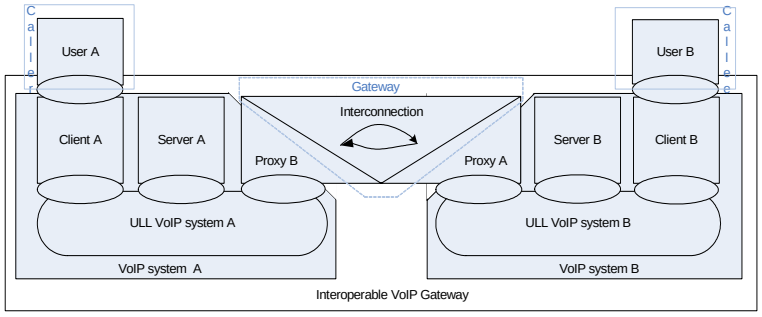
Universiteit Twente
de ondernemende universiteit

Towards interoperability between existing VoIP systems

Thesis for a Master of Science degree in Telematics
from the University of Twente, Enschede, the Netherlands
Enschede, February 26, 2008

Lianne Meppelink



GRADUATION COMMITTEE:
Dr. ir. B.J.F. van Beijnum (University of Twente)
Dr. ir. M.J. van Sinderen (University of Twente)
Prof. dr. ir. L.J.M. Nieuwenhuis (University of Twente)

Towards interoperability between existing VoIP systems

Thesis for a Master of Science degree in Telematics
from the University of Twente, Enschede, the Netherlands
Enschede, February 26, 2008

Lianne Meppelink

UNIVERSITY OF TWENTE,
Faculty of Electrical Engineering, Mathematics and Computer Science,
Department of Computer Science,
Division of Architecture and Services of Network Applications

Abstract

Since the invention of the telephone people started to have real-time conversations over a distance. With the rise of the Internet, another kind of real-time communication became popular. People were sending text messages to each other using Instant Messenger (IM) systems.

Nowadays the quality of the Internet infrastructure is good enough to have conversations over the Internet. The agreements about how to make a call using the Internet is called Voice over IP (VoIP). New VoIP systems came up and offered free VoIP calls.

Currently, many IM and VoIP systems exist. Quite often, a user has several VoIP and IM clients installed and possesses many accounts for all these several systems. Also applications came up to interconnect different IM systems and to interconnect different VoIP systems. Still, there is no universal solution that provides interconnection for all IM and VoIP systems. Furthermore, IM and VoIP system offers functionalities like login, buddy search, messaging, call setup and tear down and the actual call. The applications to interconnect different IM systems and different VoIP systems does not cover all the functionalities offered by the IM and VoIP systems.

In this thesis, five commonly used IM and VoIP systems, Windows Live Messenger, Google Talk, Yahoo! Messenger, ICQ and Skype, are presented. Each system is studied and compared to each other. Based on the characteristics, the differences and the similarities of the IM and VoIP systems, we made a design to provide interoperability between these systems.

In the design, the clients of existing VoIP and IM systems can be used. The VoIP and IM systems are interconnected by the use of a Gateway, which is situated between the VoIP systems. The presented solution is protocol independent, supports the functionalities login, buddy search, messaging and call (setup and tear down), and is extendable with more functionalities.

KEYWORDS: INTEROPERABILITY, GATEWAY, VOIP, IM, DESIGN, INTERCONNECTION, WINDOWS LIVE MESSENGER, MSN, YAHOO! MESSENGER, ICQ, SKYPE, GOOGLE TALK.

Preface

I would like to express my gratitude to my supervisors of the Univesity, especially Bert-Jan van Beijum who came in as my new supervisor and handled everything very well. I want to thank my study coordinator Jan Schut in special, for the mental support in hard times.

I would like to thank my boyfriend, Jasper Aartse Tuijn for supporting me. I also thank my family and fiends for making this possible and supporting me.

Last but not least I would like to thank my company, KPN Newtel Essence, where I started working while I was finishing this thesis. Thank you so much for the support and the believe in me!

Amersfoort, the Netherlands
15 January 2008

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides an introduction to the work reported in this Master thesis. It first presents the motivation behind the reported work followed by the objectives to be achieved. Subsequently, this chapter illustrates the approach that is followed in accomplishing these objectives. This chapter ends with a global overview of the structure of the remainder of this report.

1.1 Context

About 130 years ago, the telephone was invented. This invention gave the possibility to communicate over a distance. Soon the telephone earned a strong position in society. With the invention of the switchboard, the Public Switched Telephone Network (PSTN) was a fact.

About 100 years after the invention of the telephone, a new communication medium earned a strong position in society, called the Internet. To connect to the Internet in the early days, the PSTN was used. Nowadays also other networks, like cable and ASDL, are used to connect to the Internet.

The PSTN is a circuit switched network, and during a call, a dedicated circuit between the caller and callee is set up. No other callers or callees can enter this dedicated circuit. The Internet is a packet switched network, which provides the possibility for different nodes to be connected at the same time.

With the Internet, new communication applications appeared, such as e-mail, chat, voice and video. An example of a chat application is the Instant Messenger (IM). A contact list of an IM client shows the presence (online, offline, etc) of the buddies of the user. If a buddy is online, it is possible to chat, by sending (short) text messages. ICQ was the first IM application and was released in 1996.

Nowadays, most customers of the Internet are always connected to the Internet, and do not have to pay per online minute anymore. Applications can offer their functionalities for free, and thus the opportunity for free calls using a computer connected to the Internet. To connect computers and networks to the Internet, agreements on how to connect and how to send messages are needed. Such kind of agreements are specified in a protocol. For the interconnection of computer networks, the Internet Protocol (IP) has been standardized. The agreements of making a call over the Internet are therefore called Voice over Internet Protocol (VoIP).

VoIP conversations can be from PC-to-PC or from phone-to-PC and vice versa. PC-to-PC calls are offered mostly for free. An example of an application to make (free) calls using the Internet is "Skype". Skype has become the market leader very quickly, mostly because of its good voice quality. Skype uses its own proprietary protocol, which means the agreements on how to connect the Skype clients and their servers are not publicly available. Besides this market leader, there are several other IM and VoIP systems, like Google Talk, Yahoo! Messenger, ICQ and Live Messenger.

To have a call with a buddy, the user speaks into a microphone. This audio signal is an analog signal and is changed into an digital signal to transport to the client of the buddy. At the client of the buddy, the digital signal is converted to an analog signal again and sound at a speaker. This is possible in both directions.

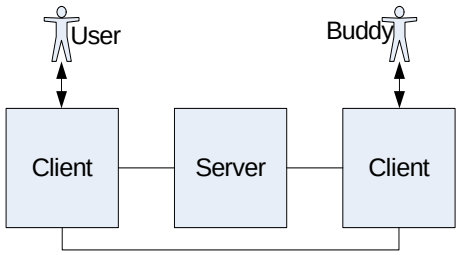


Figure 1.1: VoIP system architecture

An IM system is a system with the focus on Instant Messaging (but are nowadays often also able to do VoIP). VoIP systems have their focus on VoIP (but are often also able to do IM). At this report, the term VoIP system is used for both IM and VoIP systems, because the systems presented in this report all provide both IM and VoIP. When the term IM is used, this is used to emphasize it is only IM and not VoIP.

An typical VoIP system consists of two *clients* - the part of the VoIP system that is installed on the computer, which provides the interface to the user - and one *server* - the part of the VoIP system that provides authentication and that is the bridge between the clients. Figure 1.1 shows the architecture of such an typical VoIP system. When a *user* - a human being - logs in (the client is authenticated by the server), he will see his online *buddies* - the users of the VoIP system added to the contact list - and is able to communicate with these buddies. In Figure 1.1 the user is the person using one client and the buddy is the person using the other client, and vice versa.

To identify the client, a user name is used. At the server, this user name is associated to the IP address of the client. When the user sends a text message to his buddy, this message and the buddy name is sent by the client to the server. The server knows where to find the client of the buddy and forwards the message to the client. When the user starts a call, the call request is sent to the server and forwarded to the client of the buddy. When the call is accepted, a *peer-to-peer* connection is setup. A peer-to-peer (P2P) connection is a direct connection between the clients, which means the server is not used.

Buddies are added to the contact list by sending a buddy search request to the server. The

server searches for the client of the buddy and sends a request for acceptance to this client. When the buddy accepts, the server receives an acceptance, and sends an updated contact list, with the new buddy included, to the client of the requesting user.

The VoIP system offers two perspectives, (a) the external perspective and (b) the internal perspective. The external perspective shows the interfaces between the users and the VoIP system, the VoIP system is handled as a black box. The internal perspective shows the interactions inside the VoIP system, such as the requests and responses of the clients and server. We explain in the next section the need for an internal and external perspective.

1.2 Problem statement

In most cases, to be able to communicate, both users need to use the same VoIP system and users of different VoIP system are not able to communicate with each other. Applications like Trillian [2] makes it possible to send messages to several other VoIP systems without installing these VoIP systems. Also other companies and people have tried to provide interoperability between VoIP systems with other VoIP systems. For IM several solutions have been found to create interoperability between the systems. A start has been made to create interoperability between different VoIP services, but it is still not possible to find a universal solution to provide full interoperability for all IM and all VoIP systems.

To communicate with buddies, using a VoIP systems, consists of several steps. First we have the login phase, after a successful login, it is possible to perform a buddy search, send a message or make an audio call. This audio call can be subdivided into the setup and tear down of the call and the actual call. The biggest is the buddy search and the audio translation. How is it possible to search in another VoIP system for a user?

Most VoIP systems encode the data sent from the client to the server or from client to client. After receiving the encoded data, it is decoded. Some VoIP systems use proprietary *codecs* - for the (en)coding and decoding - and others use open source codecs. A solution to create interconnection between the VoIP systems is to translate the (audio) data sent by the clients and server into the data another VoIP system can understand. Probably both the control data (setup and tear down) and the audio data are encrypted. Because some VoIP systems use proprietary codecs, translating the encoded data is not always possible. This makes the need for a protocol independent solution: a solution using the external perspective.

1.3 Objective and research questions

Since an interoperable solution for VoIP does not exist, or at least no interoperable solution exist with full coverage of functionalities, it has to be designed. A design of this system is an architecture, which models the system in terms of functionality and structure [96]. The objective is

to design an interoperable VoIP system that, in an Internet environment, allows each user to use a single VoIP system for communication to all other VoIP users from different vendors.

The goal of the assignment is to design a system to provide interoperability for VoIP systems, which means that it should be possible to let clients of different VoIP systems communicate

with each other. To reach this goal, some research questions should be answered. The main question is:

How can VoIP systems interoperate?

This main question has the following sub questions:

A. What is the state of the art in VoIP?

B. What are the characteristics, differences and similarities of the VoIP systems and their accompanying protocols?

C. What are the requirements for a system to create interoperability between several VoIP systems?

D. What are the options to create interoperability between the VoIP systems?

E. How is the interconnection modelled and realized?

Sub question A asks for a clear overview of the way VoIP works. This is answered by providing the history of telephony and VoIP and by describing the protocols used for VoIP.

For sub question B, the systems MSN, Google Talk, Yahoo! Messenger, ICQ and Skype are discussed. They all use different protocols for the communication. Differences and especially the similarities of the VoIP systems and their accompanying protocols are important for the design of the interoperable system.

For the design of the interoperable system, requirements should be kept in mind. Different stakeholders have different requirements and are thus also important in the design process. These requirements are the answer to sub question C.

For research sub question D, several possible solutions are considered to find out what the best way to design the interoperable system is. These possible solutions are called "possible design approaches" in this report.

After selecting the desired design approach, this approach is converted into a design, which provide an answer to research sub question E.

1.4 Approach

The structure of the solution is based on the structure of Robert Parhonyi [93], and thus the approach is also based on this thesis. We divided the approach into four steps: *background information* to get acquainted with the basics of the subject, *design preparation*, to find out the best way to design the interoperable system, *interoperable system design*, the actual design and the *completion*, to conclude the thesis. Research sub questions A and B are answered in the *background information* section, and C, D and E in the *design preparation* section. The main question is answered during the *interoperable system design* and the *completion*. The four steps are subdivided into several main tasks:

1. Background information
    - Literature study on VoIP
    - Literature study on VoIP system and their accompanying protocols
    - Experiments to fill up the information about the protocols where literature lacks
    - Services of the VoIP systems presented to the users
    - Related work on interoperability between VoIP systems
2. Design preparation
    - Requirements for the interoperable system
    - Possible approaches for the design of the interoperable system
3. Interoperable system design
    - Behaviour of the interoperable system
4. Completion
    - Conclusion

The background information displayed in this report is mostly based on literature. Five commonly used VoIP systems are chosen: MSN, Yahoo! Messenger, Google Talk, ICQ and Skype. Each of these five VoIP systems uses different underlying protocol(s). Quite often, literature lacks to provide all information about these protocols. To complete the information collection, experiments with the five VoIP systems are done. Furthermore, the services presented to the users and related work on interoperability are defined.

Then we consider the requirements of several stakeholders. Furthermore, we consider possible design approaches and the choose the most feasible one.

Finally, the design of the interoperable system should concluded.

1.5 Structure

The structure of this report is based on the design process, and thus complies the approach.

Chapters 2, 3, 4 and 5 provide the background information and provide answers to research sub question A and B. The 2nd chapter provides an introduction to the telephone network and voice over IP (VoIP). Furthermore the protocols used by VoIP systems are discussed in detail. The 3rd chapter discusses five commonly used VoIP systems, named MSN, Google Talk, Yahoo! Messenger, ICQ and Skype. Based on the information gathered in this Chapter, the minimal and optional services of the VoIP systems are explained in Chapter 4. The 5th chapter provides work that is related to this research. Five applications that provide interoperability between several VoIP systems are discussed.

Chapters 6 and 7 provide the design preparation. These chapters answer the research sub questions C and D. The 6th chapter provides the requirements of several stakeholders that

should be considered for the design of the interoperable system. Chapter 7 provides several design solutions. The best and most interesting solution is chosen for the final design.

Chapter 8 gives the design of the interoperable system. This chapter shows the behaviour of the interoperable VoIP Gateway. The design is a validation of the chosen approach in Chapter 7. This is part of the answer to the main question.

Chapter 9 provides the completion of this thesis. It provides the answer to the research questions.

The Appendixes A and B consist of extra information about the State of the Art in VoIP and the research done at the five VoIP systems. The protocol messages sent by the VoIP applications are defined. Some of this information is presented in literature, other information is obtained by doing experiments: using the VoIP systems and sniffing the packets.

Chapter 2

State of the art in VoIP

This chapter provides a short introduction to the telephone network and to VoIP. The VoIP section describes also the protocols used for the setup and tear down of the call and the actual call. The first section explains the telephone network.

2.1 Introduction to the telephone network

At 14 February 1876 Alexander Graham Bell submitted the patent on the telephone, just two hours before Elisha Gray, seemingly his strongest rival. In 1871 Antonio Meucci already had a patent ready for the telephone. Unfortunately he lacked the money to submit the patent. The American Congress decided in 2002 Meucci as the real inventor of the telephone. [1]

During the early days of telephony, all telephones were directly connected to each other. Figure 2.1 illustrates this directly connected network. This network architecture did not scale up to a large number of telephones (e.g. if the network in Figure 2.1 grows by one telephone, five extra cables are needed). [82] [83]

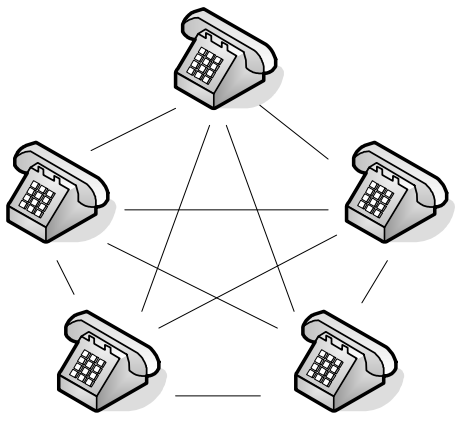


Figure 2.1: Early telephone system

In 1878 the first telephone switchboard was introduced. A switchboard makes it possible to switch between several lines, to create different connections, without connecting all phones to all other phones. Figure 2.2 shows a telephone network system using a switchboard.

The development of the first telephone switchboard is the beginning of the Public Switched Telephone Network (PSTN). In the beginning these switchboards were manned by operators who were called by the caller and had to plug in the line to create the connection between the caller and the called customer.

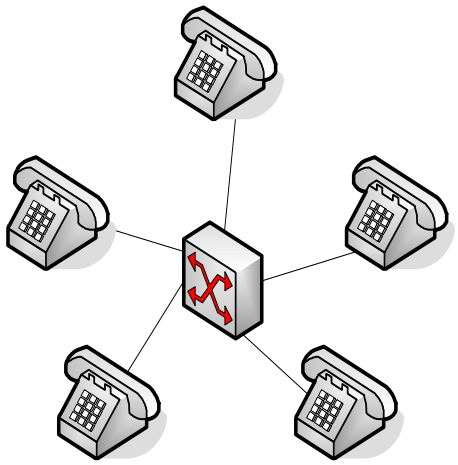


Figure 2.2: Telephone system with switchboard

In 1891 Almon Strowger patented the Strowger switch, a device which led to the automation of the telephone circuit switching. Now it was possible to eliminate the need for human telephone operators.

To keep matters simple, the telephone system was designed to perform many of its signaling operations by on-hook and off-hook operations. The on-hook operation means the telephone is not being used (the telephone handset is placed in a hook). The off-hook means the telephone is in use (the telephone handset is lifted from the telephone).

Figure 2.3 shows the sequence diagram of an example of a call. [82] explains these information in more detail. The figure shows on hook and off hook signaling between the originating office and the terminating office. The originating office is the caller side of the telephone station and the terminating office is the callee side of the telephone station. The on-hook and off-hook signaling between the originating office and the terminating office is a method used in the past. Nowadays, Signaling System 7 (SS7) or "out-of-band" signaling is the most widely used signaling system where the signals are transmitted on a separate physical channel from the call channel. It is a set of telephony signaling protocols which are used to set up the vast majority of the world's public switched telephone network telephone (PSTN) calls. SS7 is a means by which elements of the telephone network exchange information. Information is conveyed in the form of messages, like "The called subscriber for the call on trunk 11 is busy. Release the call and play a busy tone". [57] [42]

As mentioned by the name, the Public Switched Telephone Network uses a circuit switched network. A circuit switched network establishes dedicated circuits (or channels) between nodes and terminals over which the users can communicate. Each circuit cannot be used

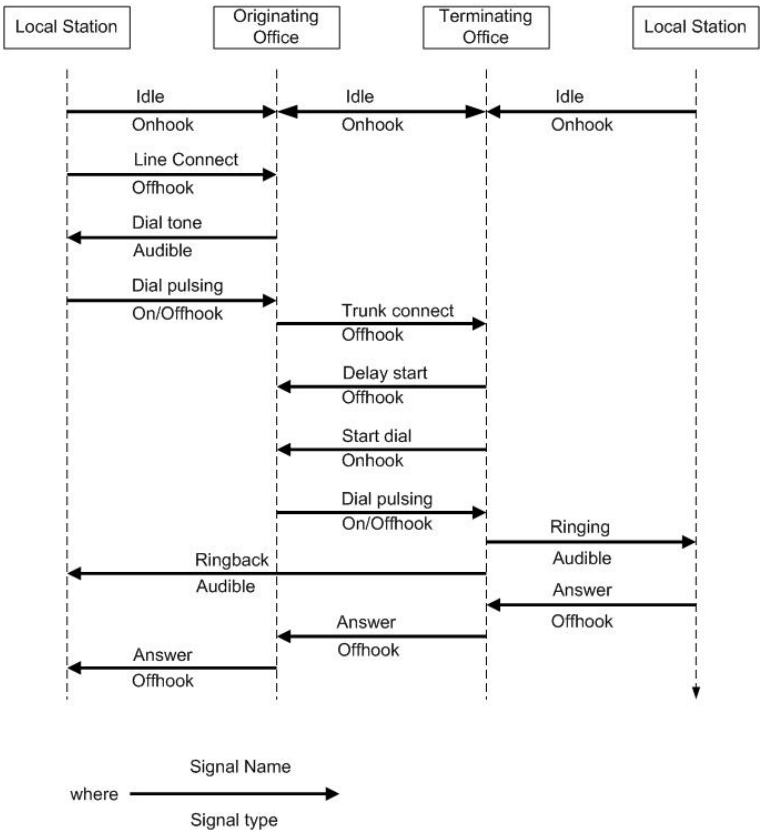


Figure 2.3: Example of initiating a call [82]

by other callers until the circuit is released and a new circuit is set up. Even if no actual communication is taking place in a dedicated circuit, that channel still remains unavailable to other users. Channels that are available for new calls are in idle state. [6]

2.2 Introduction to Voice over IP

Voice over IP (VoIP) makes it possible to make a phone call using the Internet. The data of this call travels over a packet switched network instead of a circuit switched network. A packet switched network is used in the Internet. Packets (units of information carriage) are routed between nodes over data links shared with other traffic. Packet switching is used to optimize the use of the bandwidth available in a network, to minimize the transmission latency (i.e. the time it takes for data to pass across the network), and to increase robustness of communication. [35]

VoIP was first demonstrated in the early 1980s when Bolt, Beranek, and Newman in Cambridge, Massachusetts, set up the "voice funnel" to communicate with team members on the West Coast as part of their work with the Advanced Research Projects Agency (ARPA). The voice funnel digitized voice, arranged the resulting bits into packets, and sent them through the Internet [3]. The development of IP telephony expanded in 1995 when the Israeli com-

pany VocalTec released their softphone InternetPhone that enabled computer-to-computer IP telephony. The first gateway between IP networks and the PSTN was released in the market in 1996. The Israeli company DeltaThree offered a telephone-to-telephone communication service over IP networks in 1997. [4]

The Open Systems Interconnection Basic Reference Model (OSI Reference Model or OSI Model for short) [5] is a layered, abstract description for communications and computer network protocol design, developed as part of Open Systems Interconnection initiative. It is also called the OSI seven layer model. The OSI model shows seven layers, called the Application layer, Presentation layer, Session layer, Transport layer, Network layer, Datalink layer and Physical layer.

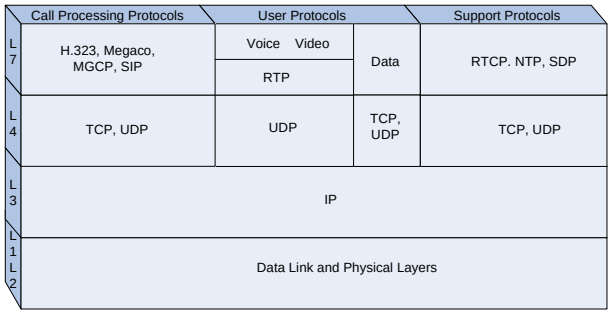


Figure 2.4: The Internet Call Processing Model

On top of the network protocol IP two transport protocols are situated, called UDP and TCP. Figure 2.4 shows these protocols situated in the OSI model. TCP is connection oriented and takes care of retransmissions. It ensures quality of the transport, i.e. packets are in order and no packets are missing. For VoIP it is not a problem if once in a while a packet is dropped, because probably it is not even noticed by the users. If it is noticed by the users, they can ask each other to repeat the text. Therefore often UDP is used for VoIP, since it is connectionless and has less overhead then TCP. UDP does not retransmit lost packets and it still uses the IP stack so packets will not necessarily be received in the order they were sent. Therefore other mechanisms to ensure the reliability of the packet stream are needed. The Real Time Protocol (RTP) helps to build the packet stream in the order as the stream was sent, and different voice compression methods have the ability to regenerate lost packets. To initiate a VoIP session, there is a need for information exchange between the clients before the session can start. The most commonly used protocol is the use of the control protocols Session Initiation Protocol (SIP) and H.323. Figure 2.4 shows the position of H.323 and RTP in the OSI model; H.323 and RTP are discussed in more detail in Paragraph 2.3.

VoIP can be subdivided into three events:

1. Setup
2. Conversation
3. Tear down

The setup is done with protocols like SIP or H.323. The conversation can be handled with RTP and the tear down again with SIP or H.323

VoIP has some (dis)advantages in comparison with the PSTN. The advantages of VoIP are [82]:

- It reduces costs.
  - All data (speech and packets) is sent over the same infrastructure, so less cables are needed and it is easier to maintain.
- It has more functionalities
  - The network is easier to expand
  - The network is less fragile
  - The implementation of a new function is much faster
  - It is possible to bring your phone(number) with you and use it somewhere else

Some disadvantages are [82]:

- A specific QoS (Quality of Service) is needed
- Speech needs priority (above for instance downloads), without priority, QoS can not be promised

In literature both the terms "VoIP" and "Internet telephony" are used to describe the possibility to use a computer to make a call. Because of the different explanations, the exact differences between the two terms are unclear. In this report, they are treated as the same.

