usage
CPU Usage History
0 %
PF Usage
Page File Usage History
1.00 GB
Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
VMware Network Adapter VMnet8
1 %
0.5 %
0 %
VMware Network Adapter VMnet1
1 %
0.5 %
0 %

Totals	
Handles	52819
Threads	1023
Processes	64

Physical Memory (K)	
Total	2079432
Available	871784
System Cache	1114448

Commit Charge (K)	
Total	1049340
Limit	4045532
Peak	1175640

Kernel Memory (K)	
Total	127576
Paged	86844
Nonpaged	40732

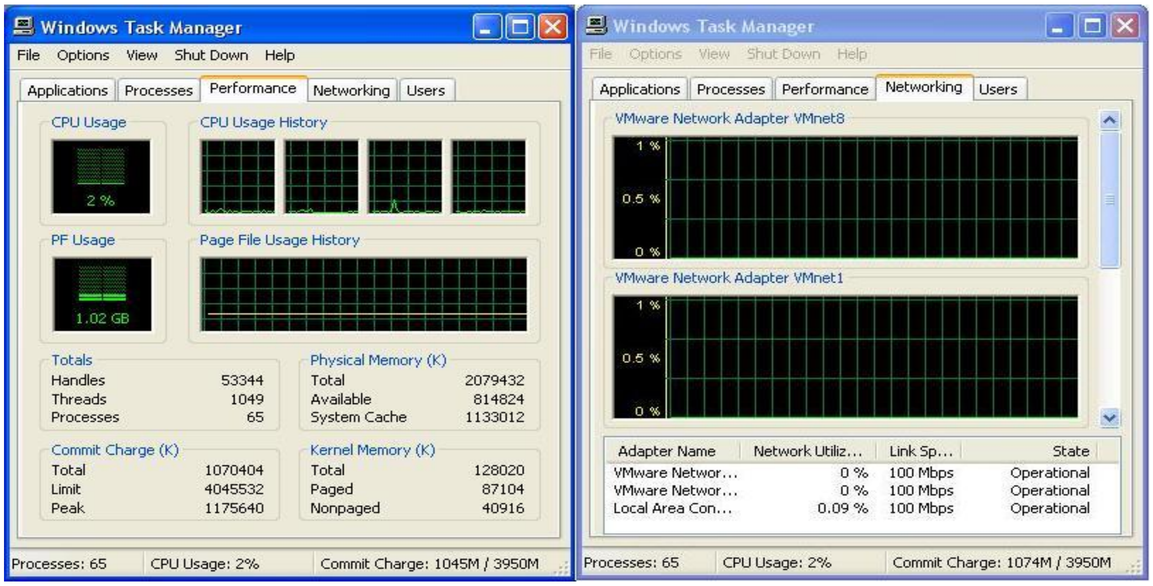
Processes: 64 CPU Usage: 0% Commit Charge: 1024M / 3950M

Adapter Name	Network Utiliz...	Link Sp...	State
VMware Networ...	0 %	100 Mbps	Operational
VMware Networ...	0 %	100 Mbps	Operational
Local Area Con...	0 %	100 Mbps	Operational

Processes: 65 CPU Usage: 0% Commit Charge: 1035M / 3950M

Figure 21: Windows Task manager before PTT session.

As seen from the figure (on the left), the CPU usage before use was 0% since there was no processes running. The network usage on the right hand side of the figure shows 0% of bandwidth usage. The following figure 21 shows the Task manager during the PTT session.


Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
CPU Usage
2 %
CPU Usage History
PF Usage
1.02 GB
Page File Usage History
Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
VMware Network Adapter VMnet8
1 %
0.5 %
0 %
VMware Network Adapter VMnet1
1 %
0.5 %
0 %

Totals	
Handles	53344
Threads	1049
Processes	65

Physical Memory (K)	
Total	2079432
Available	814824
System Cache	1133012

Commit Charge (K)	
Total	1070404
Limit	4045532
Peak	1175640

Kernel Memory (K)	
Total	128020
Paged	87104
Nonpaged	40916

Processes: 65 CPU Usage: 2% Commit Charge: 1045M / 3950M

Adapter Name	Network Utiliz...	Link Sp...	State
VMware Networ...	0 %	100 Mbps	Operational
VMware Networ...	0 %	100 Mbps	Operational
Local Area Con...	0.09 %	100 Mbps	Operational

Processes: 65 CPU Usage: 2% Commit Charge: 1074M / 3950M

Figure 22: Windows Task manager during PTT session.