The overall model to have both PC-to-PC and PC-to-phone (and vice versa) calls is called the Internet Call Processing model [82]. In the preceding paragraphs, the protocols SIP and H.323 are already mentioned. These protocols and some others are used in this Internet Call Processing model and are called the Internet Call Processing protocols. The following sections will first explain the model and afterward the protocols.

2.3 The Call Processing Model

Figure 2.5 shows the VoIP topology [82]. It shows what the architecture for a PC-to-PC, PC-to-phone and phone-to-PC call looks like. This can be P2P or with the use of a gateway. In case of a PC-to-phone or vice versa call, a gateway should always be used to switch between the circuit switched network and packet switched network.

The system handles the telephone's control operations, like off-hook and on-hook operations. These signals are converted into binary bits (packets) and later on encapsulated into the IP datagram for forwarding the packet. At the receiving end, the process is reversed.

The computer sends and receives packets to and from the gateway. The telephone on the other end is receiving tones. The gateway converts the IP based telephony message to the conventional telephone format (e.g. SS7 message syntax) and the other way around.

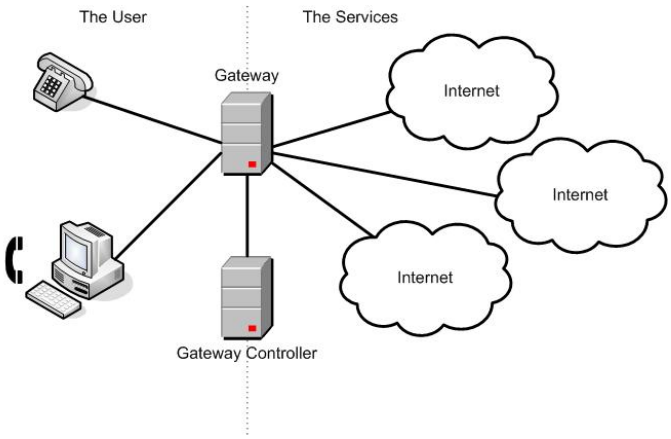


Figure 2.5: VoIP topology

The key components used in this operation are the Gateway and a node known by three names. This node is called a Gatekeeper in H.323, a Call Agent in MGCP and a Media Gateway Controller in Megaco [82]. In this report the term Gateway Controller is used.

The Gateway is responsible for the connection of the physical links of the various systems. The Gateway Controller is the overall controller of the system and thus the Gateway is a slave to the master Gateway Controller.

The OSI reference model is used to show the Internet Call Processing Model in Figure 2.4. The figure shows the following protocols [82]:

- *Call Processing Protocols:* These protocols form the basis for most of the call processing for voice and video services. They take care of the connection setup and tear down. The conversation itself is handled by the user protocols. Examples of the call processing protocols are H.323, Megaco, MGCP and SIP and will be explained later in more detail.
- *User Protocols:* These protocols are used to send the user voice (audio), video and data traffic. Examples are RTP, FTP and e-mail.
- *Support Protocols:* These protocols support the Call Processing Protocols. They do not control a call per se, but assist the Call Processing Protocols. Examples are RTCP, NTP and SDP.

2.4 The Call Processing Protocols

Internet Call Processing protocols take care of the setup and tear down of the session. The actual conversation is handled by the user protocols. There are different Call Processing Protocols; the four most important ones are:

- H.323

- The Media Gateway Control Protocol (MGCP)
- Megaco/H.248
- The Session Initiation Protocol (SIP)

In 2001, H.323 v2 was the most used standard. Figure 2.6 and Figure 2.7 show the different protocols and their use. Both diagrams have information from the year 2001. The lower bar in both diagrams is the current use of the protocol in products. The upper bar in both diagrams is the expected use of the protocol in new products. According to [82] SIP is expected to be the most used standard and according to [90] this will be Megaco. It is clear to see the shifting of the use of standards. Nowadays, SIP [43] is most used. [41]

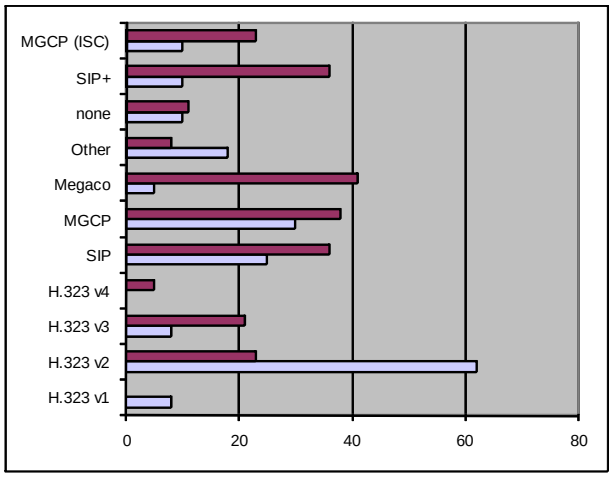


Figure 2.6: Use of protocols according to [90]

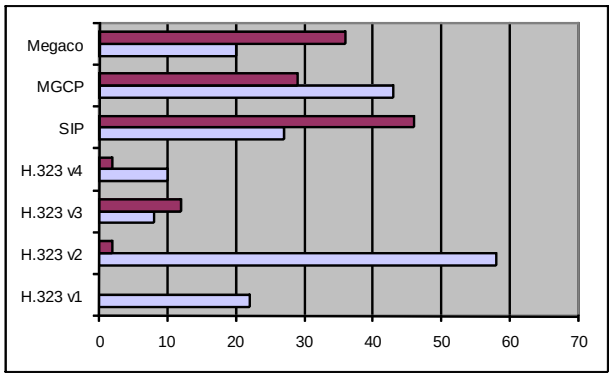


Figure 2.7: Use of protocols according to [82]

2.4.1 H.323

H.323 is a standard approved by the ITU in 1996 to promote compatibility in video conference transmissions over IP networks. H.323 was originally promoted as a way to provide consistency in audio, video and data packet transmissions. Although it was doubtful at first whether manufacturers would adopt H.323, it is now considered as a standard for interoperability in audio, video and data transmissions, as well as Internet phone and VoIP because it addresses call control and management for both point-to-point and multipoint conferences as well as gateway administration of media traffic, bandwidth and user participation. [85]

H.323 was originally designed to support multimedia services over a LAN. H.323 uses the H.245 protocol for control operations, the H.332 protocol for managing large conferences, H.225 for connection management, H.235 for security support, T.120 for document support for conferences, and H.246 for circuit-switch interworking. Furthermore some signaling protocols of ISDN could be borrowed [82]. H.323 was not designed to interwork with Web architectures, like HTTP. Its data structures and transfer syntaxes are based on the OSI Presentation Layer (layer 6 of the OSI Model). Companies as Microsoft and IBM use this protocol in many of their VoIP products. [83]

Services
The H.323 user terminal can provide real-time, two-way audio, video or data communications with another H.323 user terminal. The terminal can also communicate with an H.323 Gateway, which can also operate as a Multipoint Control Unit (MCU). The MCU supports multi-conferencing between three or more terminals and Gateways.

H.323 invokes several operations to support end-user communications with other terminals, Gateways and MCU (all these devices are called endpoints). Sometimes these operations are called phases. The seven major operations are: [82]

- **Discovery:** The discovery phase starts with finding a Gatekeeper with which it can register. The endpoint and the Gatekeeper exchange addresses. The IP multicast address 224.0.1.4 is reserved for Gatekeeper discovery.
- **Registration:** At this point the endpoint is identified (end-user terminal, Gateway, MCU) and joins the calling zone; the zone that is part of a network controlled by the Gatekeeper.
- **Connection Setup:** A connection is set up between two endpoints for the end-to-end call.
- **Capability Exchange:** Any multimedia traffic sent by one endpoint should be received correctly by the endpoint. This operation ensures this and allows the endpoint and the Gatekeeper to negotiate their capabilities.
- **Logical Channel Exchange:** This phase is used to open one or more logical channels to carry the traffic.
- **Payload Transfer:** In this phase the traffic is exchanged.
- **Termination:** Finally all logical channels should be released.

Protocol flow
Figure 2.8 shows the protocol flows for the connection setup and termination. Endpoint (EP)

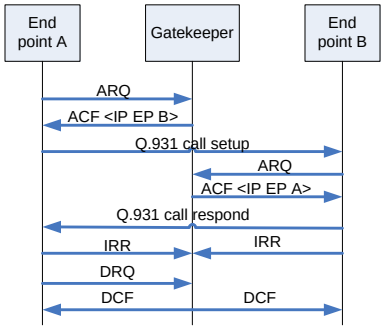


Figure 2.8: H.323 protocol flow

A sends an ARQ (Admission Request) to the Gatekeeper, the Gatekeeper returns an ACF (Admission Confirmation) with IP address of EP B. EP A sends Q.931 Call setup messages to EP B and EP B sends the Gatekeeper an ARQ, asking if it can answer call. The Gatekeeper returns an ACF with IP address of EP A and EP B answers and sends Q.931 to EP A. Both endpoints send a IRR (Information Request Response) to the Gatekeeper. If EP A disconnects the call, a DRQ (Disconnect Request) is sent to Gatekeeper. The Gatekeeper sends to both Endpoints a DCF (Disconnect Confirmation).

2.4.2 Megaco / H.248

The Media Gateway Control Protocol (Megaco) is a result of joint efforts of the IETF and the ITU-T Study Group 16. Therefore, the IETF defined Megaco is the same as ITU-T Recommendation H.248 [83] [29]. Megaco/H.248 is for control of elements in a physically decomposed multimedia gateway, which enables separation of call control from media conversion.

Services
Megaco/H.248 addresses the relationship between the Media Gateway (MG), which converts circuit-switched voice to packet-based traffic, and the Media Gateway Controller. There are two basic components in Megaco/H.248: terminations and contexts. Terminations represent streams entering or leaving the MG (for example, analog telephone lines, or RTP streams). Terminations have properties, such as the maximum size of a jitter buffer, which can be inspected and modified by the MGC.

All Megaco/H.248 messages are in the format of ASN.1 text messages. Megaco uses a series of commands to manipulate terminations, contexts, events, and signals. The following is a list of the commands:

- **Add** - The Add command adds a termination to a context. The Add command on the first Termination in a Context is used to create a Context.
- **Modify** - The Modify command modifies the properties, events and signals of a termination.

- **Subtract** - The Subtract command disconnects a Termination from its Context and returns statistics on the Termination's participation in the Context. The Subtract command on the last Termination in a Context deletes the Context.

- **Move** - The Move command atomically moves a Termination to another context.

- **AuditValue** - The AuditValue command returns the current state of properties, events, signals and statistics of Terminations.

- **AuditCapabilities** - The AuditCapabilities command returns all the possible values for Termination properties, events and signals allowed by the Media Gateway.

- **Notify** - The Notify command allows the Media Gateway to inform the Media Gateway Controller of the occurrence of events in the Media Gateway.

- **ServiceChange** - The ServiceChange Command allows the Media Gateway to notify the Media Gateway Controller that a Termination or group of Terminations is about to be taken out of service or has just been returned to service. ServiceChange is also used by the MG to announce its availability to an MGC (registration), and to notify the MGC of impending or completed restart of the MG. The MGC may announce a handover to the MG by sending it ServiceChange command. The MGC may also use ServiceChange to instruct the MG to take a Termination or group of Terminations in or out of service.

All of these commands are sent from the MGC to the MG, although ServiceChange can also be sent by the MG. The Notify command, with which the MG informs the MGC that one of the events the MGC was interested in has occurred, is sent by the MG to the MGC.

Protocol flow
Figure 2.9 shows the protocol flows for the setup of a call of Megaco. At the first part, the Off-hook state shows the Local Gateway would like to have a call. After accumulating the digits (the address of the Remote Gateway), the connection can be setup, which is noticed to the Remote Gateway by a Ring. After an acknowledgement, there is a Ring back at the Local Gateway. When the Remote Gateway reaches the Off-hook state, the call has been setup.

2.4.3 MGCP

The Media Gateway Control Protocol (MGCP) is published as RFC 3435, which obsoletes an earlier definition in RFC 2705 and integrates the Simple Gateway Control Protocol (SGMP) and the Internet Protocol Device Control (IPDC) specification. The Gateway Controller is called a Call Agent.

MGCP as primarily developed to address the demands of carrier-based IP telephone networks. MGCP is a complementary protocol for both H.323 and SIP, which was designed as an internal protocol between the Media Gateway Controller and the Media Gateway. In MGCP, an MGC primarily handles all the call processing by linking with the IP network through constant communications with an IP signaling device, for example an SIP Server or an H.323 gatekeeper.

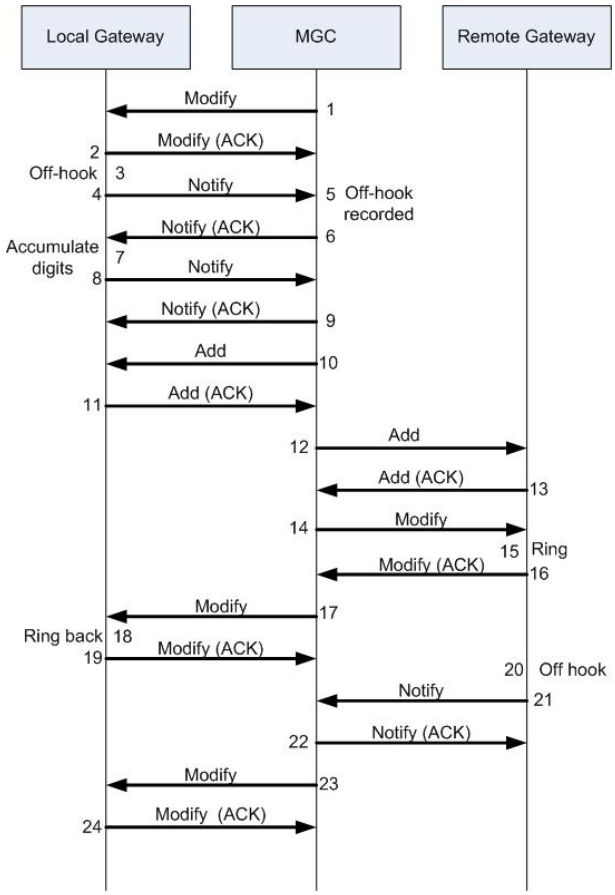


Figure 2.9: Megaco protocol flow

Services

MGCP is comprised of a Call Agent, one media gateway (MG) which performs the conversion of media signals between circuits and packets, and one signaling gateway (SG) when connected to the PSTN (Public Switched Telephone Network). MGCP is widely used between elements of a decomposed multimedia gateway.

Call Agents come with the capability of creating new connections, or modify an existing connection. Generally, a media gateway is a network element which provides conversion between the data packets carried over the Internet or other packet networks and the voice signals carried by telephone lines. The Call Agent provides instructions to the endpoints to check for any events and - if there is any - create signals. The endpoints are designed in such a way as to automatically communicate changes in service state to the Call Agent. The Call Agent can audit endpoints and the connections on endpoints. [56]

MGCP uses the following commands [28]:

- **AUEP - Audit Endpoint:** Used by the Call agent to query (the state of) a Media Gateway

- **AUCX - Audit Connection:** Used by the Call agent to query (the state of) a Media Gateway
- **CRCX - Create Connection:** Used by a Call Agent to manage an RTP connection on a Media Gateway
- **DLCX - Delete Connection:** A Media Gateway can also send a DLCX when it needs to delete a connection for its self-management
- **MDCX - Modify Connection:** Used by a Call Agent to manage an RTP connection on a Media Gateway
- **RQNT - Request for Notification:** Used by a Call Agent to request notification of events on the Media Gateway, and to request a Media Gateway to apply signals
- **NTFY - Notify:** Used by a Media Gateway to indicate to the Call Agent that it has detected an event for which the Call Agent had previously requested notification of
- **RSIP - Restart In Progress:** Used by a Media Gateway to indicate to the Call Agent that it is in the process of restarting

Protocol flow
Figures 2.10 and 2.11 shows the call setup and tear down of MGCP. The user starts the setup by starting the Off-hook state. He hears the dial tone and enters the digits (the address of the callee). When the connection is setup to the callee, the user hears a ringtone. When the user ends the connection, the on hook state is reached again.

2.4.4 SIP

The Session Initiation Protocol (SIP) [94] is a control protocol which can set up, modify and tear down sessions between session users. An interesting part of SIP is the support for mobility. If a user registers his or her location with a SIP server, SIP will direct SIP messages to the user or invoke a proxying operation to another server to a user's current location.

SIP can be used with UDP or TCP, which is in an intelligent way implemented: UDP will be used first. If problems arise and a timeout will happen, TCP will be used as a fallback.

SIP is an attractive control tool for IP telephony because:

- It can operate as stateless or stateful. It provides good scalability and robustness for the stateless implementation.
- It uses many of the formats and syntax of HTTP, thus providing a convenient way of operating with ongoing browsers.
- The SIP message (the body) can be used in any syntax, such as Multipurpose Internet Mail Extension (MIME) or the Extensible Markup Language (XML).
- It identifies a user with a URI, thus providing the user the ability to initiate a call by clicking on a Web link.

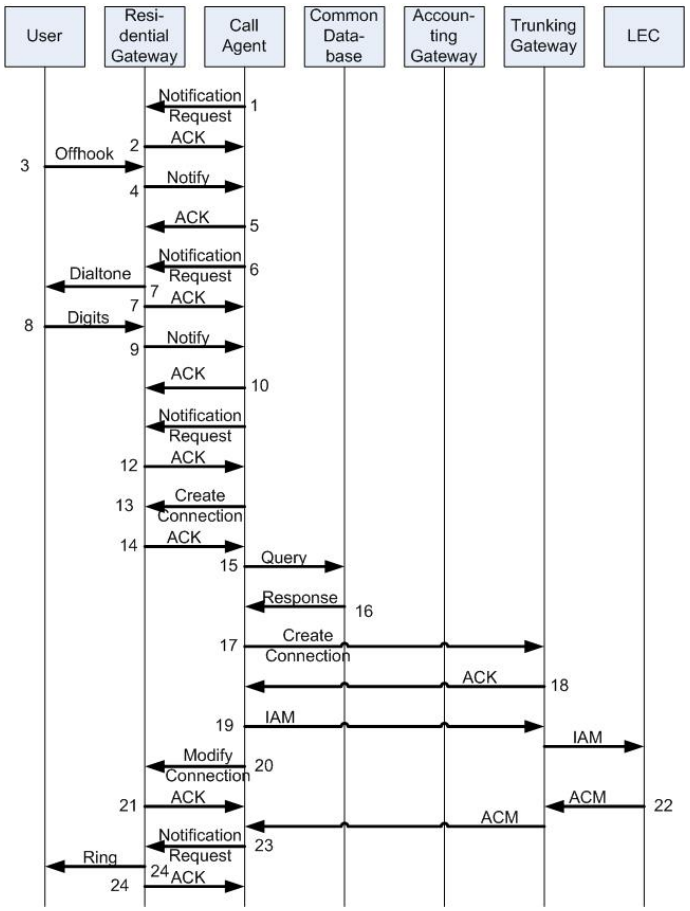


Figure 2.10: MGCP protocol flow part 1

Services
SIP supports five aspects regarding the establishment and termination of communications sessions: [84]

- **User location:** Determination of the destination end system
- **User availability:** Determination of the willingness of the call party to accept a call to this device
- **User capabilities:** Negotiation of the session parameters
- **Session setup:** Establishment of the session
- **Session management:** Modification and termination of the session

A SIP architecture consists of three types of elements: SIP User Agents, SIP servers, and Location Servers. These elements are shown in Figure 2.12. SIP User Agents are end devices such as: SIP phones, a software SIP client on a computer or handheld, or a gateway to other networks, typically a PSTN gateway. They can originate SIP requests and send and receive

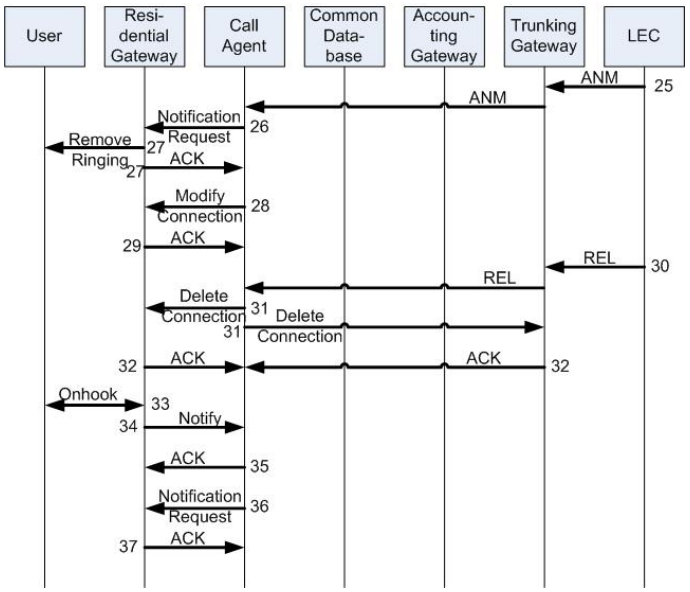


Figure 2.11: MGCP protocol flow part 2

media. A SIP User Agent contains two parts: the User Agent Client (UAC) part initiates requests and the SIP User Agent Server (UAS) part responds to received requests.

Three types of SIP servers exists: Proxy, Redirect, or Registrar servers [94]. When receiving requests from a SIP user agent or a SIP proxy for a call setup, it forwards the requests towards the intended destination. A SIP Redirect server receives requests from User Agents or proxies and returns redirection information to contact an alternative URI. A SIP Registrar receives SIP registration requests and updates Location Servers with SIP User Agent information. A Location Server is a database that contains user, routing, and location information.

SIP uses several operations (according to RFC254 and RFC3261 [94]):

- ACK: Acknowledgement of a received message
- BYE: End of the session
- CANCEL: Cancellation of the request
- INVITE: Invitation of a user to a call
- OPTION: Request for information about the possibilities of the server
- REGISTER: Registration of the user agent

Protocol flow

Figure 2.13 shows an example of a complete SIP session, starting with the registration, followed by the call setup and the call and finishing with the ending of the call. The INVITE method is explained in more detail in Appendix A.

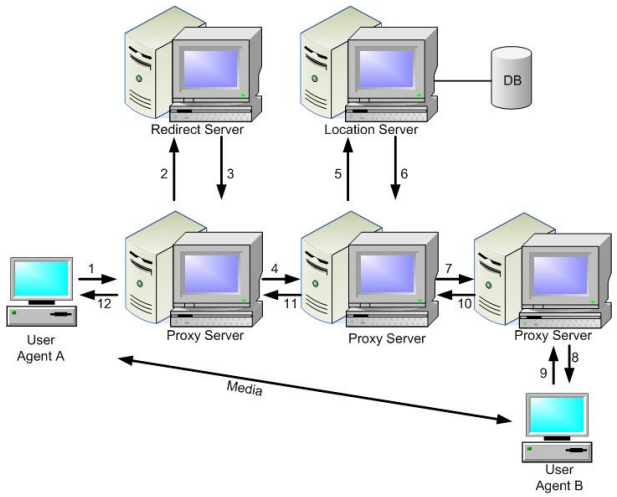


Figure 2.12: SIP elements

2.4.5 Summary

Table 2.1 shows an overview of the four Call Processing Protocols as described before. [81] [83]

2.5 The User Protocols

SIP is a protocol used to start up VoIP calls, not to send the actually desired data over the network. To send the data the User Protocol Real Time Protocol (RTP) and the Support Protocol 'Real Time Control Protocol' (RTCP) are used.

2.5.1 Real Time Protocol (RTP)

The RTP standard defines a packet structure for use in real-time applications, amongst others it includes fields for timestamps and sequence numbers necessary for synchronization. RTP runs over UDP and is often viewed as a sub layer of the transport layer. However technically the RTP protocol is implemented in the application layer. Data encapsulation is performed at the end systems and the protocol does not provide any mechanism to prevent out of order packets or quality mechanisms. In order to provide control functions RTP is usually complemented with the RTCP protocol, described in the next section.

2.6 The Support Protocols

This section describes the support protocols Session Description Protocol (SDP), Network Time Protocol (NTP) and RTP Control Protocol (RTCP).

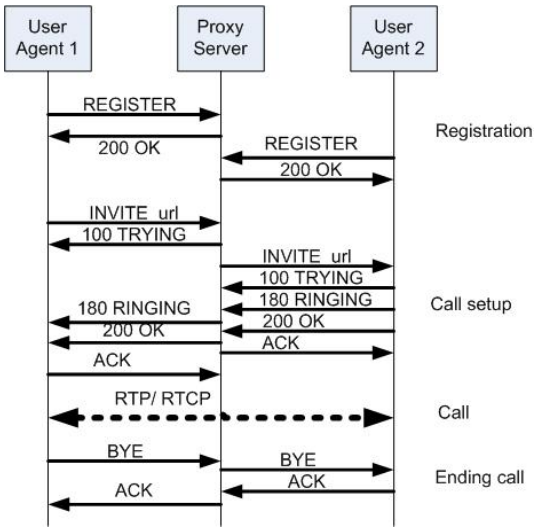


Figure 2.13: Possible protocol flow of SIP

2.6.1 RTP Control Protocol (RTCP)

The RTP control protocol (RTCP) is usually implemented in combination with the RTP protocol and is based on the periodic transmission of control packets. RTCP does not send a payload with application data, instead it sends statistical information that is necessary for the application to provide feedback on the VoIP quality. The way this information is used by the application layer is not defined in the RFC3550 standard, and depends strictly on the application. RTCP and RTP packets are distinguished by using different port numbers. Typically the RTCP port is one higher than the RTP port, RTCP also uses UDP as the transport protocol.

The statistical information RTCP gathers are for example information about loss, jitter, feedback and the round trip time. An application can use these information to increase the QoS, lower the bandwidth by using another bandwidth, or some thing else.

2.6.2 Session Description Protocol (SDP)

SDP is a text-based session description format. It is used by SIP to convey information about a media session, e.g. type of media, the transport protocol in use, format of the media, multicast addresses, port numbers, etc. [86]. SIP carries SDP encoded in MIME as a message body in a SIP message. SDP supports use of IPv6 addresses.

2.6.3 Network Time Protocol (NTP)

The Network Time Protocol (NTP) is a protocol for synchronizing the clocks of computer systems over packet-switched, variable-latency data networks. NTP uses UDP port 123 as its transport layer. It is designed particularly to resist the effects of variable latency. NTP is needed in VoIP for the QoS, if for instance the end-to-end delay is measured, the time should be synchronised.

	H.323	**SIP**	**MGCP**	**MEGACO**
Architectural Model	Peer-to-peer	Peer-to-Peer	Master/slave	Master/slave
Media types	Voice, video, limited data	Voice, video, data	Voice	Voice, video
Network scope	Intra, extra and Internet	Inter, Extra and Internet	Intranet only	Intranet only
Extensibility	Low	High	Medium	Medium
Scalability	Medium	High	Low	Low
Ease of deployment	Low	High	Medium	Medium
Standardization	ITU-T	IETF	IETF	IETF and ITU-T
Functionality	Establishing connection	Establishing connection	Signaling Control Information	Signaling Control Information
Advantages	Quite prevalent and powerful (especially for conferencing)	Relatively simple and easily extensible. It is published as RFC 2543, thus a formal IETF standard	Web-based and relatively simple	Web-based and relatively simple
Disadvantages	Complex built on ITU-T OSI layer 6	-	Lots of redundancy relative to Megaco, but not an Internet standard	-
Endpoints	Smart	Smart	Dumb	Dumb
Network	Dumb	Dumb	Smart	Smart

Table 2.1: Summary of the Internet Call Processing protocols

NTP is one of the oldest Internet protocols still in use (since before 1985). NTP was originally designed by Dave Mills of the University of Delaware, who still maintains it, along with a team of volunteers. NTP is published in RFC 1119.

NTP can operate above UDP or TCP. The time is displayed as GMT.

2.7 Conclusions

This chapter gave a short introduction to telephony and VoIP. Furthermore it provided information about the Call Processing Protocols (H.323, MGCP, Megaco and SIP), the User Protocol (RTP) and the Support Protocols (RTCP, SDP and NTP). These different protocols are complementary; they offer the functionality of VoIP, although they cannot do the setup, tear down and actual call all by there self. The four Call Processing Protocols support the

call setup and tear down, but are different. H.323 and SIP have smart endpoints and a dumb network and Megaco and MGCP have dumb endpoints and a smart network. Hence, the protocols are not equal, and thus have to be translated to make the protocols compatible.

SIP is an open and emerging protocol, unfortunately not all protocols used for VoIP are open. Proprietary protocols are not easily translated, and quite often are not possible to translate at all. This means it is not possible to interconnect all of the protocols and thus for the interconnection, a protocol independent solution should be found.

The next chapter shows the commonly used VoIP systems, and shows that the VoIP systems not always use the VoIP protocols as described in this chapter. Some of the VoIP systems use their own, proprietary protocols and are thus not open.

Chapter 3

Overview of VoIP Systems

Nowadays, many Instant Messenger (IM) and VoIP systems are available. Instant Messenger systems, which started with only messaging have voice calls included now. Examples of these systems are Live Messenger (MSN), Google Talk, Yahoo! Messenger and ICQ. The Skype system has entered the market with the audio as main target and is currently the market leader on VoIP.

Remember that in this report the term "VoIP system" means that the system offers both IM and VoIP functionality.

This chapter provides an overview of the most widely used VoIP systems. Information on these VoIP systems are found in existing reports, papers and with experiments. These experiments are performed by the author of this report, by using Wireshark [70], which is an application to snif packets. Sequence diagrams are produced using Visual Ether [69] and are summarized in readable sequence diagrams by the author. Those latter sequence diagrams are published in Appendix B

Based on the existing VoIP systems, a generic system has been made to refer to. The overview of every VoIP system is divided into thee parts:

- Features
- Entities
- Protocol

The 'features' paragraphs sum up the functionalities of the VoIP system and compare them to a generic list of functionalities. The 'entities' paragraphs discuss the entities of the VoIP systems. The 'protocol' paragraphs discuss the several protocols used by the VoIP system.

The focus in this chapter will be on Login, Buddy search, Message sending and Calling. More information about File transfer can be found in Appendix B. Detailed information about the protocols of the VoIP systems, such as the protocol flows can also be found in Appendix B.

3.1 Aspects of VoIP systems

This chapter discusses several VoIP systems and compares them with a generic VoIP system, shown in the next sections. This generic VoIP system has been developed based on analyses

of all described VoIP systems. The reason for this generic VoIP system is to compare it to the existing VoIP systems, to show easily the differences, and similarities between the several VoIP systems and their accompanying protocols.

3.1.1 Features

The VoIP systems have many features. The main services of most VoIP systems are:

- **Login**: Authentication with username (name, email address or number) and password
- **Buddy search**: Every user has a contact list with one or more buddies. To expand this list, other contacts can be searched for and added. We only focus on the searching for a buddy and expanding of the contact list
- **Messaging**: Typing in a message and sending it to one or more selected user(s)
- **Calling**: Setting up a conversation requires first the invitation of the caller, which the callee might accept. After acceptance, the conversation starts. This report only focuses on PC-to-PC calls

3.1.2 Entities

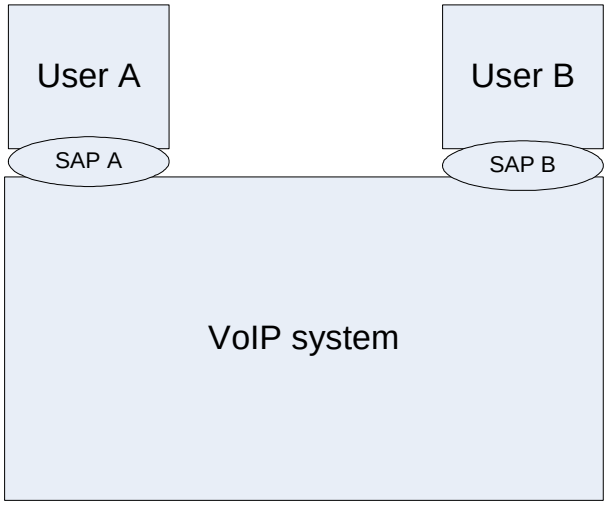


Figure 3.1: Entities of a VoIP system

Figure 3.1 shows an overview of a VoIP system. It is based on the five architectures of the VoIP systems discussed in this chapter. Each VoIP system has two users. A user only notices it is using the system to communicate with another user. He does not notice what is happening inside the system.

Figure 3.1 shows in between the users and the VoIP system Service Access Points (SAPs). These SAPs show the interactions between the VoIP system and the users. It defines the services.

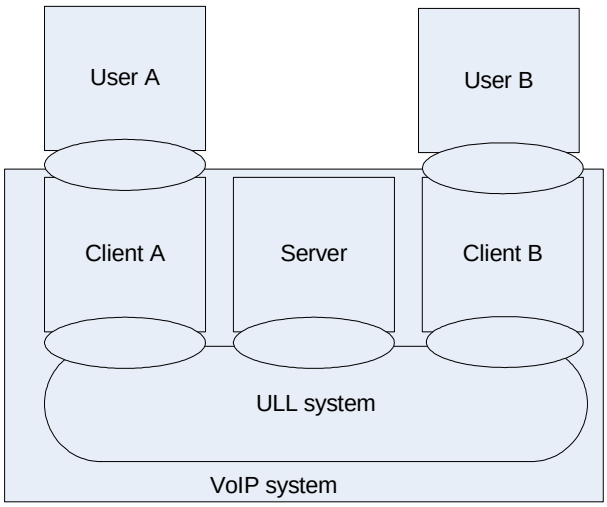


Figure 3.2: Entities of a VoIP system

Figure 3.2 shows a decomposition of Figure 3.1. Inside the VoIP system, two clients, a server and an Underlying Layer (ULL) of the VoIP system are situated. The clients offer the interface to the user and use the ULL to send their data to the server or to other clients. The server consists of the Login Server and other servers, like a chat server or VoIP server. If the clients do not use the server to communicate with each other, they use a P2P connection. Login is always done with the use of a server. For the most VoIP systems, buddy search and messaging is done with the use of a server and Calling is done with a P2P connection.

3.1.3 Protocol

In this section we present the generic protocol flow in sequence diagrams. These generic protocol flows are based on the protocol flows of the five VoIP systems as discussed in this chapter. In the 'protocol' sections of the VoIP systems, we show a short overview of the protocol(s) used by the VoIP system, the sequence diagrams are presented in Appendix B.

Login
Figure 3.3 shows the generic login sequence diagram. First a secure connection has to be set up, before login to the server is possible. Afterwards, it is possible to login to other servers.

Buddy search
Figure 3.4 shows the generic buddy search sequence diagram. First the client has to request for a buddy search. For some VoIP systems, a list of possible buddies is returned. This list contains the usernames that almost equal the search. In that case, one buddy has to be selected, in both cases Client B receives an addbuddy request. If Client B accepts, the selected buddy is added to the contact list of Client A.

Messaging
Figure 3.5 shows the generic messaging sequence diagram. First the other client is announced

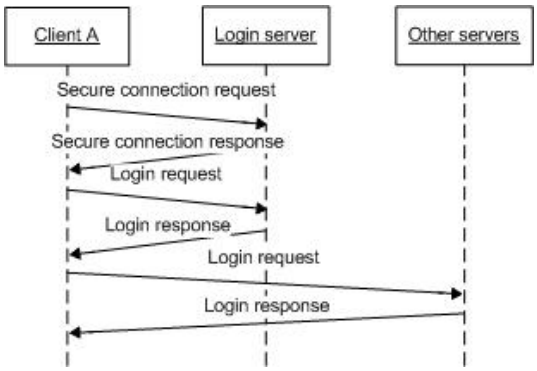


Figure 3.3: Login sequence diagram

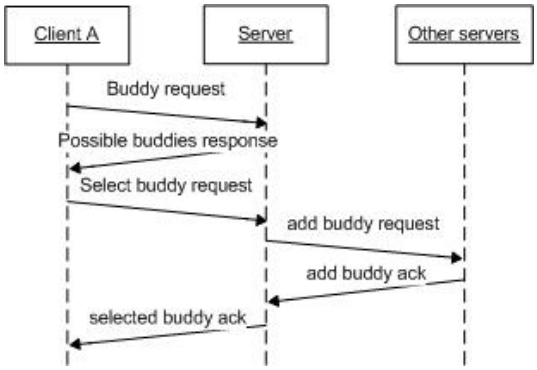


Figure 3.4: Buddy search sequence diagram

the user is typing. This is not a functionality offered by every VoIP system. Afterwards the message is sent and an acknowledgement is received in return.

Call

Figure 3.6 shows the generic call sequence diagram. First a request for a call is sent to Client B. If Client B accepts, the actual conversation takes place. Afterward both Client A and Client B are allowed to close the call, in this example Client A closes the call and Client B acknowledges it.

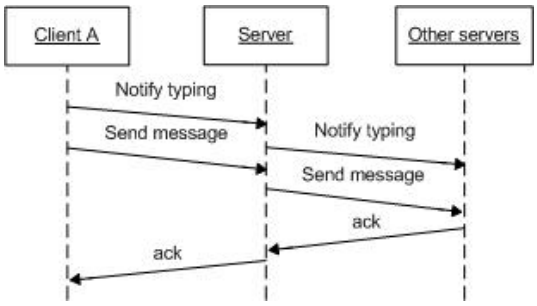


Figure 3.5: Messaging sequence diagram

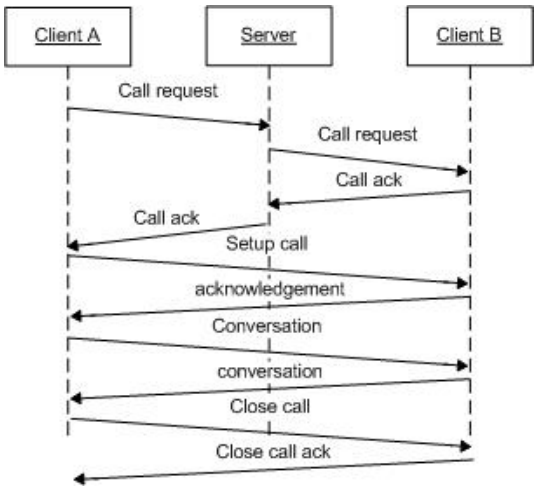


Figure 3.6: Call sequence diagram

3.2 Windows Live Messenger (MSN)

Windows Live messenger [32] (formerly known as MSN messenger) has been released by Microsoft on August 24, 1995, to coincide with the release of Windows 95. Windows messenger is a proprietary VoIP system installed on Windows XP and available for among others Windows 2000. This latter application, Windows Messenger should not be confused with the MSN messenger or Windows Live Messenger, although they have similar functionalities. MSN messenger has changed the name in 2006 to Windows Live Messenger. The word "MSN" is the word for MSN Messenger and Windows Live Messenger in Internet slang [30]. In this report the word MSN will be used when Windows Live messenger or the older version MSN messenger is mentioned. Windows messenger has not been studied.

3.2.1 Features

MSN offers the main services and furthermore offline text messages for Windows Live Messenger, video communication, Multi-user chats, SMS, avatars and other personalization tools. It is also possible to make a call to a PSTN phone.

3.2.2 Entities

The architecture of MSN consists of servers and several clients and is thus a client-server architecture. Multiple servers are used and a peer-to-peer connection between two clients is possible.

MSN uses the following servers:

- Dispatch Server (DS)
- Notification Server (NS)
- Switchboard Server SS

The DS tracks locations for the notification servers and communicates the IP address of these servers to clients. The NS manages the user presence of the users. Clients must maintain connection to the NS at all times. The NS provides information on client locations for routing purposes and provides connections to the SSs. It provides presence information notifying other users if whether or not a particular user is connected. The SS provides messaging and file transfer functionality between two clients.

The .NET Passport Login Server is also part of the authentication process. In this report it can be seen as a part of the DS, to limit the amount of servers in the architecture and the sequence diagrams. The .NET Passport Login Server provides the client attempting to sign in with a ticket to complete the authentication process with a NS. Also the Nexus Server is a part of the DS in this report. The Nexus Server is another part of the authentication process and provides a client with a URL (Uniform Resource Locator) of the .NET Passport Login Server and information necessary to authenticate.

The DS can be accessed via the domain name messenger.hotmail.com. The function of the DS is to direct the user to an appropriate NS (based on the supplied .NET passport). If

the messenger.hotmail.com domain name could not be reached (i.e. if the firewall blocks the port), the domain name gateway.messenger.hotmail.com is available for HTTP connections. The MSN client will often store the address of the appropriate NS once located, to make future connections go easier.

The NS manages user presence on the network. Logging in to the NS announces the availability of the user. The NS stores and synchronizes user details, provides notifications of new Hotmail messages and checks the version of the MSN client being used. The NS does not handle IM, the NS will set up a connection to the SS for IM.

The SS establishes and manages chat sessions between users on the network. The SS forwards IM messages to the recipients. Users do not know each others IP address. The SS facilitates file transfers, but are carried out peer-to-peer using the MSN File Transfer Protocol (MSNFTP). After both users agree to send the file, SS provides IP addresses.

While connected to the SS, the user is still connected to the NS. If the NS connection is broken, the SS notices the offline status and will not allow the connection from the user to the SS anymore.

MSN uses Microsoft's .NET Passport for creating and maintaining the user accounts. This account can be created at the Hotmail or .NET websites, not within the MSN domain. Once an account is created, it is possible to sign in to the MSN network directly.

3.2.3 Protocol

MSN uses the Mobile Status Notification Protocol (MSNP) for the login and the MSN Messenger Service (MSNMS) for the messaging. Furthermore, SIP is used for the call. The sequence diagrams of the main services are presented in Appendix B.3.

3.3 Google Talk

Google Talk (GTalk) [16] is an application offered by Google for VoIP and IM and is available since August 2005. Google's mission is to make the world's information universally accessible and useful. GTalk uses the open protocol Extensible Messaging and Presence Protocol (XMPP) and Jabber [23] for the IM part and presence events. To make VoIP, video and other peer-to-peer multimedia sessions possible, Google released libjingle [88] in December 2005. Libjingle is a library of the code that Google uses for peer-to-peer communication, and was made available under a BSD license. This license is like the GNU GPL license for free software, but with more restrictions.

GTalk does not encrypt the Jabber stream; it uses an undocumented nonstandard way of authenticating to the service, retrieving a token from a secure web server. Other clients than Google's own are required to secure their streams with Transport Layer Security (TLS) before sending the password, causing them to stay encrypted throughout the whole session. GTalk claims to fully support encryption of chats and calls before their next official release. [9] [12] [11]

3.3.1 Features

GTalk does not support many features (yet). It supports the main services as presented in Section 3.1.1. Furthermore if a buddy is offline, it is possible to leave a voicemail. This voicemail can be at most 10 minutes long and it is delivered to the buddies Gmail mailbox as an attached MP3 file.

Currently it is only possible to make PC-to-PC calls with GTalk. GTalk claims however that it is adding DTMF support to Jingle, to make PC-to-PSTN calls possible in the future. [12]

3.3.2 Entities

Google Talk uses Jabber, and thus this section discusses the Jabber architecture.

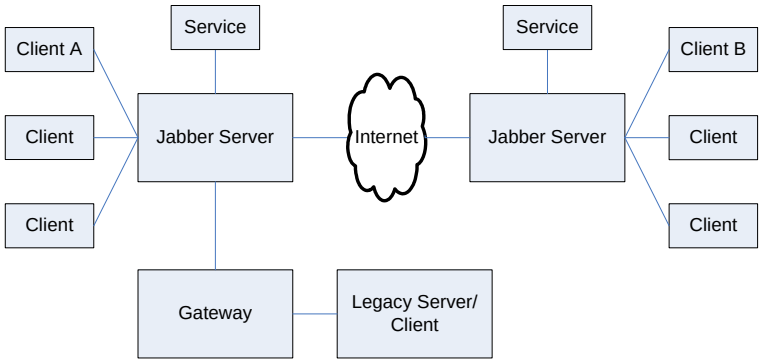


Figure 3.7: detailed GTalk architecture

A Jabber network consists of two kind of entities: clients and servers. Every client is connected to a server and servers may be connected to one another. Like said before, these connections are established using the XMPP communication protocol. Entities should identify itself on a network by a (unique) Jabber Identifier (JID). A JID looks like "username@servername", just like an email address. Additionally it is possible to add a resource to the JID. This part is recommended but optional and the whole JID will look at that case like "username@servername/resource". [25]

Figure 3.7 shows the Jabber architecture. The Jabber server is the central part and is a complex server with many jobs like delivering messages, storing contact lists, managing gateway services and many more. The Jabber architecture is a so called client server architecture [23]. It uses several servers and a peer-to-peer connection could be possible. GTalk also offers a Gateway to Legacy Server / Client to provide a gateway to other applications. This gateway, also transport called, is an example of a component which extends the basic functionalities. There exist bridges to other VoIP systems, like ICQ, Yahoo and MSN [26]

The Jabber clients with their JID belong to a specific server. This means that if a server goes down, it is impossible for the users registered to that server to use the Jabber service, which means it is not possible to use the GTalk client.

3.3.3 Protocol

GTalk uses the Extensible Messaging and Presence Protocol (XMPP) for login and messaging and Jingle for the call. Appendix B.4 shows the sequence diagrams for GTalk.

3.4 Yahoo Messenger

Yahoo! Inc is an internet service company, which operates an Internet portal and provides a full range of products and services including a search engine, the Yahoo! Directory and Yahoo! Mail. It also provides the Yahoo! Messenger [61]. The Yahoo! Messenger will be named short as Yahoo in this report. [92]

The acronym for Yahoo became: "Yet Another Hierarchical Officious Oracle". The messenger is provided free of charge. To login to the messenger and other Yahoo! Services, a Yahoo! ID is needed. [59] [60]

Interoperability between Yahoo and MSN was launched July 12, 2006. This allows for Yahoo and MSN users to talk to each other without the need to create an account on the other service. Not all features are supported though; voice calls cannot be made.

3.4.1 Features

Yahoo offers all main services as presented in Section 3.1.1. Starting with version 8.0, it is possible to create plug-ins to add more functionalities. Yahoo offers e-mail, voicemail, games, chat rooms, conferencing and webcam support.

On Microsoft Windows operation systems Yahoo offers some unique features such as IMVironments (customizing the look of Instant Message window), address-book integration and Custom Status Message. Furthermore it offers offline-messaging, BUZZing, music-stats and avatars.

3.4.2 Entities

The architecture of the Yahoo is a client-server architecture. Yahoo uses the following servers:

- Login server
- IM server
- SIP server
- STUN server

STUN (Simple Traversal of UDP (User Datagram Protocol) through NATs (Network Address Translators)) is a network protocol allowing a client behind a NAT (or multiple NATs) to find out its public address, the type of NAT it is behind and the internet side port associated by the NAT with a particular local port. This information is used to set up UDP communication between two hosts that are both behind NAT routers. The protocol is defined in RFC 3489. [48]

3.4.3 Protocol

YMSG is the Yahoo! Messenger Protocol, which is used for the login and messaging. Furthermore, SIP is used for the call. Appendix B.5 shows the sequence diagrams for the protocol flows.

3.5 ICQ

ICQ [21] was developed in 1996 by Mirabilis, the creators of the first fully functional Internet-wide instant messenger comprising presence, a contact list and rapid messaging. AOL bought Mirabilis on June 8th, 1998 for over 200 million dollars. On December 19, 2002, AOL Time Warner announced that ICQ had been issued a United States patent for instant messaging (USPTO Patent Number 6,449,344). In June 2004 ICQ celebrated its 300 millionth download from download.com where it remained the most popular program for seven consecutive years.

Before ICQ, people were connected to the internet, but not interconnected. ICQ was the missing link, a technology that made peer-to-peer communication possible. The following year ICQ took the internet by storm. Through viral marketing a chain reaction was created, resulting in one of the largest download rates for a start-up company in the history of the Internet. Since MSN arrived at the market, the popularity of ICQ decreased. [20]

ICQ has always been technologically innovative. Not only was ICQ one of the first Internet wide instant messaging services, many features that make up the core of today's IM services were first introduced by ICQ. [20]

The name ICQ should be pronounced as "I seek you", because the founders of the system wanted to make clear that it is possible to use ICQ to find somebody and talk to that person in real-time.

3.5.1 Features

ICQ features offline text messages support, video communication, Multi-user chats, SMS, greeting card, multiplayer games, searchable user directory, skins, emoticons, avatars and other personalization tools and POP3 email support. Furthermore it supports the main services as presented in Section 3.1.1.