The CPU usage during the session is 2%, which evidently confirms the efficiency of system resources consumed. On the right side, the network usage also changed slightly. The bandwidth usage is 0.09% during the session. This is due to the half-duplex nature of the application. The bandwidth usage on the remote side (receiver) interestingly remains 0%. The user's perspective of the application's performance was gathered from the questionnaires. The following graphical figure shows the response of the users with regards performance.

Question: *How would you performance of the system in terms of the response time?*

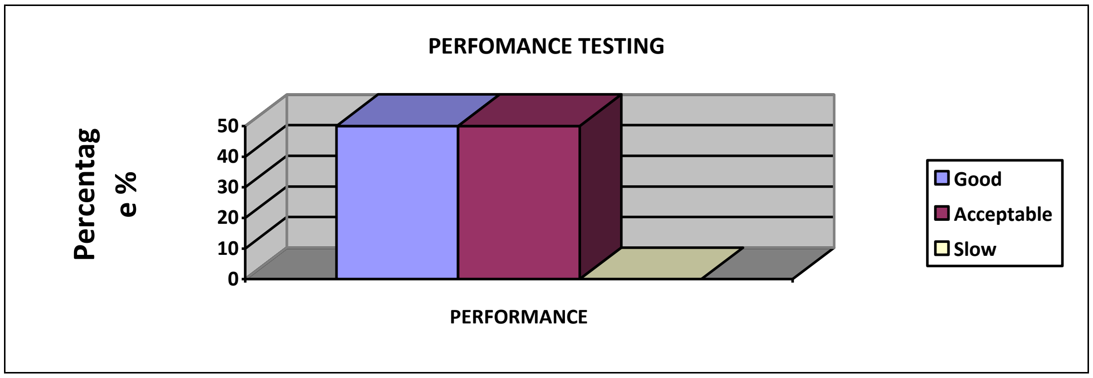


Figure 23: Performance testing from users.

SUMMARY

In this we discussed different testing strategies that were used during testing. We also discussed test results of various components of the application and gave graphical representation of some. The testing was successful as all the testing strategies conducted were successful.

The results shown that half-duplex push to talk application is efficient and very economically friendly. The application uses almost none of the system's resources as it only consumes 2% usage of the CPU and 0.09% of the bandwidth. The bandwidth usage on the phone could not be confirmed but however since it uses peer-to-peer, it is theoretically expected to be lesser than 0.09%. Thus we can conclude that the user's requirements gathered in chapter 1 were met. However since the application is still under development, we can say that the first prototype met the desired results. Thus following incremental model of software engineering life cycle, there is a need for improvements in other aspects of the application.

APPENDICES

APPENDIX A

PROJECT PLAN TERM 1

Meeting dates & times /Tasks	Mon 2th Feb 15h00	Mon 9th Feb 12h00	Mon 16th Feb 10h00	Mon 23th Feb 12h00	Mon 2nd March 10h00	Mon9th March 12h00	Mon 16th March 10h00
Comments				*meeting with (IM Venter)*	*n/a*	*meeting with (IM Venter)*	*meeting with (IM Vent*
Thesis Document				Create document using Honours Project Guidelines from the website as well as Thesis doc from Word	○ Check Index and add indexes ○ Bibliography – at least 5 entries. ○ Ask Prof I M Venter to read abstract and introduction	Finalise write-up. Let someone **proof** read your document!	
URD				Start write-up of URD.	Continue with URD write-up. Interview stakeholders.		Complete URD
RAD				Start write-up of RAD.	Write-up RAD		Complete RAD
Literature Survey	Familiarising yourself with your topic, and how it's implemented. Read and explore. literature on your topic.		Add all literature found to your bibliography – use the Harvard Notation	Read and explore. some more – use Google Scholar..	Keep on reading		
Presentation /deliverable	Write one paragraph that					Use thesis to Prepare slides	Mock presentation

	describes what you want to do, and why you want to do it					for mock presentation	
Website	Ask Frieslaar intelligent.networking@gmail.com	See previous week! Create website	Put plan onto website		Ask Frieslaar about the server. Add URD to website	Add RAD to website	

Table 1A describes the aims of each term. The aim of term one is done, only terms two, three and four still need be done.

Table 1A: planning for term 1.