ICQ users are identified by a Universal Internet Number (UIN), also called Unified Identification Number (UIN). Every user of ICQ gets his own number after registration. There exist already over 300.000.000 numbers. Buddies can be found by UIN, by name, e-mail address or some other personal details a user makes available. [18]

ICQ and AIM users are since 2000 able to add each other to their contact list without any extra server or gateway.

3.5.2 Entities

An ICQ network consists of clients and servers. Each client has to login at the server before it can communicate with buddies. In the early years of ICQ, before AOL bought it, it was possible to have a peer-to-peer connection between two clients. Since AOL bought ICQ, it

introduces the protocol OSCAR. With the OSCAR protocol, no peer-to-peer connection is set up.

ICQ uses the following servers:

- Login
- BOS (Basic Oscar Services)
- File

3.5.3 Protocol

Since AOL bought the ICQ company, ICQ uses the OSCAR protocol for login, buddy search, messaging and call. OSCAR does not offer peer-to-peer connections, which means even the call is done with the use of a server in between the two clients. Appendix B.6 shows the sequence diagrams of the protocol flow.

3.6 Skype

The founders of Skype [45] wanted something that would let the world talk for free. One of the names they came up with was "Sky peer-to-peer", which got soon shortened to "Skyper". But one of the domain names associated with "skyper" was already taken, so they dropped the "r" and made it "Skype"

Skype is a popular and free VoIP system from the developers of KaZaA, and thus uses a similar network topology as KaZaA's file sharing system. This means that Skype is a peer-to-peer communication network. It is possible to download the Skype client from the website [46] for free and make phone calls to other Skype users for free. Some additional features the user has to pay for are SkypeOut and SkypeIn; Calling a PSTN telephone number at low rates and use a subscription to receive calls from a PSTN telephone number respectively.

Skype uses a proprietary protocol and has no detailed information available. This report contains an explanation of how the Skype network works found in literature and gives analyses of the protocol used by Skype. However, keep in mind that some of the information that is given are researched guesses by the authors of some papers that analysed the Skype system and protocol and by the author of this report. Skype itself provides a Skype API, to make it possible to build plug-ins for the Skype system.

3.6.1 Features

Skype offers the main services as presented in Section 3.1.1. Furthermore it is possible to call a PSTN line (SkypeOut), or receive a call from a PSTN line (SkypeIn). Latter two have to be paid for, but this is often less then the standard PSTN companies ask for. Some phone numbers (in some countries) are even for free. Since February 2007, Skype launched its Skype Pro in Europe. For just two Euros a month users are able to receive calls from a PSTN phone (SkypeIn) and to call a PSTN phone (SkypeOut) for free.

Skype Voicemail has experienced numerous problems over the past year and users complain that many voicemail calls are never received. Additionally, the SkypeIn service occasionally never records certain incoming calls on the systems history page. These current problems have not been completely resolved. Skype supports also group text chat with an interface similar to IRC with 100 People.

On Windows XP, Skype 2.0 (and above) supports videoconferencing. For Mac, Skype Beta version 1.5.0.4 was the first version to support Video Chat, making Skype one of the few cross-platform video conferencing solutions between Windows and Mac. Skype only supports one-to-one video chat but may support multiple point video conferencing in the future.

Skypecasts are live, moderated conversations allowing groups of up to 100 people to converse, moderated by the 'host' who is able to mute, eject or pass the virtual microphone to participants when they wish to speak. Skypecasts do not support chat windows to share text information (such as URLs) with participants.

One of the few new features in Skype 2.5.0.72 beta is the ability to send SMS messages to mobile phone numbers.

3.6.2 Entities

The basic Skype architecture is composed of the following entities [66]:

- Host nodes: Host nodes are Skype systems, which enable phone calls and sending messages
- Super nodes: Any node in the overlay network with a public IP address, high amount of CPU power, memory and bandwidth has possibility to become a super node.
- Login server: Though pure P2P networks do not have any centralized servers, Skype still uses them for user authentication.

Any node in a network can become a super node. This means that every computer where the Skype client runs can be used as a super node and will use some bandwidth and CPU power. This information is available in the Skype End User License Agreement, which the user should agree on before installing the Skype application.

3.6.3 Protocol

Skype uses the proprietary protocol Skype for the login, buddy search, messaging and call. It uses a peer-to-peer connection. Appendix B.7 shows the sequence diagrams for Skype.

3.7 Conclusion

This section provides summarizing tables. Table 3.1 shows the differences and similarities of the VoIP systems. The "Buddy ack" entry means if the buddy is asked for acceptance to add the buddy to the contact list of the originating user. Table 3.2 [33] [7] [8] [95] shows a summary of the VoIP systems.

	BIT messages	potential buddies	Messaging	Call	Buddy ack
MSN	yes	no	not P2P	P2P	yes
GTalk	no	no	not P2P	P2P	no
Yahoo	yes	no	P2P	P2P	yes
ICQ	yes	yes/no	not P2P	not P2P	yes
Skype	no	yes	P2P	P2P	no

Table 3.1: Differences and similarities of VoIP systems

For all VoIP systems counts that the call have to be accepted before the actual call can take place. With the buddy search, GTalk and Skype are different to the rest; they do not have to accept the buddy search request before the client can add the buddy to the contact list.

As said in Chapter 2, some protocols are proprietary protocols. Skype is an example of a proprietary protocol. Because of these proprietary protocols, interoperability is not that simple to arrange. In the following chapters, we search for a solution to create interoperability between VoIP systems.

	MSN	GTalk	Yahoo	Skype	ICQ
Creator / Owner	Microsoft	Google Inc.	Yahoo!	Niklas Zennstrom, Janus Friis (eBay)	Mirabilis
First public release	July 22, 1999	August 23, 2005	June 21, 1999	2003	November 1996
Latest stable version	Live Messenger	1.0.0.100	8.1.0.195 (Windows), 2.5.3 (Mac)	2.5.0.151 (Windows), 1.3.0.53 (Linux),	5.1
Software license	Proprietary	Proprietary	Proprietary	Proprietary	Proprietary
IM	Yes	Yes	Yes	Yes	Yes
Encryption	No	No	No	Yes	No
File Transfer	Yes	Yes	Yes	Yes	Yes
Graphical smileys	Yes	Partial	Yes	Yes	Yes
Unicode (UTF-8)	Yes	Yes	Yes	Partial	Yes
Plug-in system	No	No (but gateway)	Yes	Yes (API)	No
Third party add-ons	Yes	Yes	No	Yes	Yes
Message Logging	Yes	Yes	Yes	Yes	Yes
Webcam support	Yes	No	Yes	Yes	Yes
PCtoPc calls	Yes	Yes	Yes	Yes	Yes
VoIPout	Yes (LiveCall)	No	Yes (Phone-Out)	Yes	Yes (ICQphone)
VoIPin	No	No	Yes (Phone-In)	Yes	No
Protocol	MSNP (login), MSNMS (messaging)	XMPP	YMSG	Skype	OSCAR
Call protocol	SIP	Jingle	SIP	Skype	OSCAR
Compatible with	Yahoo	None	MSN	None	AOL
Website	get.live.com	www.google.com/talk	messenger.yahoo.com	www.skype.com	www.icq.com

Table 3.2: Summary of VoIP systems

Chapter 4

VoIP system services

Chapter 3 described five commonly used VoIP systems. This chapter provides the minimum and optional services offered by those VoIP systems.

4.1 Services

Figure 4.1 shows the structure of the entities of a VoIP system. The figure shows two clients; a user client and a buddy client. Furthermore it has a server in between those clients. Figure 4.2 shows the services as presented to the user. The VoIP system will be handled as a blackbox. These services are described in this chapter.

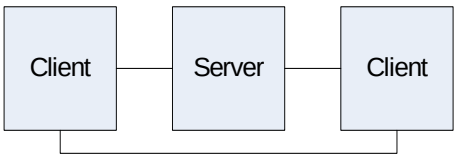


Figure 4.1: VoIP system entities

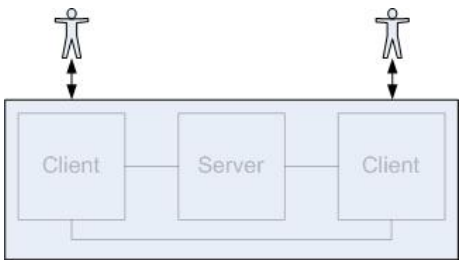


Figure 4.2: VoIP system entities

The VoIP systems support at least the minimum set of the four features as presented in Chapter 3.1.1. These features are Login, Buddy search, Messaging and Call and are described shortly:

- For most VoIP systems applies that the **Login** is done with the use of one Client and

one Server; the Client performs a request to login, and the Server confirms the login (or not).

- For the **buddy search**, a request to search for a buddy is sent by the client to the server. If the server finds the buddy, the buddy receives a request for acceptance. If the buddy accepts, the user receives a new contact list with the new user included from the server.

- When the user sends a **message** to its buddy, the client sends the message to the server, which forwards the message to the client of the buddy.

- The **call** is divided into call setup, tear down and the actual call. The setup is done by sending a request to the server, which asks the client of the buddy for acceptance. If the buddy accepts, IP addresses are exchanged. Now the actual call can take place by a peer-to-peer connection (the line between the two clients in Figure 4.1.

Notice that the features as performed in the preceding section can be different from VoIP system to VoIP system. The description of the preceding paragraphs are the ones mostly used.

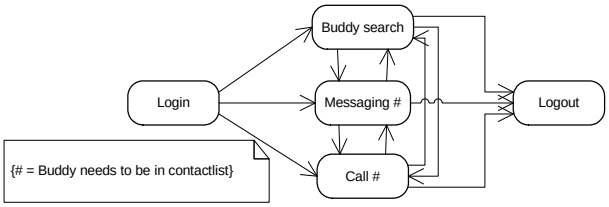


Figure 4.3: Login, Buddy search, Messaging and Call

Figure 4.3 shows the order of actions that can happen. A user always has to login first, before it can do something else. After a correct login, it is possible to search for a buddy, to send a message, or setup a call and have a conversation. From the buddy search, messaging, and call state it is always possible to switch between each other. From each of these states, it is possible to logout. Contacting a buddy (sending a message, a file or having a call) is only possible if the buddy is in the contact list.

In this chapter, the four features are called *service elements*. The interactions between the clients and servers and the user and VoIP system are called *service primitives*. Furthermore, the service primitives including their parameters are given. These parameters will be shown between parentheses behind the service primitives in Section 4.1 and Section 4.1.2.

4.1.1 Minimum services

This section provides the minimum services offered by each VoIP system. The interactions are discussed per service element.

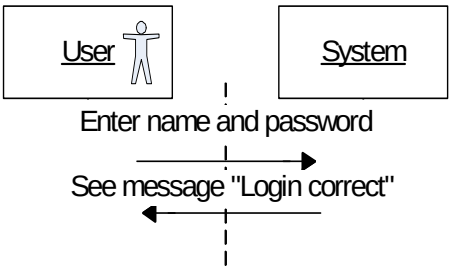


Figure 4.4: Login

Login

The service primitives and parameters of the service element Login are:

- Enter name and password (username, password)
- See message "Login correct" (login correct)
- See message "Login incorrect" (login incorrect)

Figure 4.4 shows the interactions between the User and the VoIP system for a correct login. When the user enters his user name and password, the client sends a login request including this user name and password system (to the server). The user receives a positive or negative confirm. If the confirm is positive, the user is logged in. Otherwise the login fails and the user can try again.

Buddy search

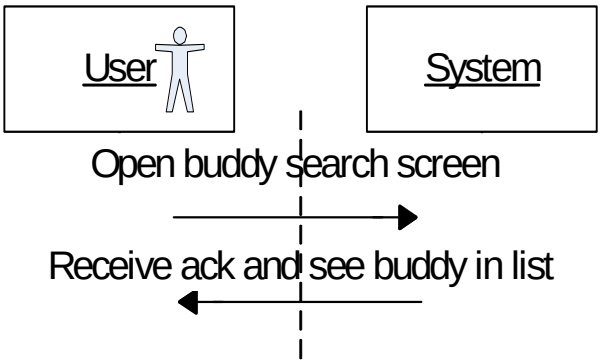


Figure 4.5: Buddy search

To extend the contact list, a buddy can be searched. The service primitives and parameters of the service element buddy search are:

- Open buddy search screen (open)

- Enter buddy name (buddy name)
- Receive ack and see buddy in list (contact list)

Figure 4.5 shows the interactions between the User and the VoIP system for a buddy search. To extend a contact list, the user should first open the screen of the client to search for a buddy. In the text field, the user enters the name of desired buddy. If the buddy is found, the contact list will be extended.

Messaging

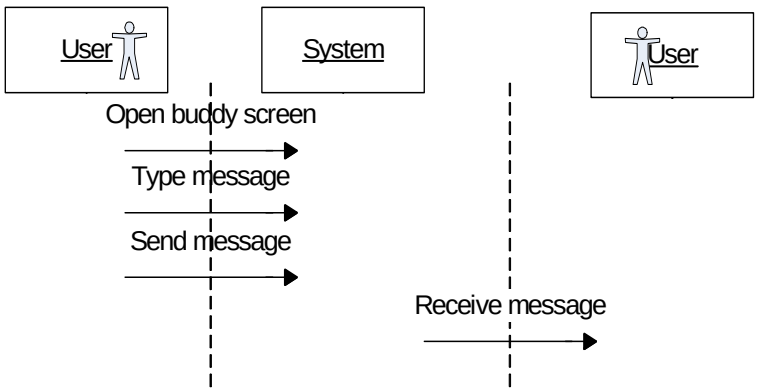


Figure 4.6: Messaging

The service primitives and parameters of the service element messaging are:

- Open buddy screen (open)
- Type message (message)
- Send message (message)
- Receive message (message, buddy name)

Figure 4.6 shows the interactions between the user and the VoIP system when sending a text message. To send a message, the user opens the screen of the client used for sending a text message to the buddy. Now he types a message. If the user sends the message (e.g. press the "send" button), the buddy receives the actual text message.

Call

The service primitives and parameters of the service element call are:

- Start call (setup)
- Receive request for call (request for call, buddy name)

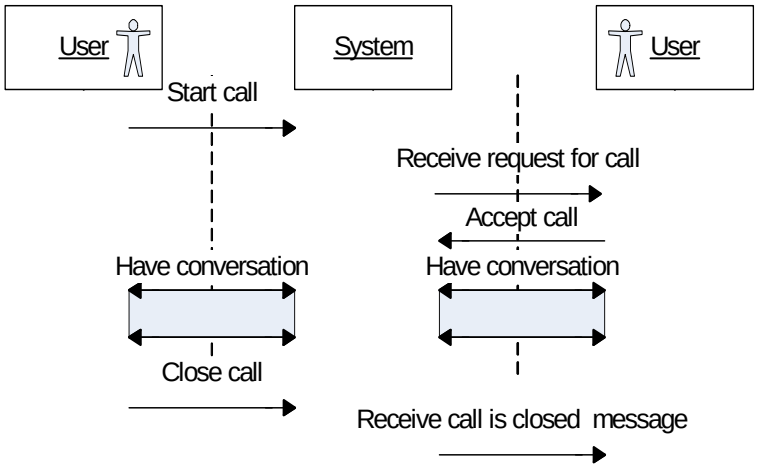


Figure 4.7: Call

- Accept call (accept)
- Decline call (decline)
- Receive decline message (decline message, buddy name)
- Have conversation (audio, buddy name)
- Close call (close)
- Receive call is closed message (closed by buddy, buddy name)

Figure 4.7 shows the interactions between the users and the VoIP system for a call. If the user starts a call (mostly by pressing the 'start call' button), the buddy receives a request for the call. If the buddy declines the call, this is displayed by the client of the originating user. If the buddy accepts, the client is able to let the user have a conversation with its buddy. If the user closes the call (mostly by pressing the 'end call' button), the buddy sees that the "conversation is closed" displayed at its client.

4.1.2 Optional services

This section provides the optional services. To show the correct sequence of the interactions, also the minimum services are shown. For the login and call, no optional services exist, so these two service elements will not be repeated. The optional services are marked in the sequence diagrams and in the service primitives list with a star (*).

Buddy search

The service primitives and parameters of the service element buddy search are:

- Open buddy search screen (open)

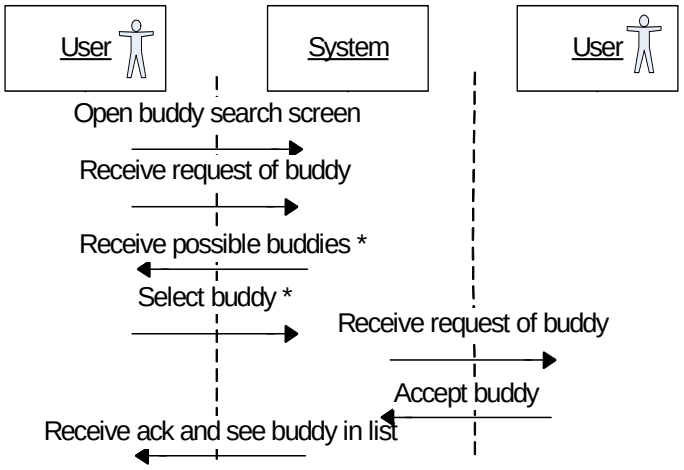


Figure 4.8: Optional services for buddy search

- Enter buddy name (buddy name)
- Receive possible buddies (possible buddies) *
- Select buddy (buddy name) *
- Receive request of buddy (request, buddy name) *
- Accept buddy (accept, buddy name) *
- Ignore buddy (decline, buddy name) *
- Receive ack and see buddy in list (contact list)

Figure 4.8 shows the interactions between the User and the VoIP system for a correct buddy search. To extend a contact list, the user should first open the screen of the client to search for a buddy. In the text field, the user enters the name of desired buddy. Some VoIP systems first send a list with possible buddies, and the buddy needs to select the desired one. Other VoIP systems skip the possible buddy part. Both continue with an optional request to the buddy to accept the originating user. If the buddy accepts, the originating user sees the extended contact list, with the buddy on it. If the buddy ignores or declines the request, nothing happens; The contact list is not extended with this buddy.

Messaging

The service primitives and parameters of the service element messaging are:

- Open buddy screen (open)
- Type message (message)
- Receive "Buddy is typing" message (BIT message, buddy name) *

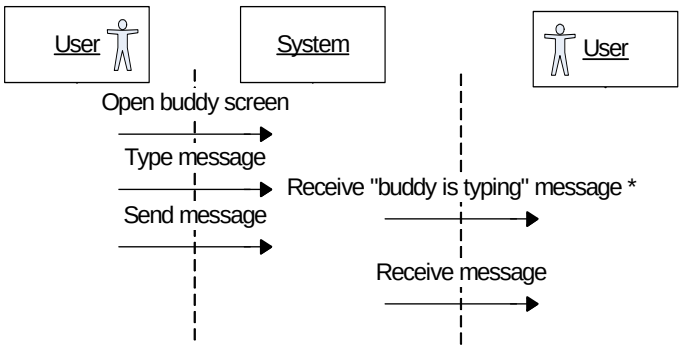


Figure 4.9: Optional services for messaging

- Send message (message)
- Receive message (message, buddy name)

Figure 4.9 shows the interactions between the User and the VoIP system for sending a text message. To send a message, the user has to open the screen of the client used for sending a text message to the buddy. Now he types a message. While typing, the buddy receives a message that the originating user is typing. If the originating user sends the message (e.g. press the "send" button), the buddy receives the actual text message.

4.2 Differences

This section provides an overview of the differences in offered services by the VoIP systems. The differences are described per service element. These differences are important to keep in mind during the design. The solution should be able to handle the differences and respond to it in a proper way.

4.2.1 Login

The login procedure is not completely equal for all VoIP systems, in some cases the User has to enter a name (GTalk, Yahoo, Skype), some cases an email address (MSN) and in some cases a number or an email address (ICQ). Some VoIP systems provide a cookie to allow login to a second (or even third) server.

4.2.2 Buddy search

The VoIP systems Skype and ICQ (in case the people search will be used) offer the possibility to search for keys, like a (part of a) name, provide a list of potential buddies and the User can select one of them. Yahoo, MSN, GTalk and ICQ (in case the search will be on number of email address) do not offer this possibility. The entered name (name, number, email address) should match exactly an existing user and will be added to the contact list after accepting

BIT message	MSN, ICQ, Yahoo
No BIT message	GTalk, Skype
Potential buddies	Skype, ICQ (partly)
No potential buddies	MSN, GTalk, Yahoo, ICQ (partly)
Buddy ack	MSN, Yahoo, ICQ
No buddy ack	GTalk, Skype

Table 4.1: BIT messages, potential buddies and buddy acceptance

the buddy. MSN, Yahoo and ICQ require the user to accept the request to be added to the contact list, before the buddy can be added to the contact list. GTalk and Skype do not offer this service.

Table 4.1 summarizes the VoIP systems sending BIT messages and requires the acceptance of the buddy before extending the contact list.

4.2.3 Messaging

Only for the VoIP systems MSN, Yahoo and ICQ a message like "Buddy is Typing" will be displayed at the client and thus has to be send and received. GTalk and Skype do not offer this functionality.

table 4.1 also summarizes the VoIP systems that offer potential buddies.

4.2.4 Call

For the call, all services offered by MSN, GTalk, Yahoo, ICQ and Skype are equal.

Chapter 5

Related work

This chapter describes systems that offer interoperability. The need of this chapter is to see the possibilities that are on the market right now for the interoperability of IM and VoIP systems and protocols. Based on these IM and VoIP systems (and the lack of these systems), the following steps to design an own interoperable system can be made, which will be done in the next chapters.

For now, three kind of interoperability applications are presented. The first sections discuss PSGw and Uplink. These applications work as a router in between Skype and SIP, to translate Skype into SIP and SIP into Skype.

Furthermore, GTalk-to-VoIP will be discussed. This application makes it possible to translate GTalk to SIP. Because MSN and Yahoo also use the protocol SIP, these three VoIP systems are interoperable.

Third, systems to make several IM and VoIP systems interoperable are discussed. The interoperable IM systems focus on the interoperability of IM systems and have now also started to try to integrate VoIP. The interoperable VoIP systems focus on the interoperability of VoIP systems. Three commonly used interoperable IM systems are [22]:

- Gaim
- Miranda IM
- Trillian

These systems are not able to support all functionalities of the messengers included. Furthermore, it takes a while before new offered functionalities by the IM systems are supported.

The essence of the three interoperable IM systems are mostly the same. Different accounts have to be filled in in these application and all buddies of these different accounts will be put in one contact list. Trillian will be discussed in Section 5.4. The other applications are not discussed, because the essence of the applications are the same.

Two commonly used interoperable VoIP systems are:

- Wengophone
- Gizmo

Also these two interoperable VoIP systems are in essence the same, several accounts have to be filled in and contact lists will be combined. Only Gizmo will be discussed in Section 5.5.

5.1 PSGw

Personal Skype to H.323/SIP gateway (PSGw) [38] is a system that allows connecting the Skype network with H.323 and SIP networks. PSGw works as a router that should be placed between a Skype and a H.323 or SIP network and route calls according to user-defined rules. PSGw supports only a single concurrent connection between a Skype and a SIP/H.323 network.

PSGw runs on top of Skype, which means that a Skype client should run at the same computer as the PSGw application. At the PSGw client several user data should be filled in, like the SIP user client the call should be forwarded to. An incoming Skype call will be handled by PSGw instead of Skype and PSGw will forward it to the SIP client. This SIP client can run on a different computer.

It is also possible to have a SIP to Skype call. The PSGw system forwards and converts the SIP call to the Skype system.

An upcoming application by PSGw is the first "Multiline Enterprise Software Skype Exchange" (MESSiX). MESSiX allows handling multiple Skype connections at the same computer.

To have a call using SIP, a SIP phone is needed. Several (free) SIP phones can be found on the Internet. Furthermore, a SIP address to forward the call to is needed. A free SIP address can be received from Ekiga [54].

5.2 Uplink

The idea is the same as for PSGw, although this one needs more software to be installed to let it work exactly the way you want. First of all, the client Uplink is needed, which is the Skype-to-SIP converter. Second, the Virtual PBX Axon is needed, and third the SIP softphone Express Talk. The Axon Virtual BPX will act as a SIP provider and generates a SIP account. When a buddy calls Skype, Uplink will take the call over (Uplink is added as a plug-in to Skype) and forwards it to the SIP softphone Express Talk. As with PSGw, Skype has to run at the PC to let Uplink work with it.

Uplink [50] connects SIP protocol calls to the proprietary Skype phone network, but it does not support H.323. It works in both directions.

Uplink connects both incoming and outgoing calls to the Skype network, it can be used to make SkypeOut calls from a SIP device or SIP PBX and can receive SkypeIn calls and direct them to your SIP extension. It fully complies with the RFC3261 for SIP signaling and offers quick and easy operation.

5.3 GTalk-to-VoIP

Ruslan Zalata has implemented a GTalk to VoIP gateway. He has made an application to convert GTalks JingleAudio to H.323 and SIP (and the other way around). The core of their implementation is a VoIP soft-switch. It consists of a number of VoIP signal processing stacks, like H.323, SIP (including Yahoo and MSN its "dialects") and Jingle Audio. It also includes media codecs, transcoders and conferencing rooms (the last are just a subtype of transcoders).

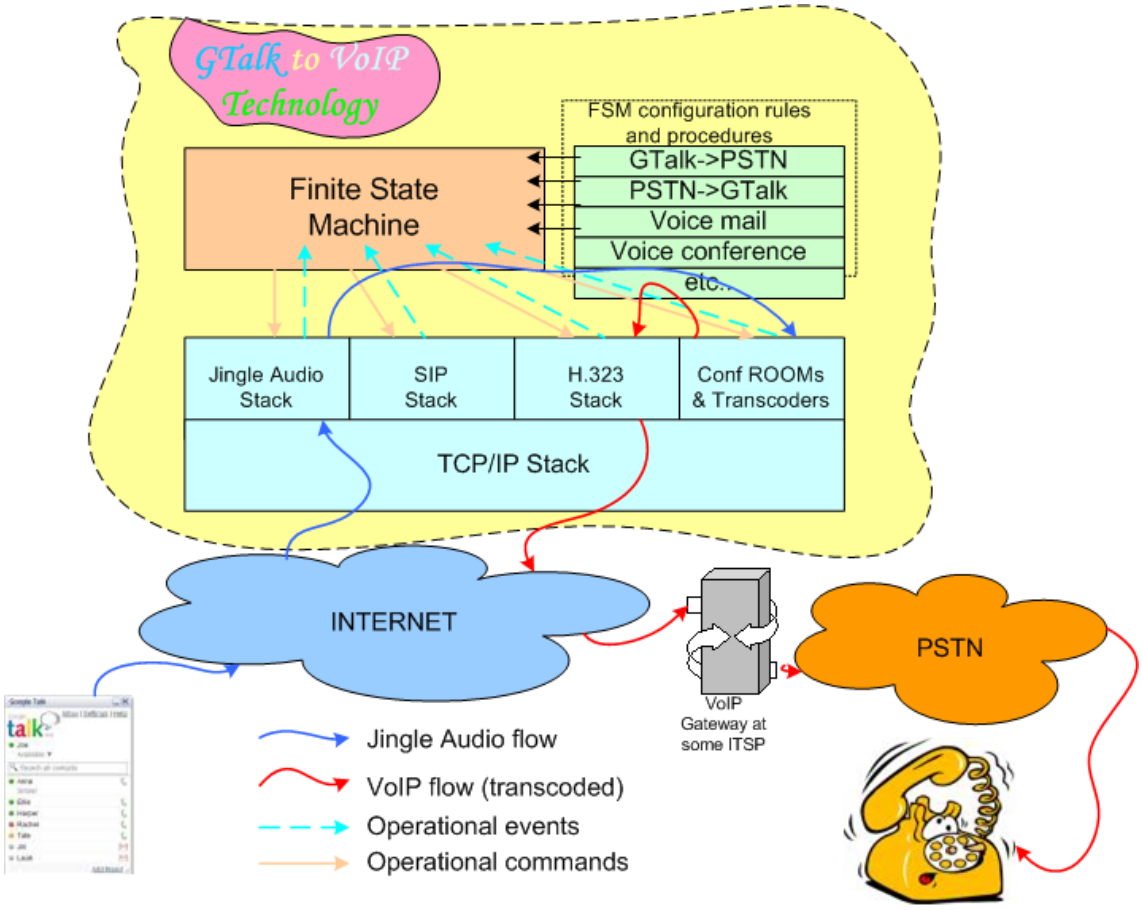


Figure 5.1: GTalk to VoIP technology [15]

GTalk-to-VoIP, also written as GTalk2VoIP, makes it possible to let the protocols H.323 and SIP, interact with the protocols of the VoIP systems MSN, GTalk and Yahoo. They offer the possibility to add MSN and Yahoo contacts to the Google Talk contact list, or GTalk and Yahoo to the MSN contact list. Users have to be added to GTalk named someuser%hotmail.com@msn.gtalk2voip.com. Voice calls can be made to these added buddies by pushing the call button.

Text messages sent to such contacts will not reach MSN users, though they are working on this feature and will be available soon

To start using the service using for instance an MSN client, a user has to add a new buddy, the service@gtalk2voip.com buddy. This buddy will handle commands. To know which commands, this special buddy will send a command list, as displayed in the following box:

```
service008@gtalk2voip.com says:
Welcome to GTalk2VoIP - Free Voice Gateway for Google Talk, MSN and Yahoo!

service008@gtalk2voip.com says:
Type 'HELP' for more information

service008@gtalk2voip.com says:
You entered:  help

service008@gtalk2voip.com says:
.
====== AVAILABLE COMMANDS ======
HELP - Display this message.
MYPAGE - Display URL to your personal account page.
WEBCALL - Display URL to your personal Web Call.
MSG to text - Send off-line message text to user to .
IM to text - Send Instant Message text to user to .  Example:  IM msn:billgates@hotmail.com Hello
Billy!
VMLIST - Display content of your voice mail box.
VMPLAY id - Play voice mail message number id .
VMDEL id - Delete voice mail message number id
VMSEND user - Send voice mail to other gtalk user .
CONF - Create conference room for you.  Room cookie will be sent back.
JOIN cookie - Join someone's else conference room using cookie .


service008@gtalk2voip.com says:

COST phone - Display amount of credits charged for 1min unit while calling to phone .  Example,
to figure price for calling american 800 services type:  COST 1-800
CREDITS - Go buy more credits.
CALL phone [via] - Make an outgoing voice call to telephone number phone which is represented in
E.164 format.  Example, to call Google type:  CALL
1-650-253-0000 .  You can also setup a registered service provider name in optional via parameter to
place call throu.
CALL sipuser@sipprovider.com - Make an outgoing voice call to SIP phone sipuser@sipprovider.com .
Example:  CALL pbx@stalker.com .
DTMFTONES : Type in any string of digits or Asterisk or # sign while talking to send DTMF tones.
SMS phonenumber text - Send text an SMS message to phone .  Example:  SMS +1-234-5678901 Hello, just
testing
============== END =============


service008@gtalk2voip.com says:

Please, enter command:

service008@gtalk2voip.com says:
Your personal account page:  http://www.gtalk2voip.com/users/?auth=xxxxxxxxx
```

5.4 Trillian

Trillian [2] is a multiprotocol instant messenger built by Cerulean Studios. It is a fully featured, stand-alone chat client that supports AIM, ICQ, MSN, Yahoo, and IRC. It provides functionalities that are not possible with original network clients, while supporting standard

features such as audio chat, file transfers, group chats, chat rooms, buddy icons, multiple simultaneous connections to the same network, server-side contact importing, typing notification, direct connection, proxy support, encrypted messaging, SMS support, and privacy settings.

Trillian itself has no protocol included; it supports other protocols. Trillian has two versions: a basic and a pro version. The basic version is for free, the pro function has to be paid for. The latter offers more functionalities, like the use of plug-ins to support more protocols (and thus the opportunity to support more VoIP systems).

Trillian connects to multiple instant messaging services without the need of running multiple clients. Users can create multiple connections to the same service, and can also group connections under separate identities to prevent confusion. All contacts are gathered under the same contact list. [49]

There is a plug in to interact with Skype, called SkyLlian, but it needs Skype to be installed in order to be used.

To eliminate duplicates and simplify the structure of the contact list, users of Trillian Pro can 'bundle' multiple contacts of the same person into one entry in the contact list.

5.5 Gizmo Project

The Gizmo Project was founded by Michael Robertson. The Gizmo Project uses open standards for call management, SIP and Jabber. It uses several proprietary codecs, like Skype. The Gizmo Project client is proprietary/closed source software. It is run by the company SIPphone. It offers free calls to Gizmo Project, Yahoo, GTalk, or MSN users and paid calls to PSTN phones. It uses a peer-to-peer VoIP network.

Since the Gizmo Project is based on SIP, it can interoperate with other SIP-based networks directly, although some data will always be routed via central SIPphone-servers, making it less than ideal in these settings from a privacy and security perspective. This includes the popular PBX applications Asterisk [53] and Ekiga [54], which avoid the phone system and is thus free of charge. Gizmo uses encryption (SRTP) for Gizmo to Gizmo calls.

The text chat function of Gizmo Project utilizes the Jabber protocol. Users using Gizmo can be reached through the Jabber protocol at username@chat.gizmoproject.com

Gizmo also offers Call-in: To call from a phone to Gizmo. However, both Call-in and Call-out are not available in every country.

Chapter 6

Requirements

In the preceding chapters, VoIP has been explained and five commonly used VoIP systems have been discussed. These VoIP systems offer both IM and VoIP functionalities. Furthermore, in Chapter 5, related work has been discussed. Many interoperability systems already exist and provide full integration between systems for IM. Many interoperability systems already try to provide full interoperability between VoIP systems. This latter is still not fully covered; no existing interoperability system offers full interoperability between all VoIP systems.

The following two chapters will provide information needed for the design of an interoperable system for fully interoperability of the five commonly used VoIP systems as presented in Chapter. To create a good design, it is important to take the requirements of different stakeholders into account. This chapter focuses on the requirements to create interoperability, which illustrates the conditions the interoperable system should fulfill.

When creating interoperability, several stakeholders have to be considered. These stakeholders are:

- Users
- Application Providers (APs)
- Interoperability Provider
- Designers and implementers

The users are the people of flesh and blood, which control the application. The APs are the companies creating and maintaining the VoIP systems, an example is Microsoft for MSN. The Interoperability provider is the one which will offer the functionalities necessary for interoperability. For instance a gateway can be used to convert and forward the messages. The designers and implementers are the ones creating the interoperable system.

The following sections will discuss the requirements of each stakeholder. Notice that no surveys on this topic are done. The requirements are just logical requirements considered by the author. Furthermore, the requirements do not have the same weight of importance. In Chapter 9 the most important requirements (and which are ignored) will be discussed. Also Paragraph 7.6, the conclusion of the possible approaches for an interoperable system reverts to these requirements.

6.1 User requirements

The user requirements focus on the needs and expectations of users, which can vary in weight from user to user. The users have the following requirements:

- Use one VoIP client
- Continue using familiar client
- Many features in one VoIP system
- User-friendly
- Users must not notice the use of another server
- Plug-in availability

Furthermore, users want costs as low as possible, and the service quality and the speech quality should be high. It also should be a trustful and secure application. Although the goal to design a system for interoperability can be broad interpreted, the focus will not be on building a new VoIP system, but on a gateway that handles the interconnection to combine existing VoIP systems and their protocols. This means the cost, reliability, service and speech quality requirements only have limited influence to the design. These requirements are part of the existing VoIP systems itself. Those VoIP systems will not be changed, only plug-ins will be used to manage the desired result.

Use one VoIP client: Users want to use just one single client to make a call. They do not want to install additional software, just use the client software of the existing VoIP system. This requirement means that as many as possible VoIP systems should be interconnected. Furthermore, users want to have it as easy as possible, so they just want one account and want to contact everybody in their contact list and extend this contact list with new buddies.

Continue using familiar client: Users do not like to change the client they are currently using and thus the client they are familiar with, because they are used to the current layout and features. Furthermore they like to keep their contact list (buddies); with a new client they need to build there contact list from scratch.

Many features at one VoIP system: A user would like to have a VoIP system with all services (like messaging and calling) included. Some newly VoIP system as described in Chapter 5 do not offer all features (yet).

User-friendly: The application should be user-friendly. It should be easy in use and the actions they have to execute (like login) should be intuitive.

Users must not notice the use of another server: Users just want to communicate with their buddies. They probably do not know they are connected to a server, so do not bother them with changing the settings to connect to a server of the interoperable system (too).

Plug-in availability: The Interoperable communication system requires an interface to the Client, in order to be able to send and receive messages between Client and Gateway.For every VoIP system to be able to communicate with the Gateway, an open or known interface should be available. This could be, for example, specified in an API.

6.2 AP requirements

The APs all want to have the best and most used application. Some companies have chosen for an open protocol or even an open source product. Others (like Skype) have a proprietary protocol. The requirements of the APs are:

- No or minimal changes to their VoIP system
- Reliability
- Performance

No or minimal changes to their VoIP system: Open source VoIP systems can be changed; vendors of closed VoIP system only want to change their VoIP system if it has enough benefit. An application like Skype probably does not want to change. Of course, it is the question if, how and who has to change the open source VoIP systems.

Reliability: If the user uses the client of the AP in combination with a server of the interoperable system, this server should be as reliable as the vendor's server, which means the security level and the speech and quality level should be as high as they are of the APs. Otherwise it could look like a failure of the AP, or it looks like the AP offers less quality.

Performance: If the user uses the client of the AP in combination with a server of the interoperable system, this server should be always available and have a stable performance, because otherwise it could look like a failure of the AP.

6.3 Interoperability Provider requirements

The interoperability provider has the following requirements:

- Scalability
- Performance
- Availability
- Reliability

Scalability: The number of users should not be limited, it should be possible to support extra users and thus extra client instances.

Performance: If the user uses the client of the AP in combination with a server of the interoperable system, this server should be always available and have a stable performance, because otherwise it could look like a failure of the AP.

Availability: A VoIP system like MSN is nowadays installed on most computers, at home, at work and at public places. The new system should be easily available, also at other computers.

Reliability: If the user uses the client of the AP in combination with a server of the interoperable system, this server should be as reliable as the vendor's server, which means the security level and the speech and quality level should be as high as they are of the APs. Otherwise it could look like a failure of the AP, or it looks like the AP offers less quality.

6.4 Designers and implementers requirements

The designers and implementers have the following requirements:

- Universal solution
- Few implementation work
- Easily extendable

Universal solution: The interoperable system should include at least the five most commonly used VoIP systems, or even more. Furthermore it should provide both IM and VoIP.

Few implementation work: Possible approaches for the interoperable system exists. Some approaches need a lot of implementation work, like implementing a gateway in combination with a new client (or the whole VoIP system), other approaches need less implementation work, for instance when only a gateway has to be implemented. The less implementation work the better.

Easily extendable: The currently used VoIP systems offer new versions and new features once in a while, and sometimes even change their protocols. The design of the interoperable system should have the possibility to be easily extended to support these changes. Furthermore, new VoIP systems will probably be built in the future, so the interoperable system should also be able to be extended to support new VoIP systems.

6.5 Conclusion

This chapter informs about the requirements of the interoperable VoIP system. Summarised, the interoperable VoIP system should use an existing client with no changes at user side, to make it user-friendly. To interact with the existing clients at the gateway side, a plug-in is needed (API). As many as possible VoIP systems should be integrated and also as many as features (like IM and audio). The interoperable VoIP system should be reliable and scalable with a stable performance. The user should not use the use of (another) server and the implementation should be easily extendable.

This concludes that we need a solution using the existing clients; all VoIP systems with the ability for a plug-in could be integrated. We look for a solution that supports both IM and VoIP. The next chapter gives a five possible solutions for these requirements.

Chapter 7

Design approaches

The preceding chapters provided an overview of the existing VoIP systems and the available solutions (related work) to create interoperability between those VoIP systems. Because no universal solution for interoperability among VoIP systems exists, this chapter discusses several possibilities for a universal solution. These possibilities are called possible design approaches.

All these approaches will be explained one by one in the following sections. Also advantages and disadvantages of the approaches are discussed. Finally, the chapter is concluded with the selection of one approach. First a short overview, the approaches are:

1. **Interconnected existing VoIP systems:** In this approach a gateway takes care of the interconnection between the VoIP systems, which is done at service level. The clients of the existing VoIP systems are used without any change for the user and with an extension at the Gateway side.

2. **Changes to the existing VoIP clients:** For this approach the clients of the existing VoIP systems are changed, to let them connect to the gateway. This gateway takes care of the interconnection; the gateway needs to understand the protocols of all VoIP systems.

3. **Self made client:** For this approach, we design an own client, which will be used by the user, while the buddy still uses an existing VoIP client. This self made client is connected to a gateway, which takes care of the interconnection.

4. **Self made peel client:** For this approach, again we design an own client, which embeds all client software components of existing VoIP systems in a single client. For example, the MSN component of the client can now communicate with the MSN buddy, the GTalk component of the client with the GTalk buddy, etcetera.

5. **Web client:** This approach is almost equal to the third approach. Instead of a self made client installed at the computer, the client is web based and online reachable. The gateway takes care of the interconnection.

Approach one, two, three and five include a gateway that act as an interpreter that (a) translates the protocol messages from one protocol to another, or (b) translates the services offered at higher level. The difference between (a) and (b) lies in the behaviour of the gateway, which is influenced by the location of the gateway.

7.1 Approach 1: Interconnected existing VoIP systems

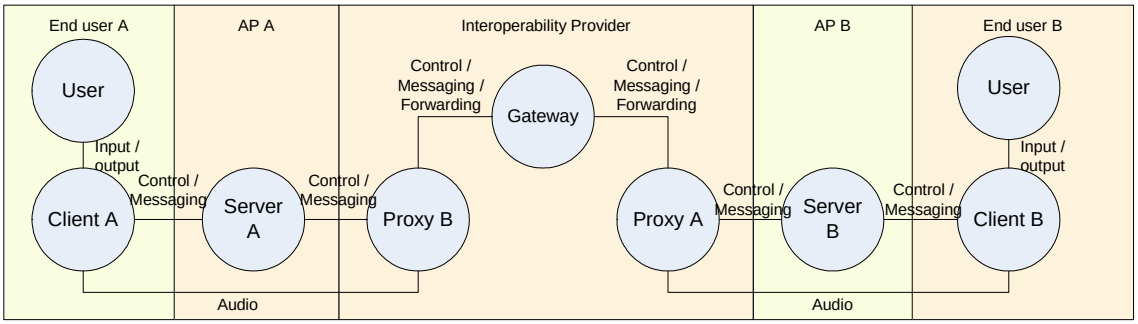


Figure 7.1: Overview of approach 1

Figure 7.1 shows the overview of the first approach. The figure shows in the middle the Gateway. On the left side of the Gateway, the figure shows VoIP system L, which consists of Client A, Server A and Proxy B (which can be seen as another Client at this moment). On the right side of the Gateway, the figure shows VoIP system R, which consists of Client B, Server B and Proxy A (which can also be seen as another Client at this moment). User A has an account at VoIP system L and User B has an account at VoIP system R.

The Interoperability is realized by the Proxies and the Gateway, as shown in Figure 7.1. The proxy element represent clients from a VoIP System R in a VoIP system L, these entities are responsible for the syntactic and (potentially semantic) translation of VoIP control plane messages and VoIP user data from between VoIP system R and an "abstract VoIP system". The responsibility of the Gateway is to properly route control plane messages and user plane data among Proxies. The Gateway is an intermediary between calls from users of VoIP system L and VoIP system R. The Gateway tunnels the control, messaging and audio from one Proxy to the other, to let the User think he is directly connected to his buddy (without the Proxies), but in the system, the Client is actually connected to the representation of the Client of the buddy, the Proxy.

The Proxies are extended clients in a VoIP system. This means, Proxy B is a extended client in VoIP system L, almost like Client A. Without the interconnection, Proxy B can also be a buddy of Client A. Using this approach for interconnection, the two proxies tunnel the data (control, messaging and audio) to create a connection between Client A and Client B. In this way, for the APs the Proxy and Client will look like two normal Clients connected to each other, and for users it looks like the users are directly connected to each other.

Somewhat more specific, the proxies are representations in VoIP system R of clients in VoIP system L (and vice versa), i.e. these are not just clients, proxies are representations of clients in another VoIP technology.

As can be seen in the figure, the Client is always connected to its own server and can have peer-to-peer connections with the Proxy. Furthermore, the Client can have peer-to-peer connections with other Clients.

Initially, the user can not yet communicate with buddies in another VoIP system. Furthermore, no proxies exist for this client yet. To initiate a relationship with a user from another VoIP system, the user first needs to add a representation of the Gateway to his contact list. This representation provides the possibility to communicate with the Gateway and is called from now on the Gateway Client. The Gateway Client is not shown in Figure 7.1. The Gateway Client is the buddy client of the user client in the same network. In Figure 7.1 this means the Gateway Client is situated where the Proxy is. After adding this Gateway Client as a buddy, the user can send text messages to the Gateway Client. These messages should contain commands, to command the Gateway what to do. To know what commands are supported by the Gateway, the Gateway Client could send a message with a list of the supported commands to the client, so the user can read it.

Let us assume User A wants to add User B to its contact list, which means User A and User B become buddies. We assume that User B is online and ready to receive messages from another user. To initiate this relationship, the following steps take place to add a buddy:

1. User A uses Client A to send a message to the Gateway Client, containing the ID of Client B.
2. The Gateway creates an account at VoIP system R, using a unique identifier by combining the user ID of User A and an ID for VoIP system R.
3. The Gateway starts an instance of the client process for VoIP system R. In fact, the client using this new account is called Proxy A.
4. The Gateway makes Proxy A send a buddy search request of Client B
5. The VoIP system Server B responds with the Client B data
6. The Gateway creates an account at VoIP system L, using a unique identifier by combining the user ID of User B and an ID for VoIP system L.
7. The Gateway starts an instance of the client process for VoIP system L. In fact, the client using this new account is called Proxy B.
8. The Gateway Client sends a message to Client A, containing the unique ID of Proxy B, which is read by User A.
9. User A responds to the message by performing a buddy search, Client A sends the buddy search request to its system, searching for Proxy B.
10. Client A receives the message of the Gateway Client and adds the ID of Proxy B to the contact list.
11. The Gateway knows how to tunnel data from Proxy B to Proxy A, and User B seems like a buddy of User A.

Two Proxies are created at this moment (and the Gateway Client still exists, which makes it three Proxies). After performing the buddy search of Proxy B by Client A, Client A is also added as a buddy to the contact list of Proxy B. At VoIP system R the same happens; after searching for Client B by Proxy A, Proxy A is also added as a buddy to the contact list of Client B.

The above mentioned steps are executed only once, i.e., when User A and User B become "buddies". When User A or User B goes online by starting the client, this will be noted by the Gateway and the Gateway starts the corresponding proxies Proxy A or Proxy B. As a consequence, the status (online, away from computer, etcetera) of Proxy B is mirrored by the status of Client A and vise versa. Similarly, the status of Proxy A is mirrored by Client B and vise versa. Now, the configuration is ready to start communication sessions. Let us assume again that User A takes the initiative, hence User A is the caller, and User B is the callee. The following steps take place to setup the call:

1. Client A knows the ID of the Proxy B. Client A uses this ID to check the contact list in order to see whether Proxy B (and thus Client B) is online. Let us assume Client B is online
2. User A initiates the communication according to the mechanisms of VoIP system L. Proxy B, recognized the call from Client A.
3. Proxy B signals the Gateway to setup a call. Proxy B is the unique proxy of Client B hence, the Gateway knows that this concerns a call from Client A at VoIP system L to Client B at VoIP system B.
4. The Gateway signals Proxy A to setup a call at VoIP system R. Proxy A is the unique proxy of Client A hence, the proxy knows this concerns a call to Client B at VoIP system R.
5. Proxy A uses the mechanisms of VoIP system R to setup a call to Client B at VoIP system R.
6. Client B knows the ID of Proxy A, recognizes the ID and User B accepts the call using the mechanisms of VoIP system R.
7. Proxy A will be informed by this acceptance and signals the Gateway about the acceptance of the call
8. Proxy B is now allowed to accepts the call between Client A and Proxy B using the mechanisms of VoIP system L.
9. Audio sent from Client A to Proxy B is translated and tunneled by the Gateway to Proxy A, which sends the audio to Client B, using the mechanisms of VoIP system R.
10. Audio sent from Client B to Proxy A is translated and tunneled by the Gateway to Proxy B, which sends the audio to Client A, using the mechanisms of VoIP system L.