APPENDIX B

PLANNING TERM 2

Tasks	**7th April**	**14th April**	**21st April**	**28th April**	**17th May**	**19th May**	**2th June**
Comments	*Combined meeting @14h00 Separate meeting@10 h00*	*Separate meeting with Mr. Norman*	*Combined meeting*	*Separate meeting with Mr. Norman Meet with R. Harris from Vodacom*	*Separate meeting with supervisor*	*Combined meeting*	*Separate meeting with supervisor*
Thesis Document	Edit/ update thesis – implement changes	Complete editing. Start with the write up of the UIS	Start write up of the OOA & complete the UIS write-up	Submit updated Thesis document, Project plan		Complete write-up of OOA. Complete write up.Hand in comp-leted document to supervisor	
OOD	Start thinking about The design			Start with the OOD	Submit initial OOD to MN	Refine the Design	
UIS		Start with User Interface Specificatio n	Complete UIS	Read about Python	Refine UIS	Update changes to UIS & If needed - update changes to UIS	
GUI & protot ype	Start with the prototype	Download Ftp libaries	Request Celtius PoC demo	Program GUI/ prototype	Consult M.s.c.students.	Program GUI/ prototype - busy	Finalise GUI/ prototype
Previous projects	Look at previous project	Look at previous projects	Work on the Symbian FTP library	Vodacom representative about PTT	Speak to long		

Presentation						*Prepare slides for mock presentation on 26th May*	*Mock presentation Update presentation for The 27thMay 2009*
Web-site	Update web site				Check and update		Put new plan, thesis & presentation on web site 26 May

Table 2A: detailed planning for term 2

APPENDIX C

PLANNING TERM 3

Tasks	**14th July**	**21th July**	**28th July**	**4rd Aug**	**11th Aug**	**18th Aug**	**25th Aug**	**28th Au 6th Sept**
Thesis Document	Finalise the editing of the documentation - & editing	Update any changes to the design – e.g. objects	Make changes to object's pseudo code as you develop the software, document all changes etc. <u>in the code</u> & **start** on the User's guide (User's Guide a deliverable for the next term only!)					**<u>Finalise Docume and han the 7th</u>**
Platform SIP Stack	Configure Pjsip symbian	Connect two pjsip clients through openser Speak to masters students	Try Asterisk Cell phones needed	Sort emulators problem with sound streaming	Try desktop implementation with SIP communicator/ try android emulator	Start working on sip communicator	Replace screenshots with screenshots of the current program (it will have changed)	
Configure SIP servers	Finish the Openser Configuration	Try to connect the clients through the SIP server. Speak to master students for help			Configure Asterisk			
Programming task	Start coding on carbide c++	Symbian Pjsip			java	Implement PTT using SIP communicator	Finalise programming & testing	

Testing and refining with a basic data set		Test the connection		Peer to peer connection between 2 clients made	No solution with sound streaming on the emulators.	Tried android emulator. Still no solution		
Presentation							Prepare slides	*Prepare presentatio*
Website	**Update NB**	**Update NB**					**Update NB**	**Update NE**

Table 3A: detailed planning for term 3

APPENDIX D

PLANNING TERM 4

Task	15th Sept	22th Sept	29th Sept	06th Nov	13th Nov	20th Nov	3th Nov	Presentation; 4th Nov
Thesis Document	Finalize document based on the feedback	Write installation Guide		Start writing up		Finalize Document.		PRESENTATION
Consultations	Francois	Seyi for Phone				Users who will test the application		
Implementation: PC	Add and fix some functionality. Fix the communication problems encountered.			Last week for the PC PTT problems.				
Implementation: Phone	Identify mobile phone to be used.		Nokia E series ideal	Port application on the mobile phone. Update firmware on the phone.		Make the application Half-duplex.	Try build a GUI	
Testing		Read on user acceptance Testing: SE	Read on validation Testing	Write testing Questionnaires	Test the PC PTT	Finalize testing on Both platforms.		
Website	All downloads be on the home page					Update Website		