The translation and tunneling of the audio needs more explanation. Client A reads the signal of a microphone and detects the speech of User A (the audio signal) and creates an audio stream according to the standards of VoIP system L. The audio stream is transferred to Proxy B using the mechanisms of VoIP system L. Proxy B receives the audio stream and transforms the audio stream to an audio signal according to the standards of VoIP system L. The Gateway feeds this audio signal into Proxy A. Proxy A creates an audio stream according to the standards of VoIP system R. The audio stream is transferred to Client B using the mechanisms of VoIP system R. Client B receives the stream and transforms the audio stream into an audio signal according to the standards of VoIP system R. The audio signal is fed to

loudspeakers and is finally heard by User B. Similarly, the speech of User B is transferred to User A.

Using this approach, the Proxies should alway be in the same state (online/offline) as the accompanying clients. Once the ID's of the proxies are entered in the contact list, the communication between users is transparent, i.e., the users need not be aware of the fact that they are using different systems.

This approach has some advantages and disadvantages. To identify these (dis-)advantages the requirements formulated in Chapter 6 are considered.

Advantages:

- The user can use one client
- The user can use existing VoIP systems and thus the client the user is familiar with
- All features (messaging, VoIP, etcetera) are included
- It is user-friendly after the buddy is added to the contact list
- The user does not notice the Gateway after the buddy is added to the contact list
- The APs do not have to make any changes to their VoIP systems, they only have to provide an API
- Network problems (like NAT issues, firewalls, etcetera) are handled by the existing VoIP systems and their protocols
- GTalk and MSN have already reached interoperability with a same kind of approach [15]

Disadvantages:

- The user notices the Gateway during the buddy search, for exact to be, the user needs to know how to connect to the Gateway Client.
- The commands send to the Gateway Client (like buddy search in another network) is done by sending text messages. This can be difficult to understand by the user.
- Multiple instances of the client should be possible to have, this rises the question whether it is scalable enough.

7.2 Approach 2: Changes to the existing VoIP clients

Figure 7.2 shows the overview of the second approach. The figure shows in the middle the Gateway. At the right side is VoIP system R which consists of Proxy A, Server B and Client B. The right side of the figure handles as described in approach 1. The left side of the figure shows VoIP system L and consists of Client A and Server A. As can be seen, there is no Proxy

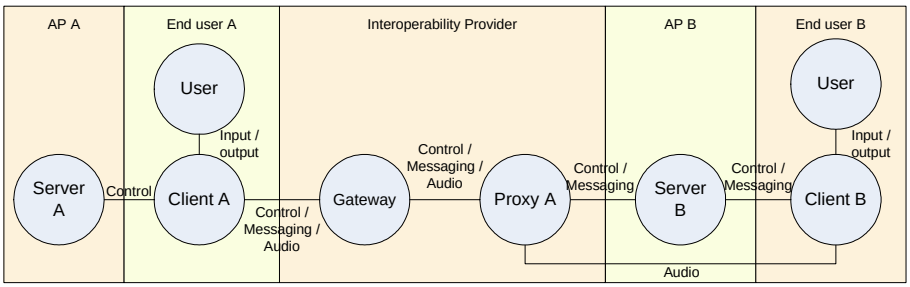


Figure 7.2: Overview of approach 2

B in this approach. Actually, the Gateway acts as Proxy B. User A has an account at VoIP system L and User B has an account at VoIP system R.

The Interoperability is realized by the Proxy and the Gateway, as shown in Figure 7.2. The proxy element represent clients from a VoIP System L in a VoIP system R, these entities are responsible for the syntactic and (potentially semantic) translation of VoIP control plane messages and VoIP user data from between VoIP system R and an "abstract VoIP system". In the first approach, the responsibility of the Gateway was to properly route control plane messages and user plane data among Proxies. However, in this approach is only one proxy, which means the Gateway has to act as a proxy as well. This makes the responsibility of the Gateway to properly route control plane messages and user plane data from the proxy and to the proxy. Furthermore, the Gateway needs to understand every 'language' of the VoIP systems connected to the Gateway. In other words, if Skype is connected to the Gateway, the Gateway needs to understand the (encrypted) Skype messages and Skype data and if GTalk is connected to the Gateway, the Gateway needs to understand the XMPP data.

For this approach, Client A has to login to Server A, its own server, and when the authentication is confirmed, the Client logs in to another server, the Gateway. Now User A can communicate with other users in his network by using his own server. To communicate with users of another VoIP system, the Gateway is used.

The Gateway is connected to Client A. To understand the messages send by and to Client A, the Gateway needs to understand the protocol of Client A. This means the Gateway needs to understand the protocols of all VoIP systems that could be connected to the Gateway. The Gateway translates the incoming protocol messages in outgoing protocol messages; the Gateway is situated at protocol level.

Initially, the user can not yet communicate with buddies in another VoIP system. Furthermore, no proxy exist for this client yet. To initiate a relationship with a user from another VoIP system, the client (let us assume Client A) first needs to login to the Gateway. After this authentication is successful, the Gateway will create a Proxy A at VoIP system R. Again, the communication between the Gateway and VoIP system R is equal to the situation explained in approach 1.

Notice that the User has nothing to do with the login to the Gateway. It is the Client that initiates and finishes the login to the Gateway. The client should know how and what to do,

so the client needs an extension to create this possibility.

Let us assume User A initiates a relationship with User B, which means User A and User B become buddies. We assume that User B is online and ready to receive messages from another user. To initiate this relationship, the following steps take place to add a buddy:

1. Client A logs in to the Gateway
2. The Gateway creates an account at VoIP system R, using a unique identifier by combining the user ID of User A and an ID for VoIP system R.
3. The Gateway starts an instance of the client process for VoIP system R. In fact, the client using this new account is called Proxy A.
4. The Gateway makes Proxy A send a buddy search request for Client B
5. The VoIP system server B responds with the Client B data
6. The Gateway translates and forwards the Client B data to Client A
7. Client A receives the message of the Gateway and adds the ID of Client B to the contact list.

Already some difficulties arise: The Gateway is not able to understand all VoIP system protocols. Furthermore, User A also wants to communicate with other buddies, in the same VoIP system. The client should know which data should be sent to the VoIP system server, and which data should be sent to the Gateway. Furthermore, two contact lists exist; the contact list of Client A created by its own Server A and the contact list of Client A created by the Gateway (with buddies of other VoIP systems). Let us assume for now this is all taken care of. After adding the buddy to the contact list, the configuration is ready to start communication sessions. Let us assume again that User A takes the initiative, hence User A is the caller, and User B is the callee. The following steps take place to setup the call:

1. Client A knows the ID of the Client B. Client A uses this ID to check the contact list in order to see whether Client B is online. Let us assume Client B is online
2. User A initiates the communication according to the mechanisms of VoIP system L, however the Gateway is used as the server in stead of Server A
3. The Gateway signals Proxy A to setup a call at VoIP system R. Proxy A is the unique proxy of Client A hence, the proxy knows this concerns a call to Client B at VoIP system R
4. Proxy A uses the mechanisms of VoIP system R to setup a call to Client B at VoIP system R
5. Client B knows the ID of Proxy A, recognizes the ID and User B accepts the call using the mechanisms of VoIP system R
6. Proxy A will be informed by this acceptance and signals the Gateway about the acceptance of the call

7. The Gateway translates and forwards the acceptance to Client A
8. Audio sent from Client A to the Gateway is translated and tunneled by the Gateway to Proxy A, which sends the audio to Client B, using the mechanisms of VoIP system R.
9. Audio sent from Client B to Proxy A is translated and tunneled by the Gateway to Client A, using the mechanisms of VoIP system L.

The translation and tunneling of the audio is done in the way of the first approach. Instead of using Proxy B, the Gateway is used, which means the audio stream is transferred to the Gateway. The Gateway receives the audio stream and transforms the audio stream to an audio signal. The Gateway feeds this audio signal into Proxy A. Proxy A creates an audio stream according to the standards of VoIP system R. The audio stream is transferred to Client B using the mechanisms of VoIP system R. Client B receives the stream and transforms the audio stream into an audio signal according to the standards of VoIP system R. The audio signal is fed to loudspeakers and is finally heard by User B. Similarly, the speech of User B is transferred to User A. The moment Proxy A receives the audio stream and transforms it to an audio signal, the Gateway transforms this signal into an audio stream according to the standards of VoIP system L.

This approach has some advantages and disadvantages. To identify these (dis-)advantages the requirements formulated in Chapter 6 are considered.

Advantages:

- The user can use one client
- The user can use existing VoIP systems and thus the client the user is familiar with
- All features (messaging, VoIP, etcetera) are included
- It is user-friendly, the user does not have to send text commands, he can use the buttons at the client
- The user does not notice the Gateway
- It is better scalable then approach 1; less proxies are needed

Disadvantages:

- The client of the VoIP systems have to be changed and the user has to install new software on his computer. If the APs change the VoIP system, it will not be difficult for the user to install the client. However, if somebody else has to do it, issues like how to distribute the changed client to make users use it and how to implement it will arise.
- Network problems (like NAT issues, firewalls, etcetera) between Client A and the Gateway need to be taken care of by the implementers of the Gateway.
- It is not for all data streams (like the proprietary protocol Skype) possible to convert

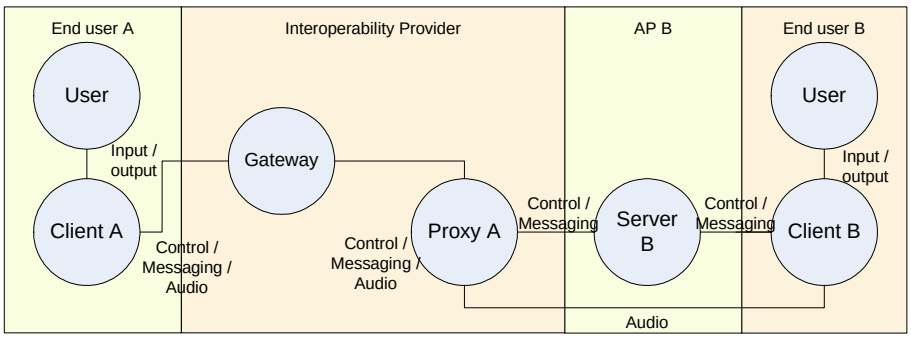


Figure 7.3: Overview of approach 3

7.3 Approach 3: Self made client

Figure 7.3 shows the overview of the third approach. The figure shows on the left side of the middle the Gateway. On the left side of the Gateway, the figure does not show a VoIP system L, it only shows Client A (and its User A). On the right side of the Gateway, the figure shows VoIP system R, which consists of Client B, Server B and Proxy A. The right side of the figure handles as described in approach 1. User A has an account for Client A and User B has an account at VoIP system R.

Client A is not an existing client, as used in approach 1 and 2. Client A is a self made client, with a self chosen protocol. Client A is able to communicate with other clients using its server (the Gateway). Client A logs directly in to the Gateway. Other servers are not needed for this Client. If buddies are also using the self made client, it is possible to communicate like existing VoIP systems do; Client A sends a message to the Gateway), which forwards the message to Client A2 and vise versa.

If the buddy is in another VoIP system (let us say VoIP system R), Proxy A in VoIP system R tunnels messages to the Gateway. When Client A adds Client B as a buddy, User B will not notice the use of the Gateway, if Client B wants to add Client A to his contact list, it needs to add the Gateway Client to his contact list first, as described in the first approach.

Initially, the user can not yet communicate with buddies in another VoIP system. Furthermore, no proxy exist for this client yet. To initiate a relationship with a user from another VoIP system, the client (let us assume Client A) first needs to login to the Gateway. After this authentication is successful, the Gateway will create a Proxy A at VoIP system R. Again, the communication between the Gateway and VoIP system R is equal to the situation explained in approach 1.

Notice that the User initiates the login to the Gateway, it is the 'normal' login procedure of the Client to its server.

Let us assume User A initiates a relationship with User B, which means User A and User B become buddies. We assume that User B is online and ready to receive messages from another user. To initiate this relationship, the following steps take place to add a buddy:

1. Client A logs in to the Gateway
2. The Gateway creates an account at VoIP system R, using a unique identifier by combining the user ID of User A and an ID for VoIP system R.
3. The Gateway starts an instance of the client process for VoIP system R. In fact, the client using this new account is called Proxy A.
4. The Gateway makes Proxy A send a buddy search request of Client B
5. The VoIP system server B responds with the Client B data
6. The Gateway translates and forwards the Client B data to Client A
7. Client A receives the message of the Gateway and adds the ID of Proxy B to the contact list

As can be seen, the buddy search seems equal to approach 2. The difference is at the end; when the Gateway informs Client A about the buddy data (sends the contact list). First, in approach 2 there are two contact lists (of the Server and of the Gateway) and for this approach only one contact list exist (the Gateway is the Server). Second, the Gateway has to translate at protocol level for the communication between Client A and the Gateway. For approach 2 this is rather difficult, because of the closed protocols like Skype. In this approach, the protocol used is a self chosen protocol and thus the Gateway is able to understand the protocol.

After adding the buddy to the contact list, the configuration is ready to start communication sessions. Let us assume again that User A takes the initiative, hence User A is the caller, and User B is the callee. The following steps take place to setup the call:

1. Client A knows the ID of the Proxy B. Client A uses this ID to check the contact list in order to see whether Client B is online. Let us assume Client B is online
2. User A initiates the communication to the Gateway
3. The Gateway signals Proxy A to setup a call at VoIP system R. Proxy A is the unique proxy of Client A hence, the proxy knows this concerns a call to Client B at VoIP system R
4. Proxy A uses the mechanisms of VoIP system R to setup a call to Client B at VoIP system R
5. Client B knows the ID of Proxy A, recognizes the ID and User B accepts the call using the mechanisms of VoIP system R
6. Proxy A will be informed by this acceptance and signals the Gateway about the acceptance of the call
7. The Gateway translates and forwards the acceptance to Client A
8. Audio sent from Client A to the Gateway is translated and tunneled by the Gateway to Proxy A, which sends the audio to Client B, using the mechanisms of VoIP system R

9. Audio sent from Client B to Proxy A is translated and tunneled by the Gateway to Client A

Again, the call steps looks similar to the call steps of approach 2. The main difference is that the Gateway uses a self chosen protocol for the communication between the Gateway and Client A.

This approach has some advantages and disadvantages. To identify these (dis-)advantages the requirements formulated in Chapter 6 are considered.

Advantages:

- The user can use one client
- All features (messaging, VoIP, etcetera) are included
- It is user-friendly, because the layout can be self designed (plug-ins are not necessary)
- The user does not notice the Gateway
- The APs do not have to make any changes to their VoIP systems, they only have to provide an API (for Proxy A)
- The VoIP system is easy to extend and also a version for mobile phones and PDAs can be made

Disadvantages:

- Users have to get used to a new client
- Network problems (like NAT issues, firewalls, etcetera) should be taken care of by the implementers of the Gateway
- It is not widely adopted and available
- It is hard to compete with the existing well-known VoIP systems
- It needs a lot of research and implementation to build a good VoIP system, considering layout issues and reliability and quality of the gateway and the connection between the gateway and the client

7.4 Approach 4: Self made peel client

Figure 7.4 shows the fourth approach and shows on the left side Client A including a converter, in the middle a Server and on the right side Client B.

This approach can be seen as follows: The user installs the Clients of all VoIP systems it wants to use (and achieves an account for every VoIP system it wants to use). Now it can use the Client of MSN to communicate with an User of the MSN system, the GTalk Client to communicate with an User of the GTalk system, etcetera.

In stead of installing the clients of all VoIP systems, just one self made peel Client can be installed, which is Client A in Figure 7.4. To use all the VoIP systems, accounts for the VoIP

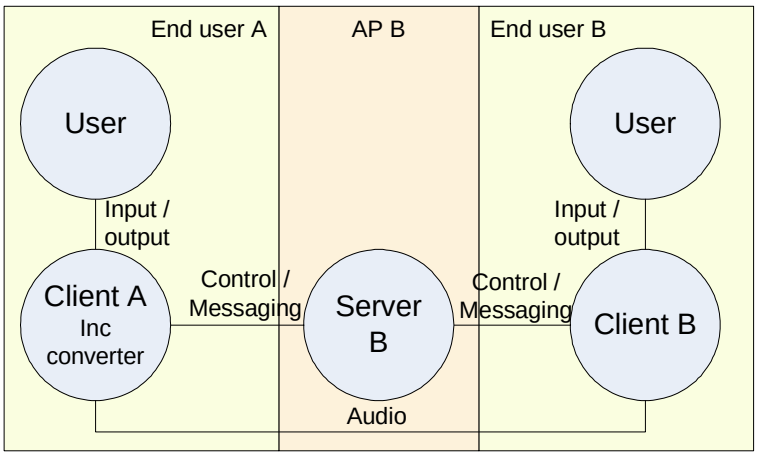


Figure 7.4: Overview of approach 4

systems the user wants to use have to be activated at this Client. Now the MSN component of Client A is used to communicate with the buddy using a MSN Client, the GTalk component to communicate with a buddy using a GTalk Client, etcetera. An example of such a system is Trillian [2]. Only this new self made peel Client has to be used, other Clients do not have to be installed on the computer.

Let us assume User A initiates a relationship with User B, which means User A and User B become buddies. We assume that User B is online and ready to receive messages from another user. To initiate this relationship, the following steps take place to add a buddy:

1. User A logs in. This means Client A uses, for example, its MSN account to log in to the MSN Server, its GTalk account to log in to the GTalk Server, and etcetera
2. User A uses Client A to send a message to the Gateway Client, containing the ID of Client B
3. Client A sends a buddy search request using the same kind of account as the buddy is, e.g. it uses its MSN account to search for a MSN buddy, and etcetera
4. Client A receives the message of the Server and adds the ID of Client B to the contact list.

The user will see all of its buddies in one contact list, so the different contact lists of the different VoIP systems are combined. The communication with a buddy is done directly by the protocol of the AP (which can differ from buddy to buddy). To display the data the way we want in stead of the way the existing Clients do, a converter (which is included in the client) should interpret the incoming data. The APIs of the several VoIP systems should be used to be able to interpret this data and respond to it.

Now, the configuration is ready to start communication sessions. Let us assume again that User A takes the initiative, hence User A is the caller, and User B is the callee. The following steps take place to setup the call:

1. Client A knows the ID of the Client B. Client A uses this ID to check the contact list in order to see whether Client B is online. Let us assume Client B is online
2. User A initiates the communication according to the mechanisms of the VoIP system
3. Client B knows the ID of Client A, recognizes the ID and User B accepts the call using the mechanisms of the VoIP system
4. Client A will be informed by this acceptance
5. Audio sent from Client A to Client B could be sent over a peer-to-peer connection between Client A and Client B and vise versa

Notice that both the buddy search as the call steps are simply done according to the mechanisms of the VoIP system, e.g. according to the MSN system in case of a MSN buddy, GTalk system in case of a GTalk buddy, and etcetera.

This approach has some advantages and disadvantages. To identify these (dis-)advantages the requirements formulated in Chapter 6 are considered.

Advantages:

- The user can use one client
- All features (messaging, VoIP, etcetera) are included
- It is user-friendly, only setting up is time consuming, because all accounts should be filled in by the user, but this only have to be done once
- The user does not notice the Gateway after the buddy is added to the contact list
- The APs do not have to make any changes to their VoIP systems, they only have to provide an API
- Network problems (like NAT issues, firewalls, etcetera) are handled by the existing VoIP systems and their protocols
- One client logs in to all servers belonging to the user accounts
- It is as scalable as the existing VoIP systems
- It is as secure and trustful as existing VoIP systems
- This solution is already available for IM (Trillian [2]) and partly for VoIP (Gizmo [55], etc).

Disadvantages:

- Users have to get used to a new client
- A user needs an account for every VoIP system it wants to use and have to fill them all in

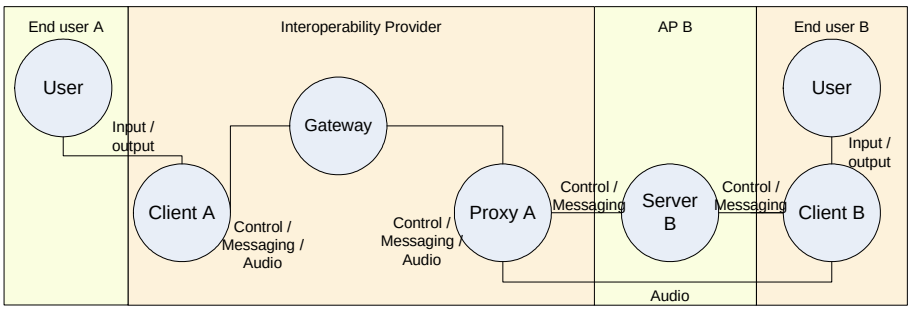


Figure 7.5: Overview of approach 5

7.5 Approach 5: Web client

Figure 7.5 is almost equal to Figure 7.3. The difference is that Client A is situated at the Interoperability Provider for this fifth approach, in stead of at the End User A. The self made client, Client A, is a web based client.

For this approach a web client is used. The user can communicate to his buddies by this web Client. The web Client is directly connected to the Gateway. The user has to login to the Gateway; no other servers will be used by this Client. Notice that the buddy uses an existing client of an existing VoIP system.

The user visits a website where the web Client will be displayed. After login to the Gateway, the Client can communicate with buddies. The Gateway has to convert messages to and from Proxy A. The Proxy handles as the Proxy described in approach 3. Also the buddy search and call steps are equal to the steps in approach 3.

This approach has some advantages and disadvantages. To identify these (dis-)advantages the requirements formulated in Chapter 6 are considered.

Advantages:

- The user can use one client
- All features (messaging, VoIP, etcetera) are included
- It is user-friendly, because the layout can be self designed (plug-ins are not necessary)
- The user does not notice the Gateway
- The APs do not have to make any changes to their VoIP systems, they only have to provide an API (for Proxy A)
- The VoIP system is easy to extend and also a version for mobile phones and PDAs can be made
- It is widely available

Disadvantages:

- Users have to get used to a new client
- Network problems (like NAT issues, firewalls, etcetera) should be taken care of by the implementers of the Gateway
- It is not widely adopted and available
- It is hard to compete with the existing well-known VoIP systems
- It needs a lot of research and implementation to build a good VoIP system, considering layout issues and reliability and quality of the gateway and the connection between the gateway and the client
- Because the client is on the Interoperability Provider side, the connection to the client can be slow

7.6 Conclusion

Approach 2 is technically not possible, because some VoIP systems have proprietary protocols, which makes the data streams unreadable. Furthermore most of the data streams are encrypted. This approach will be discarded.

Approach 4 is technically possible and is already partly done in commercial projects. VoIP is not completely covered by those projects yet, but people are working on it. This makes it less interesting for this master thesis to continue researching this subject.

Approach 3 and 5 are almost equal, because they consist of a self-made client and a gateway. For the user this approach is most user-friendly, because all functionalities can be integrated in one client. The focus for this thesis is on the gateway and not on the client, designing a client is too much work for this thesis. If we would focus on the client, also aspects like security between client / server, codecs, protocols and etcetera should be complied with.

Approach 1 focuses on the gateway with two Proxies and can be extended in the future by approach 3 or 5. For now this is the best choice to study further, because it focuses on how the different protocols can interact, based on events.

To refer to the requirements discussed in Chapter 6, approach 1 seems to obey most of these requirements. With approach 1, the user can continue using its familiar application and probably all features can be covered (in the future). It will use the protocols of the existing applications, so reliability and security issues are of limited influence.

Unfortunately there are two worrisome issues for approach one, (i) the text-based commands sent to the gateway; it is not sure if the users will understand it, and (ii) for every user and every buddy a Proxy has to be set up. If many people start using this solution, it is not clear whether it is scalable.

Chapter 8

Design of the Gateway

In Chapter 7, the Interconnected existing VoIP systems approach has been chosen as the design approach for this chapter. For this approach the clients of the existing VoIP systems are used. A Gateway takes care of the interconnection between these VoIP systems. This chapter explains the approach in more detail and describes the design of the Gateway and the proxies.

First we give a short overview and the functional requirements for the Interoperable VoIP Gateway and afterwards the structure of the design. Furthermore we discuss the behaviour of the Gateway and the proxies. The behaviour is specified with sequence diagrams and are modelled based on the service primitives and elements as presented in Chapter 4. Finally we give a discussion about encountered problems and this chapter is ended with a conclusion.

8.1 Functional requirements

This section formulates the functional requirements for the Interoperable VoIP Gateway. These requirements define how the features of an interoperable VoIP Gateway are initiated, handled and acknowledged.

The functional requirements define the main characteristics and functionalities of the Interoperable VoIP Gateway. Those characteristics are defined in Paragraph 3.1.1 and are the login, buddy search, messaging and call. The call can be divided into call setup (which is subdivided into initiation and acknowledgment), call teardown and call modification.

Login
Requirement: The Interoperable VoIP Gateway requires users to login to their own server.

The user should login to its own server(s). Only if this is valid, the Proxies can be buddies and thus only then the Gateway can be used.

Buddy search
Requirement: The Interoperable VoIP Gateway must provide the possibility for users to search for buddies in another VoIP system.

To extend the current contact list and be able to communicate with users from other VoIP systems, it should be possible to search for users in the other VoIP system. The Gateway must therefore be able to forward the search requests into the other VoIP system.

Messaging
Requirement: The Gateway in combination with the Proxies should be able to forward messages from one VoIP system to the other VoIP system.

The Proxies should be able to receive messages sent by a user. To tunnel those messages to the other VoIP system, the Gateway should know the final destination, in order to send the messages to the correct destination.

Call initiation
Requirement: The Interoperable VoIP Gateway requires users to initiate the call

The user is always the one who will initiate the communication. This applies also for login, buddy search and messaging. The Gateway (and the Proxies)will never initiate any of these processes, it just responds to the events.

Call acknowledgment
Requirement: The Interoperable VoIP Gateway requires users to acknowledge an incoming call.

A user should acknowledge the Gateway about the acceptance of the call. Again the Gateway is not allowed to accept a call by itself, the user should do this. The Gateway is allowed to accept the call after it is triggered by an event of the user.

Call termination
Requirement: The Interoperable VoIP Gateway requires users to terminate the call.

A user should terminate the call, which is noticed by the Gateway. Again the Gateway is not allowed to terminate a call by itself, the user should do this. The Gateway is allowed to terminate the call between the Gateway and the buddy Client, after it is triggered by an event of the user.

8.2 Structure

This section describes the detailed structure of the design.

Interoperable VoIP Gateway

Figure 8.1 shows the architecture of the Interoperable VoIP Gateway. It shows two end users, two VoIP systems and the Gateway. Each VoIP system is subdivided into two clients, a server and the underlying layer, for the connection between the clients and server. Furthermore, the figure shows the caller and the callee. The User needs at least a little knowledge about the Gateway. First of all, the user needs to include a Gateway Client to its contact list. The Gateway Client is a Client in the VoIP system of the User Client and is the interface to the Interoperable VoIP Gateway. The Gateway Client is able to receive text commands, which makes it possible for the user to command the Gateway. The Gateway Client is not shown in Figure 8.1.

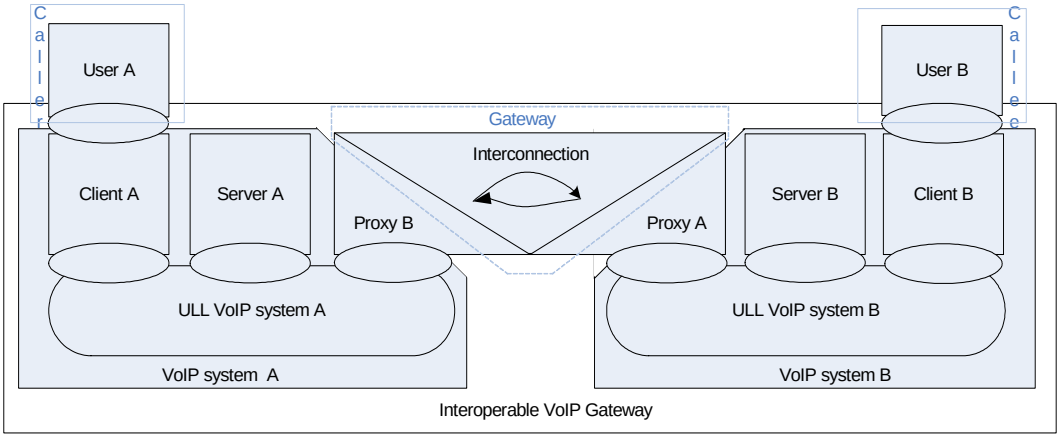


Figure 8.1: Architecture of the Interoperable VoIP Gateway

The Proxies and the Gateway are situated at the Interoperability Server. The Gateway consists of plug-ins (the plug-ins use the API of the VoIP system, see Section 8.4.2) for the Clients and Interconnection.

Notice that VoIP system A and VoIP system B are the VoIP systems MSN, GTalk, Yahoo, ICQ or Skype and do not have to change their architecture or functionalities. Only at the Gateway, the two Clients have a plug-in (the combination Client and Plug-in is the Proxy) to handle the incoming data (seen from the ULL) and outgoing data (seen from the Gateway), this will be explained in more detail in Section 8.4.2.

Proxies and Gateway

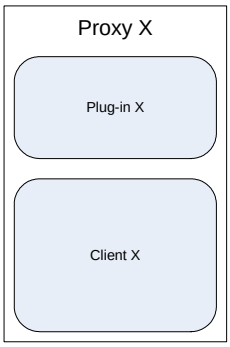


Figure 8.2: Proxy

The Proxies are the combination of the VoIP system Client and the plug-in to this Client. Figure 8.2 shows the Proxy divided in the Client and Plug-in.

Figure 8.3 shows the Proxy (the Client and Plug-in) and the Gateway (the Plug-in and interconnection). The figure also shows two numbers, number 1 and number 2, to show the

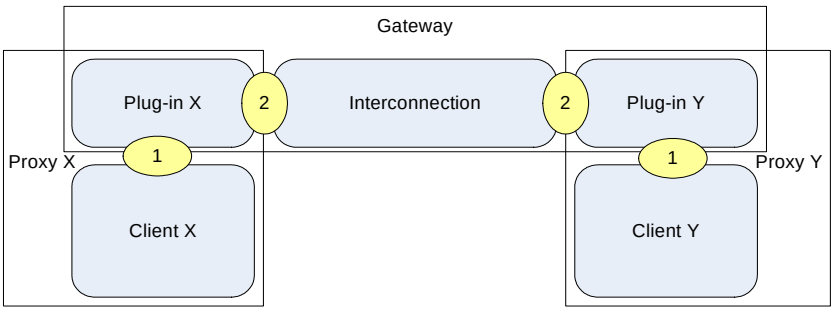


Figure 8.3: Proxy and Gateway

connections between the Client and Plug-in and the Plug-in and Interconnection. When the data from the Server or buddy Client is fed to the Client, the plug-in translates this data to the Gateway language using the API of the VoIP system. This means that API functions are used at number 1 in the figure to get this data. Inside the Plug-in, the data is translated into the Gateway language. Inside the Gateway (remember the Gateway consists of the plug-ins and the Interconnection), the communication is done with this Gateway language (number 2 in the figure).

Interconnection

The Interconnection is responsible to handle the incoming events and make sure the correct outgoing events are triggered. The following situations can occur:

1. **Message forwarding:** The text message is translated to the Gateway language and just forwarded from one plug-in to the other plug-in

2. **Buddy search** When the user of a VoIP system wants to search for a buddy in another VoIP system, it has to command the Gateway to perform this search. This is done by sending a text message to the Gateway Client. When this message arrives at the Gateway, the Gateway starts a buddy search in the other VoIP system. When the searched buddy is found, the Gateway sends the buddy information to the originating user. Furthermore, it sets up a new Proxy, which represents the Client of the buddy.

3. **Call setup and tear down handling:** When a user starts a call, this is noticed at the Gateway. The Gateway starts a call at the other VoIP system to the desired buddy Client. If the Gateway receives an accept of this buddy, it will accept the originating call request of the user. Now the call is setup. Furthermore, if the user closes the call, the Gateway handles it too.

Also audio has to be connected between Proxy A and Proxy B. This is done differently then the three items just discussed. The audio forwarding is handled by a so called Virtual Audio Cable, which has the end points at the number 1s of Figure 8.3

1. **Audio handling:** When the call is setup, there is a connection between Client A and Proxy B, and between Client B and Proxy A. The Gateway forwards the audio between the two Proxies, i.e. Proxy A and Proxy B. This is done by connecting the audio output of Client X (see Figure 8.3) to the audio input of Client Y and vice versa.

8.3 Behaviour of the Interoperable VoIP Gateway

This section explains the behaviour of the Interoperable VoIP Gateway. First the term 'event' is explained by an example. When a user starts a call (with the Client to the Proxy), the plug-in notices the request for a call. This event is handled by the Gateway by sending a request for call to the other user (via the other Proxy to the other Client). If the call is accepted, the Gateway notices the "accept" event and is allowed to accept the originating call request. This example shows that the Gateway responds to the incoming events to the plug-ins of the Proxies and it triggers new events on the other plug-ins of the Proxies.

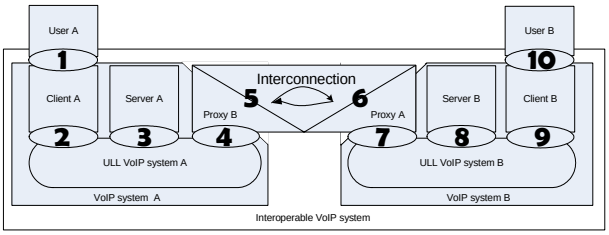


Figure 8.4: SAP numbering

Next we explain the behaviour of the Gateway per service element. The interactions are illustrated with sequence diagrams. These sequence diagrams use Service Access Points (SAPs), which numbers are equal to the SAPs in figure 8.4. The sequence diagrams are based on the information gathered in Chapter 3 and 4 and Appendix B.

8.3.1 Login

For the login, the Gateway is not used. When the user logs in, the client logs in to its own server, situated at the VoIP system.

8.3.2 Buddy search

Figure 8.5 shows the minimum interactions for a buddy search in another VoIP systems. To search a buddy in another VoIP system, the user opens the Gateway buddy screen, types a message and sends the message to the Gateway client. This message contains the command to search for a buddy, the buddy name the user is looking for and the VoIP system where the buddy can be found.

When the Gateway Client receives the message with the buddy search command, it first creates a representation of Client A, the Proxy A. Afterwards, it enters the buddy name to search for a buddy and the Proxy receives an extended contact list, with this new buddy included. The Gateway will use this buddy information to set up a representation of Client B, the Proxy B. Furthermore, the Gateway sends a text message with the information of Proxy B to Client A. User A can now perform a buddy search in its own VoIP system, searching for Proxy B. Notice the two number four SAPs in figure 8.5, the first is the SAP between the ULL and the Gateway client and the second is the SAP between the ULL and Proxy B.

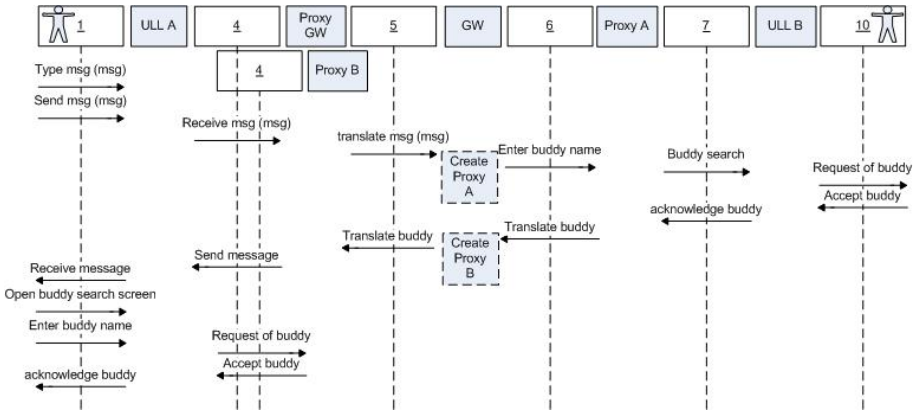


Figure 8.5: Sequence diagram buddy search minimum

Figure 8.6 also shows the interactions for a buddy search. In this figure, all optional interactions are included too. The main difference to Figure 8.5 is when the Gateway starts the buddy search and the Proxy receives the possible buddies. In stead of just receiving an extended contact list, as can be seen in Figure 8.5 a list of possible buddies is returned. The user should choose between those possible buddies first. This is done by sending a text message with the possible buddies list to User A. User A can now send back a text message with the command to the Gateway to search for one specific buddy. Now the Gateway can start the buddy search, Proxy A receives an extended list, the Gateway sets up Proxy B as representation of Client B and User A receives a text message with the information about Proxy B, and searches for this buddy.

Buddies can be added for example to the contact list with the following structure:
someuser%gmail.com@interoperableserver@hotmail.com or
someuser%hotmail.com@interoperableserver@gmail.com, etcetera.

where someuser%gmail.com and someuser%hotmail.com is the identification of the Client, @hotmail.com and @gmail.com shows the VoIP system and the total key is the identification for the Proxy.

8.3.3 Messaging

Figure 8.7 shows the sequence diagram. To send a message from User A to User B, User A opens the screen of the buddy (Proxy B) and types the message. While typing, Proxy B receives a BIT message. When User A sends the message, Proxy B receives this message. The Gateway opens the screen of the buddy (Client B), gets the message from Proxy B and enters this message into Proxy A. Client A receives a BIT message and after Proxy A sends the message, User B receives the message. Notice that the BIT messages are optional, not every VoIP system supports BIT messages. For those who do not support BIT messages, the received BIT messages are just ignored and not sent into that system and if there is no BIT message, but the other VoIP system does need a BIT message, the Gateway generates a BIT message.

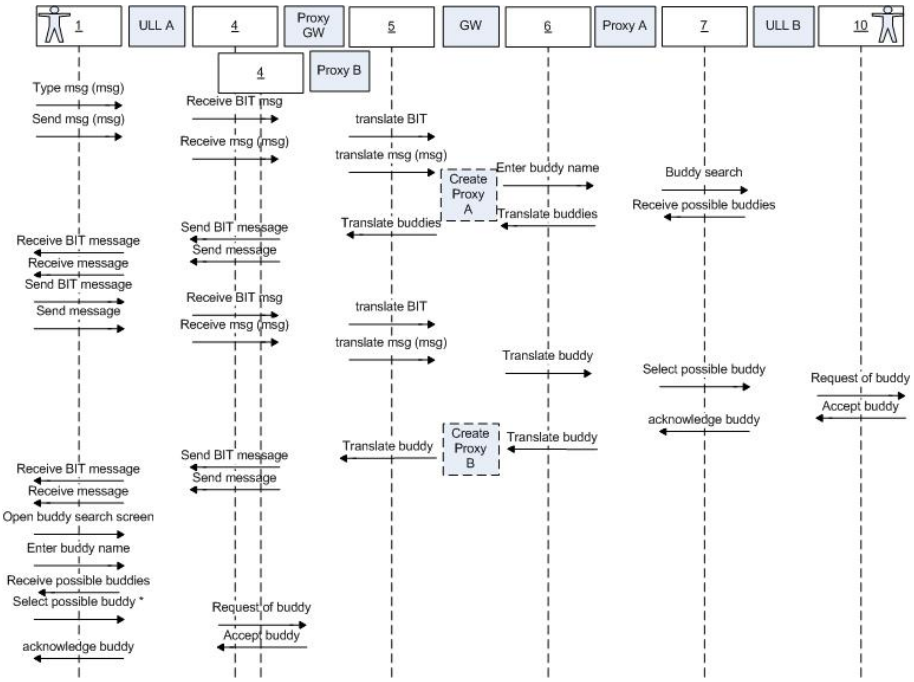


Figure 8.6: Sequence diagram buddy search full

8.3.4 Call

Figure 8.8 shows the interactions for the call. First the User starts the call. Proxy B receives the request to accept the call. The Gateway handles this request by starting a call from Proxy A to Client B. User B receives the request to accept the call. If User B accepts, Proxy B receives an acknowledgment. The Gateway sees this acknowledgment and knows it is allowed to accept the call from User A to Proxy B. Two connections exist now, between Client A and Proxy B and between Proxy A and Client B. The incoming audio at the Proxies are forwarded by the Gateway to outgoing data at the other Proxy. In this way, User B hears the voice of User A and User A hears the voice of User B. If a User ends the call, this is noticed by the Gateway. The Gateway will respond to this by closing the other connection too.

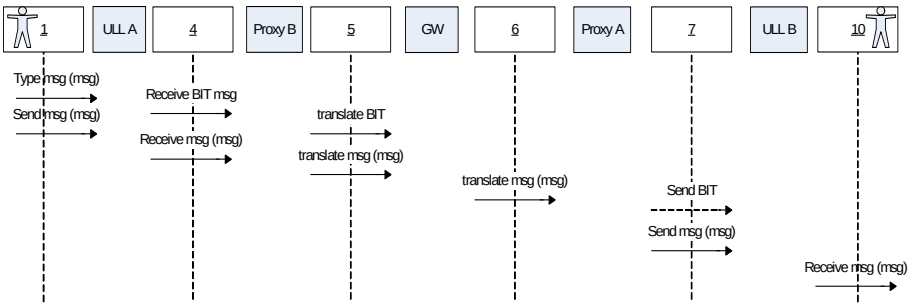


Figure 8.7: Sequence diagram messaging

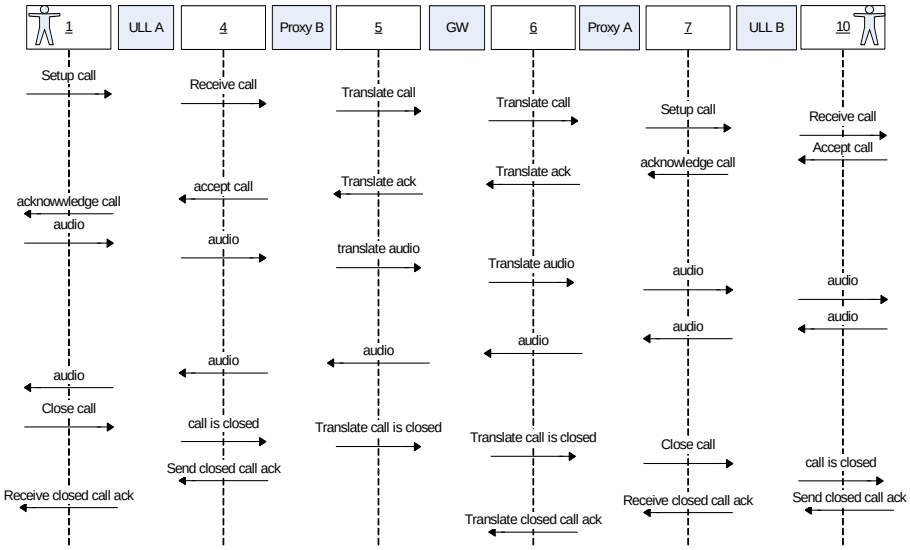


Figure 8.8: Sequence diagram call

8.4 Behaviour of the Proxies

This section explains the behaviour of the proxies. We will focus on the following three items:

1. **Multiple instances** have to run at the Gateway to create Proxies for all users and buddies.
2. **Plug-ins** have to be used at the Gateway to extend a Client to a Proxy
3. **Buddy search** requests have to be translated by the Gateway between the Proxies
4. **Messaging** messages have to be translated by the Gateway between the Proxies
5. **Audio forwarding** is not an event the Gateway can respond to, the audio has to tunneled between the proxies

8.4.1 Multiple instances

The VoIP systems MSN, GTalk, Yahoo, ICQ and Skype do not offer multiple instances of their clients. However, using some tricks or installing extensions, provides the possibility to create multiple instances for all VoIP systems.

MSN
MSN Messenger Polygamy [31] is an extension to MSN to create multiple instances.

GTalk
GTalk can offer multiple instances by execute a trick. A parameter named "nomutex" has to be added to GTalk. Information about this parameter and others can be found in [10] and how to execute this trick can be found in [14]. Also Google Talk Poligamy [13] can be used, this is an extension to GTalk to offer multiple instances.

Yahoo
Yahoo Multi Messenger [58] is an extension to Yahoo to create multiple instances.

ICQ
ICQ has two possibilities: An extension to ICQ lite [19], or an application to change the registry [34]. With both possibilities, ICQ can offer multiple instances.

Skype
Skype can have multiple instances by using different credentials [39]. Under Windows XP, this can also be realized by using different Windows XP users.

8.4.2 Plug-in possibilities

First we will look into the way the clients can be extended to act as a proxy. Therefor we need to examen the API of the clients and determine a way to implement this behaviour as an extension. This extension, also called a plug-in, is necessary to be able to get information out of the Client and put information into the Client.

All researched VoIP systems support plug-in facilities. Table 8.1 shows the VoIP systems and their websites for development. These websites include information about the plug-ins.

MSN	`http://dev.live.com` (SDK) [73]
GTalk	`http://code.google.com/apis/talk/open_communications.html` [71]
Yahoo	`http://developer.yahoo.com/messenger` (SDK) [75]
ICQ	`http://www.icq.com/webtools/app-develop.html` (API) [72]
Skype	`https://developer.skype.com/` (API) [74]

Table 8.1: Plug-in facilities

Chapter 4 introduces the sequence diagram for the call.The following steps in the sequence diagram are mentioned:

- Start call
- Receive request for call
- Accept call
- Have conversation
- Close call
- Receive call is closed message

Furthermore in stead of an Accept call, also a reject call can be sent.

The functions accept and generate the incoming and outgoing events. The events should be mapped with functions understood by the Gateway. Table 8.2 presents the mapping.

Start call	callSetup(id)
Receive request for call	callRequest(id)
Accept call	callAccept(id)
Reject call	callReject(id)
Have conversation	Mapping explained in Section 8.4.5
Close call	callTeardown(id)
Receive call is closed message	callTeardownAck(id)

Table 8.2: Mappings of the Gateway

Table 8.2 shows the functions of the Gateway code. The (id) is the source or the destination, i.e. the Client or buddy Client. It can be a name, e-mail address or another unique name.

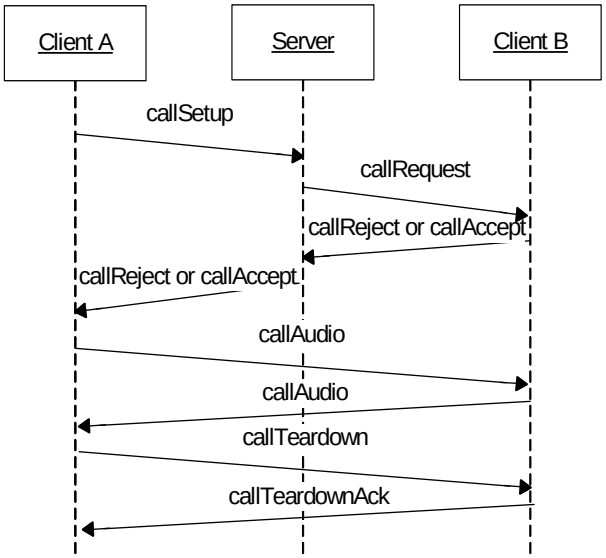


Figure 8.9: Sequence diagram of the call functions

Figure 8.9 shows the sequence diagram of the call, showing the call functions.

When a call is setup by Client A and sent to Proxy B, the Gateway receives callRequest(id) and transforms it to a callSetup(id) from Proxy A to Client B. Table 8.3 shows the rest of the mappings.

The following sections present three VoIP systems, GTalk, Skype and ICQ, and shows the mapping between the functions of the Gateway and the functions provided by the APIs of the three VoIP systems. These three VoIP systems are randomly chosen, we did not look closer at the APIs of Yahoo and MSN.

Every next section starts with a table; this table shows the mapping between the functions of the Gateway and the functions of the VoIP system API. Furthermore the functions of the VoIP system API are explained.

Incoming	Outgoing
callRequest(id)	callSetup(id)
callAccept(id)	callAccept(id)
callReject(id)	callReject(id)
callAudio()	*callAudio()*
callTeardownAck(id)	callTeardown(id)

Table 8.3: Mappings inside the Gateway

GTalk

callSetup()	↔	Call::InitiateSession(destination)
callRequest()	↔	CallClient::OnSessionState(call, session, STATE_RECEIVED_INITIATE)
callAccept()	↔	Call::AcceptSession(call_→sessions()[0])
callReject()	↔	Call::RejectSession(call_→sessions()[0])
callTeardown()	↔	Call::TerminateSession(session) and Call::Terminate()
callTeardownAck()	↔	Call::Terminate()

Table 8.4: Mappings between GTalk API and the Gateway

The destination of the InitiateSession(destination) is the JID (Jabber ID). The session of the OnSessionState contains information about the caller. For the call teardown, the Session and the call is closed.