Table 4A: detailed planning for term

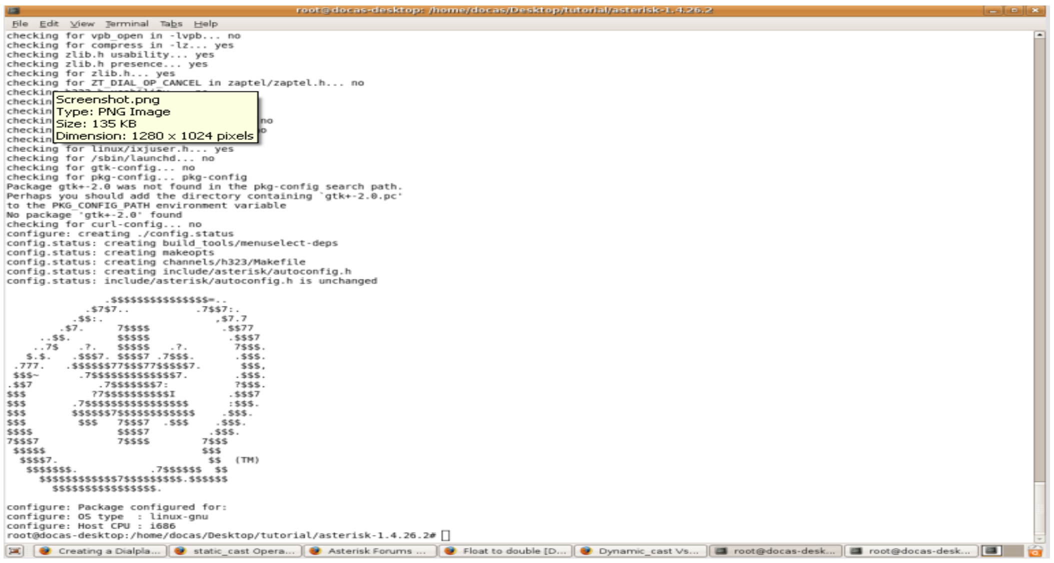
APPENDIX E

Installation Guide

SIP SERVER: INSTALLING ASTERISK

The Asterisk serves as a SIP server to relay the voice streams from source to destination. The server is run on Linux Ubuntu operating system. Firstly you need to download asterisk from http://downloads.digium.com/pub/asterisk/asterisk-1.4-current.tar.gz.

Step 1: On your terminal and untar Asterisk.tar.gz as ***tar zxvf asterisk.tar.gz*** under any folder name you like.
Step 2: Compiler asterisk by typing ***./configure*** in your terminal, the output should be the following



```
checking for vpb_open in -lvpb... no
checking for compress in -lz... yes
checking zlib.h usability... yes
checking zlib.h presence... yes
checking for zlib.h... yes
checking for ZT_DIAL_OP_CANCEL in zaptel/zaptel.h... no
checkin
checkin
checkin                                         no
checkin                                         o
checkin
checking for linux/ixjuser.h... yes
checking for /sbin/launchd... no
checking for gtk-config... no
checking for pkg-config... pkg-config
Package gtk+-2.0 was not found in the pkg-config search path.
Perhaps you should add the directory containing `gtk+-2.0.pc'
to the PKG_CONFIG_PATH environment variable
No package 'gtk+-2.0' found
checking for curl-config... no
configure: creating ./config.status
config.status: creating build_tools/menuselect-deps
config.status: creating makeopts
config.status: creating channels/h323/Makefile
config.status: creating include/asterisk/autoconfig.h
config.status: include/asterisk/autoconfig.h is unchanged

               .$$$$$$$$$$$$$$$=..
            .$7$7..          .7$$7:.
          .$$:.                 ,$7.7
        .$7.     7$$$$           .$$77
     ..$$.       $$$$$            .$$$7
    ..7$   .?.   $$$$$   .?.       7$$$.
   $.$.   .$$$7. $$$$7 .7$$$.      .$$$.
 .777.   .$$$$$$77$$$77$$$$$7.      $$$,
 $$$~      .7$$$$$$$$$$$$$7.       .$$$.
.$$7          .7$$$$$$$7:          ?$$$.
$$$          ?7$$$$$$$$$$I        .$$$7
$$$       .7$$$$$$$$$$$$$$$$      :$$$.
$$$       $$$$$$7$$$$$$$$$$$$    .$$$.
$$$        $$$   7$$$7  .$$$    .$$$.
$$$$             $$$$7         .$$$.
7$$$7            7$$$$        7$$$
 $$$$$                        $$$
  $$$$7.                       $$  (TM)
   $$$$$$$.           .7$$$$$$  $$
     $$$$$$$$$$$$7$$$$$$$$$.$$$$$$
       $$$$$$$$$$$$$$$$.

configure: Package configured for:
configure: OS type  : linux-gnu
configure: Host CPU : i686
root@docas-desktop:/home/docas/Desktop/tutorial/asterisk-1.4.26.2#
```

Figure 24: Snap shot of Asterisk.

Step 3: **make** ; **make install**

Step 4: start the Asterisk server as follows **asterisk** and then **asterisk –r** to connect

Registrations occur in the server through the following.

Step 1: ***sudo –i*** to login as root and type password

Step 2: ***vim /etc/asterisk/sip.conf*** the following is an example of how to add users to sip.conf file

```
[ivan]
Username=ivan
Authname=ivan
Secret=pwdivan
Type=friend
Host=dynamic
Context=tutorial
```

Installing clients

PC

Firstly you have to install Netbeans IDE and below are the steps.

1. Make sure that you meet the necessary requirements.
2. Download the NetBeans IDE 6.5 for JavaFX 1.1 installer file for Microsoft Windows (an EXE extension) or Macintosh OS X (a DMG extension).

Secondly download the source code from http://csunx.uwc.ac.za/~hkobo/Pc.rar and unzip the file. Open the sip communicator through Netbeans. Build the project and run it. Register the client in the Asterisk by entering the details specified in sip.conf, see step 4 above.

Mobile

Since we used PJSIP as our SIP stack, its only wise to follow their guide as it is well documented and broad. Firstly download the source code from http://csunx.uwc.ac.za/~hkobo/Pc.rar and follow the instruction guide on http://trac.pjsip.org/repos/wiki/DevelopingSymbianAppWithCarbide.

APPENDIX F

DEMOGRAPHICAL QUESTIONS

Participant	Gender	Level of study	Age	Use of IM	Frequency of IM usage
1	M	Postgraduate	26-28	Yes	Daily
2	M	Undergraduate	23-25	Yes	Once a month
3	M	Undergraduate	20-22	Yes	Daily
4	F	Undergraduate	20-22	Yes	Once a week
5	M	Postgraduate	23-25	Yes	Once a month
6	M	Postgraduate	20-22	Yes	Once a week

Table 5A: Demographics of user testing.

TASK 1: CHOOSE A SCENARIO

Scenario 1: secretary arrange a staff meeting

Suppose the secretary of UWC computer science department have to inform all the staff members about a meeting which is to take place. The secretary decides to use our PTT application instead of other means since all the staff members are on the same network.

Scenario 2: emergency

Honours computer science students stays late in the computer laboratory. Suppose there is an emergency during the course of the night. One of them decides to use our PTT application to call the campus control personnel

Scenario 3: social

Suppose two students who are text instant message users do not have sufficient rates to carry out their daily leisure pursuit. Since they both have Wi-Fi enabled mobile phones and they use the PTT to communicate as normal.

Did you manage to complete your task? Yes☐ NO ☐

If No, why?

Are you satisfied with the efficiency of the application?

Please motivate your answer:

How would you rate the usability of the application?

Very easy☐ Easy ☐ Moderately easy ☐ Difficult☐ Very difficult ☐

Motivate your answer?

How would you rate the interface?

PC PTT: 1 ☐ 2 ☐ 3☐ 4☐ 5☐

Mobile PTT: 1 ☐ 2 ☐ 3☐ 4☐ 5☐

What modification would you suggest if any?

What kind of problems have you encountered during the process?

Performance of the system in terms of the response time?

Good ☐ acceptable ☐ too slow ☐

Which one would you prefer click->talk->click->close or press and hold->talk->release->close? Motivate please:

In what kind of situation/s would it be more appropriate for you to use the application?

Any comments or recommendations to improve the application?

BIBLIOGRAPHY

[1]. http://www.cellular-news.com/story/15292.php (March 2009).

[2]. R. Thomas, "Parents Guide to MXit", 2006.
http://www.radioislam.co.za/Library/Family/mxit-ParentsGuide2MXit.pdf

[3]. R. Koivisto, *'Towards the Next Wave of Mobile Communications:* Push-to-Talk over a Cellular: Still Searching the Flow of **S**uccess*" In Proceedings of the Research Seminar on Telecommunications Business, TML-C19, pp 45-96, 2005.*

[4].http://www.forum.nokia.com/Tools_Docs_and_Code/Tools/Platforms/S60_Platform_SDKs/, September 2009.

[5].http://www.forum.nokia.com/Tools_Docs_and_Code/Tools/IDEs/Carbide.c++, September 2009.

[6]. http://www.pjsip.org, September 2009.

[7]. R.S. Pressman, *Software Engineering: A Practitioner's Approach,* 6th ed. McGraw-Hill, 2004, pp 406-408.

INDEX