Call setup

According to [80]:
The high-level object that manages the important actions in a voice call is called Call. A Call object manages any number of peer-to-peer Session objects, each representing one peer-to-peer connection. The Call object is the top level object to make calls, accept or reject incoming calls, monitor the status of the call, and performs other high level actions on call connections. CallClient wraps all required steps for making a call in its MakeCallTo method. Here are the basic steps taken by CallClien:

1. *Create the Call object by calling PhoneSessionClient::CreateCall*

2. *Connect to the PhoneSessionClient::SignalCallDestroy to monitor when the all sessions in the call have ended and the call object is being destroyed. A call is destroyed when all the Session objects are destroyed, which can happen by request of the current user, by request of the other user, or by a connection failure.*

3. *Connect to the new Call object's SignalSessionState signal to monitor progress of the connection and send notifications to the user ("sent initiate ," "received initiate ," "in progress," and so on).*

4. *Send the connection request to the other user. Call Call::InitiateSession with the Jabber ID (JID) of the person to connect to.*

5. *Listen for the STATE_INPROGRESS message associated with that session, which will indicate that the connection request has been accepted and begun. The Call object will handle all the details of connecting and managing the connection for you.*

The following code from CallClient::MakeCallTo starts a call to another user, specified by JID.

```
// Let us know when the Call object is destroyed, so we can close the UI
// or alert the user.
// This only needs doing once per PhoneSessionClient.
phone_client()→SignalCallDestroy.connect(this, &CallClient::OnCallDestroy);

...

// Create the call object.
call_ = phone_client()→CreateCall();

// Connect to receive session notifications.
call_→SignalSessionState.connect(this, &CallClient::OnSessionState);

// Make the connection request to the other user
session_ = call_→InitiateSession(buddy_jid);

// libjingle audio engine only handles one active voice channel at a time.
// Set the focus on the newly created conversation.
phone_client()→SetFocus(call_);

...
// Listen to the progress of the call and alert the user.
void CallClient::OnSessionState(cricket::Call* call,
cricket::Session* session,
cricket::Session::State state) {
if (state == cricket::Session::STATE_INPROGRESS) {
console_→Print("Call connected.");
} else if (state == cricket::Session::STATE_RECEIVEDREJECT) {
console_→Print("Other side rejected the request.");
}
...other conditions...
}
}
```

Receiving a Call

1. *An incoming call triggers PhoneSessionClient to send its SignalCallCreate signal. You connected to this signal earlier as part of your initial setup. When an incoming call request is received, the PhoneSessionClient creates a new Call object and sends this signal, along with the Call object. Because SignalCallCreate is sent whether you or someone else created the Call object, the only way to find out what caused this call is to connect to the Call object's SignalSessionState signal.*

2. *The new Call object sends a SignalSessionState signal describing the new connection request. SignalSessionState notifications include the managing Call object, the Session object (which contains information about the caller), and an enumeration indicating what is happening (outgoing call, incoming request, etc). See Session for*

a description of the important enumeration values. Incoming call requests are sent STATE_RECEIVEDINITIATE.

3. *Alert the user that a new call request has been made, and allow them to accept or reject the request. libjingle will respond to the call request automatically with session negotiation stanzas, but will not begin exchanging data until the user has explicitly accepted a connection request. Accept a call using Call::AcceptSession; reject a call using Call::RejectSession. You must pass in the session holding this request. Either use the Session object you received from SignalSessionState, or use the first Session object in the Call object's Session collection (accessed using Call::Sessions). Each new connection request generates a new Call object with one Session object, so there should not be a problem with finding the right session on an incoming call.*

4. *If the user accepted the call, activate the voice channel of the new call. Before you can begin talking over a session, you must first call PhoneSessionClient::SetFocus(Call* call) to activate the channel. This is because the audio engines in the example implementation only allow one voice channel to be active at a time. Once a session has focus, you can start talking over it.*

The following example code demonstrates receiving a call. This code is taken from call-client.cc

```
// Handler connected to PhoneSessionClient::SignalCallCreate on initialization.
void CallClient::OnCallCreate(cricket::Call* call) {
call→SignalSessionState.connect(this, $CallClient::OnSessionState);
}

// Handler called when a Session object changes state or is first created.
void CallClient::OnSessionState(cricket::Call* call,
cricket::Session* session,
cricket::Session::State state) {
if (state == cricket::Session::STATE_RECEIVEDINITIATE){
// This is an incoming call.  Alert the user of the caller's JID.
buzz::Jid jid(session→remote_address());
console_→printf("Incoming call from '%s'", jid.Str().c_str());

// In this thread or in a new thread, request user input to accept or reject the call.
// Input not shown.
...  bool acceptCall = user input value;

// Accept or reject the call.
// The call will be the first item in the Call::Sessions() collection.
if(acceptCall){
call_→AcceptSession(call_→sessions()[0]);
phone_client()→SetFocus(call_);
}
else{
call_→RejectSession(call_→sessions()[0]);
}
}
...  other session states ...
}
```

Terminating

1. *When you are connected, you can mute or unmute a call with the Call::Mute method or terminate a session with the Call::Terminate method. (Some methods exposed by*

Call apply to specific sessions, while others apply to all sessions. See the reference documentation for details.)

Skype

callSetup()	↔	Call(id)
callRequest()	↔	get Call (property) where property status = RINGING and type = Incoming_p2p
callAccept()	↔	set Call ⟨id⟩ status INPROGRESS
callReject()	↔	set Call ⟨id⟩ status REFUSED
callTeardown()	↔	set Call ⟨id⟩ status FINISHED
callTeardownAck()	↔	get Call ⟨id⟩ status FINISHED

Table 8.5: Mappings between Skype API and the Gateway

The id in table 8.5 are the source or destination.

This sections shows the objects Call, Get call, and set call [78] [79].

CALL

Syntax

- *CALL ⟨target⟩[, ⟨target⟩]**

Response

- *CALL ⟨call_ID⟩ ⟨status⟩*

Parameters

- *⟨target⟩ - targets to be called. In case of multiple targets conference is created. Available target types:*
- *USERNAME - Skype username, e.g. "pamela", "echo123"*
- *PSTN - PSTN phone number, e.g. "+18005551234", "003725555555"*
- *SPEED DIAL CODE - 1 or 2 character speeddial code*

GET CALL

Syntax

- *GET CALL ⟨id⟩ property*

Response

- *CALL ⟨id⟩ property ⟨value⟩*

Parameters and response values

- ⟨*id*⟩ - *call ID (numeric);*
- *property - property name. Refer to CALL object for the list of properties.*

SET CALL INPROGRESS

This enables you to resume a call, for example after placing it on hold. Syntax:

- → *SET CALL* ⟨*id*⟩ *STATUS INPROGRESS*
- ← *CALL* ⟨*id*⟩ *STATUS INPROGRESS*

Parameters:

- ⟨*id*⟩ - *call ID (numeric)*

The following statuses are available:

- *UNPLACED - call was never placed.*
- *ROUTING - call is currently being routed.*
- *EARLYMEDIA - with pstn it is possible that before a call is established, early media is played. For example it can be a calling tone or a waiting message such as all operators are busy.*
- *FAILED - call failed - try to get a FAILUREREASON for more information.*
- *RINGING - currently ringing.*
- *INPROGRESS - call is in progress.*
- *ONHOLD - call is placed on hold.*
- *FINISHED - call is finished.*
- *MISSED - call was missed.*
- *REFUSED - call was refused.*
- *BUSY - destination was busy.*
- *CANCELLED (Protocol 2).*
- *VM_BUFFERING_GREETING - voicemail greeting is being downloaded.*
- *VM_PLAYING_GREETING - voicemail greeting is being played.*
- *VM_RECORDING - voicemail is being recorded.*
- *VM_UPLOADING - voicemail recording is finished and uploaded into server.*
- *VM_SENT - voicemail has successfully been sent.*

- *VM_CANCELLED - leaving voicemail has been cancelled.*
- *VM_FAILED - leaving voicemail failed; check FAILUREREASON.*

The following types are available:

- *INCOMING_PSTN - incoming call from PSTN.*
- *OUTGOING_PSTN - outgoing call to PSTN.*
- *INCOMING_P2P - incoming call from P2P.*
- *OUTGOING_P2P - outgoing call to P2P.*

Example to answer and end a call:

```
/**
End a CALL.
@throws SkypeException when connection is bad.
/
public void finish() throws SkypeException {
setStatus("FINISHED");
}

/**
Answer a ringing CALL.
@throws SkypeException when connection is bad.
/
public void answer() throws SkypeException {
setStatus("INPROGRESS");
}
```

ICQ

callSetup()	↔	createRvSession(), sendRv(voiceReqRvCmd(destination))
callRequest()	↔	sendRv(voiceReqRvCmd(source))
callAccept()	↔	sendRv(voiceReqRvCmd(source))
callReject()	↔	sendRv(voiceRejectRvCmd(rejectioncode))
callTeardown()	↔	sendRv(voiceRejectRvCmd(rejectioncode))
callTeardownAck()	↔	sendRv(voiceRejectRvCmd(rejectioncode))

Table 8.6: Mappings between ICQ API and the Gateway

The rejectioncode in Table 8.6 is a public static final int. The rejectioncode CANCELLED = 1, other values are (not yet) defined.

net.kano.joscar.rvcmd.voice shows the JOSCAR API [76] with the information about voice commands.

Since AOL is the owner of ICQ, ICQ uses OSCAR as the underlying protocol. ICQ has its own API, but did not respond to our request to make it available to us. We have found another API for ICQ, named JOSCAR [76]. JOSCAR is a library for connecting to AOL Instant Messenger from Java, and thus it is possible to connect to ICQ.

The interface RvSession represents a single "rendezvous session." Briefly, every rendezvous command sent between two buddies contains a "session ID" which is used to group a set of commands logically. Each of the request, acceptance, and rejection commands are rendezvous commands, and each command contains the same "session ID" as the initial file send request Alice sent.

These "sessions" are never formally created or ended on the OSCAR protocol level, but the author of JOSCAR decided creating a class for it would be the best way to do it. Thus, to send a rendezvous using a RvProcessor, first one must create a new and then send the RV commands through that.

Because OSCAR does not use a sessions [77], formally no sessions are created and thus no sessions can be ended. Nevertheless, it is possible to 'create' a session with createRvSession and send a callSetup request using voiceReqRvCmd. To end the call the object voiceRejectRvCmd can be used. This is also used to reject the incoming call. Creating this object let the listeners know the 'call session' is ended.

Creating the call:

```
cmdMap.put("voicechat", new CLCommand() {
public void handle(String line, String cmd, String[] args) {
RvSession session = bosConn.rvProcessor.createRvSession(
args[0]);

session.addListener(bosConn.rvSessionListener);

try {
session.sendRv(new VoiceReqRvCmd(0,
RvConnectionInfo.createForOutgoingRequest(
InetAddress.getLocalHost(),
-1)));
} catch (UnknownHostException e) {
e.printStackTrace();
}
}
});
```

Accept the call invitation:

```
/**
A rendezvous command used to accept a voice chat invitation.
/
public class VoiceAcceptRvCmd extends AbstractAcceptRvCmd {
/**
Creates a new voice chat session acceptance command from the given
incoming RV ICBM.

@param icbm an incoming voice chat acceptance RV ICBM command
/
public VoiceAcceptRvCmd(RecvRvIcbm icbm) {
super(icbm);
}

/**
Creates a new outgoing voice chat invitation acceptance command.
/
public VoiceAcceptRvCmd() {
super(CapabilityBlock.BLOCK_VOICE);
}
}
```

Reject or end the call:

```
/**
A rendezvous command used to indicate that a voice chat invitation has been
denied or cancelled.
/
public class VoiceRejectRvCmd extends AbstractRejectRvCmd {
/**
Creates a new voice chat invitation rejection command from the given
incoming voice chat invitation rejection RV ICBM.

@param icbm an incoming voice chat invitation rejection RV ICBM command
/
public VoiceRejectRvCmd(RecvRvIcbm icbm) {
super(icbm);
}

/**
Creates a new outgoing voice chat invitation rejection command with the
given rejection code.

@param rejectionCode a rejection code, like {@link ♯REJECTCODE_CANCELLED}
/
public VoiceRejectRvCmd(int rejectionCode) {
super(CapabilityBlock.BLOCK_VOICE, rejectionCode);
}
}
```

Summary

With the provided functions in this chapter, it is possible to show the mappings between the three VoIP systems. The arrow shows the direction the data is going. The Gateway translates the function of VoIP system A to its own function and translates it again to the function of VoIP system B. Notice that the Gateway transforms the incoming callRequest to a callSetup and translates the callRequest function to the function of VoIP system B (and vice versa).

Table 8.7 shows the mappings between Google Talk, the Gateway and ICQ (and vice versa). GTalk Proxy B GW Proxy A ICQ

GTalk	Proxy B	GW	Proxy A	ICQ
→	CallClient::OnSessoionState(call, session, STATE_RECEIVED_INITIATED)	callRequest → callSetup	sendRv(voiceReqRVCmd(destination))	→
←	Call::RejectSession()	callReject	sendRv(voiceRejectRvCmd(rejectioncode))	←
←	Call::AcceptSession()	callAccept	sendRv(voiceAcceptRvCmd())	←
→	Call::Terminate()	callTeardown	sendRv(voiceRejectRvCmd(rejectioncode)	→
←	Call::Terminate()	callTeardown Ack	sendRv(voiceRejectRvCmd(rejectioncode))	←
←	Call::Terminate()	callTeardown	sendRv(voiceRejectRvCmd(rejectioncode))	←
→	Call::Terminate()	callTeardown Ack	sendRv(voiceRejectRvCmd(rejectioncode))	→
←	Call::InitiateSession(buddy_jid)	callRequest ← callSetup	sendRv(voiceReqRvCmd(source))	←

Table 8.7: Mappings between GTalk, the GW and ICQ

Table 8.8 shows the mappings between Google Talk, the Gateway and Skype (and vice versa).

GTalk	Proxy B	GW	Proxy A	Skype
→	CallClient::OnSessoionState(call, session, STATE_RECEIVED_INITIATED)	callRequest → callSetup	Call(id)	→
←	Call::RejectSession()	callReject	set call(id) status REFUSED	←
←	Call::AcceptSession()	callAccept	get call(id) status IN-PROGRESS	←
→	Call::Terminate()	callTeardown	set call (id) status FINISHED	→
←	Call::Terminate()	callTeardown Ack	get call (id) status FINISHED	←
←	Call::Terminate()	callTeardown	set call(id) status FINISHED	←
→	Call::Terminate()	callTeardown Ack	get call (id) status FINISHED	→
←	Call::InitiateSession(buddy_jid)	callRequest ← callSetup	callSetup get Call (property) where property status = RINGING and type = Incoming_p2p	←

Table 8.8: Mappings between GTalk, the GW and Skype

Table 8.9 shows the mappings between Skype, the Gateway and ICQ (and vice versa).

As we read in the last sections, GTalk creates a session when setting up a call. ICQ does not create a session, but JOSCAR does, which means a session is setup and over this session the call is created. Skype sets up a session before the first contact starts. So, the session already exists and it only has to create a call. The sessions and calls are not in detail equal, but the idea is the same.

Skype	Proxy B	GW	Proxy A	ICQ
→	Get Call (property) where property status = RINGING and type = Incoming_p2p call-Request	callRequest → callSetup	sendRv(voiceReqRVCmd(destination))	→
←	set call(id) status REFUSED	callReject	sendRv(voiceRejectRvCmd(rejectioncode))	←
←	get call(id) status IN-PROGRESS	callAccept	sendRv(voiceAcceptRvCmd())	←
→	set call (id) status FINISHED	callTeardown	sendRv(voiceRejectRvCmd(rejectioncode))	→
←	get call (id) status FINISHED	callTeardown Ack	sendRv(voiceRejectRvCmd(rejectioncode))	←
←	set call(id) status FINISHED	callTeardown	sendRv(voiceRejectRvCmd(rejectioncode))	←
→	get call (id) status FINISHED	callTeardown Ack	sendRv(voiceRejectRvCmd(rejectioncode))	→
←	Call(id))	callRequest ← callSetup	sendRv(voiceReqRvCmd(source))	←

Table 8.9: Mappings between Skype, the GW and ICQ

8.4.3 Buddy search

Figure 8.5 and 8.6 show the "Translate message" and "Translate buddy" steps and "Create Proxy A" and "Create Proxy B". The moment Client A sends a buddy search command (the client sends a message with the command buddy search in it), the Proxy gets this message and translates it to an event. The Gateway starts a new Proxy A and this new Proxy performs a buddy search request to its VoIP system (which is the buddy VoIP system of the current User). The found buddy is added to the contact list of Proxy A and the Gateway starts a new Proxy B as an representation of Client B (The buddy Client). When Client A performs a buddy search requets for Proxy B and Proxy B is added to its contact list, the communication can start.

8.4.4 Messaging

Figure 8.7 shows the "Translate message" steps. The moment Client A sends a message to Proxy B, Proxy B translates the message data stream into a text-readable strings and hands it over to the 'Message forwarding' Interoperable Interpreter. This Interoperable Interpreter translates the text-readable strings into the message data stream according to the rules of VoIP system R.

8.4.5 Audio forwarding

As can be read in Chapter 5, Gizmo provides VoIP interoperability between Yahoo, GTalk and MSN. Also GTalk-to-VoIP provides interconnection between Yahoo, GTalk and MSN converting the call setup and tear down to SIP. Furthermore, Skype can be converted to SIP and vice versa. SIP is not used for the audio itself, but also the audio transfer can be converted using these related work systems. This proves converting is possible between MSN, Yahoo, GTalk and Skype. Unfortunately ICQ is not supported by these related work systems and thus there is no prove ICQ can be interoperable with Yahoo, GTalk, MSN and Skype as far as audio converting is concerned. The related work systems do not explain exactly how they work. So in the following paragraphs, the own design idea is discussed again.

The provided design can be extended at several abstraction levels. For instance, now a Proxy has a plug-in on top of it, but is could also be possible to situate a Gateway at the underlying protocol, before the data stream reaches the Proxy. This creates the possibility to directly use the data streams (the packets), in stead of the service. It is even possible to have this solution for one VoIP system and another solution for other VoIP system.

This leads to two possible solutions for the audio forwarding:

1. Forward data stream
2. Virtual audio cable

For the first solution, the easiest way is when the protocol used to transmit the data stream from Client A is the same as the protocol used by Client B. Furthermore, by the setup of the call, the codec preferences should be indicated. If the codecs are the same, the data stream can be directly forwarded.

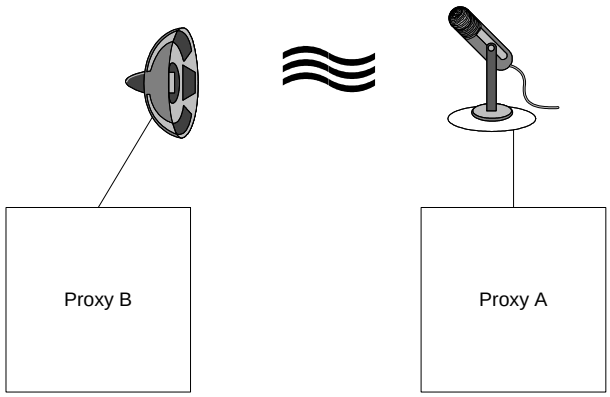


Figure 8.10: Possible way of audio forwarding

If the protocol used for the data stream of Client A is not the same as the protocol used by Client B, the data stream should be converted. This is only possible if the protocol is understandable. Some VoIP systems have proprietary protocols, so this will not be a suitable solution.

The second solution for the audio forwarding is using the Proxies with their plug-ins. This can be sketched at the following way: The incoming audio at Proxy B will sound at a speaker. Next to the speaker a microphone is situated. This microphone is connected to Proxy A, which now converts it according its protocol and sends the audio to Client B. Figure 8.10 shows this sketch.

Of course, the scenario with microphones and speakers is not scalable and thus not usable. In stead, a virtual audio cable [51] can be used. This virtual audio cable transfers audio streams from one client (i.e. Proxy B) to another (i.e. Proxy A). It creates a set of "Virtual Cables" each of them consists of a pair of the waveform input/output devices. Any client can send audio stream to an output side of a cable, and any other client can receive this stream from an input side. All transfers are digitally, providing no sound quality loss. Figure 8.11 shows the Virtual Audio Cable audio forwarding.

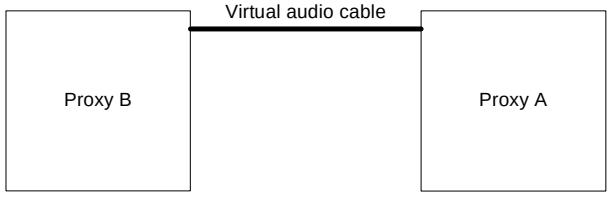


Figure 8.11: Audio forwarding by a Virtual Audio Cable

Many VoIP systems already use virtual audio cables. VoIP phones also provide this kind of gateways (for example to forward audio between the Skype client and the phone).

Figure 8.8 shows the "Translate audio" steps. The moment Client A sends the audio to Proxy B, Proxy B uses the virtual audio cable to translate and tunnel the audio to Proxy A.

8.5 Conclusion

The advantage of this design is that it does not matter if the VoIP systems use peer-to-peer connections or not. It does not matter what is in between the client and the Proxy, and which protocols are used. As long as the Proxy and its plug-in converts the protocol messages into services, the Gateway can handle it. At this way ICQ (not P2P) can be integrated with the other VoIP systems (P2P). The Interoperable VoIP Gateway is thus protocol independent.

By implementing this system, the structure of the protocol messages are not important; the offered services (events) are important. Only text messages and audio has to be converted directly.

To do the buddy search, a command is sent to the Gateway and is handled by this Gateway. Because of the command handling functionality of the Gateway, in the future more commands can be added. This means the Gateway is extendable.

The functional requirements as presented at the beginning of this chapter are all fulfilled by this design. The Gateway is not needed for the login and thus the user logs in to its own server. Users are able to extend their contact list with buddies of other VoIP systems by using Proxies and are able to send those buddies text messages, which are forwarded by the Gateway. The user has to initiate the call; the Gateway is not allowed to initiate a call. Furthermore, only if the user accepts the call, the Gateway can accept the call. Without the acceptance of the buddy, no call can be made.

Plug-in functionality is offered by all VoIP systems. We looked closer to three VoIP systems and their APIs. We can conclude that for Skype, ICQ and GTalk we are able to get data out of the Client and translate it in our own language. Multiple instances are not offered, but with tricks and extensions, multiple instances of all VoIP systems can be arranged. For the audio forwarding virtual cables can be used.

Chapter 9

Conclusion

This chapter concludes the master thesis. It provides the solution (as presented in Chapter 8) and the answers to the research questions (as presented in Chapter 1). One of the research questions is about the requirements, so the requirements (as discussed in Chapter 6) are reflected and discussed. Furthermore we discuss which problems are solved and the (dis)advantages of the solution. Finally future work is presented.

9.1 Solution summary

The solution is called the *Interoperable VoIP Gateway*. It introduces a *Gateway*, which interconnects the existing VoIP systems by using *Proxies*.

The client of one VoIP system is connected to a Proxy (representation of the buddy client). The Gateway is able to interact with this Proxy, translate the incoming data and puts it into another Proxy (the representation of the user client), to create an indirect connection between two clients.

The buddy search is done by sending a text message command to the Gateway; the Gateway will search for the buddy client in another VoIP system and creates the representation of the user client and a representation buddy client. Text messages are sent from the client to the Proxy of the buddy; the Gateway tunnels the text message to the Proxy of the user, which sends the text message to the buddy client.

The call setup and tear down is done by responding to events. For instance, when the client sends a call request, the Gateway responds to this request by creating another call request to the buddy client at the other VoIP system. When the buddy client sends a positive response, the Gateway is allowed to provide a positive response to the originating request. The audio sent from the client to the proxy is tunneled by the Gateway to the other Proxy, which sends it to the buddy client.

9.2 Conclusions per research questions

This section discusses the research (sub)questions as presented in Section 1.3.

What is the state of the art in VoIP?

Chapter 2 provides the history of telephony and VoIP and discusses four major Call Processing Protocols. It seems that SIP is the winner, and this protocol will be even used more in the future. As can be seen, two of the five VoIP systems discussed in Chapter 3 already use SIP (Yahoo and MSN). Although not mentioned in this report before, many other VoIP systems [27] also provide SIP. Also large Telecom operators like KPN for the Netherlands are working on the support of SIP. They are replacing their core-networks to the standardised Next Generation Networking (NGN) architecture called "IP Multimedia Subsystem" (IMS). The IMS supports SIP too.

Because SIP is used more and more, another approach for the design could have been to translate every other protocol to SIP. The Uplink and PSGw software (see Chapter 5) shows it is possible to translate Skype into SIP and vice versa. GTalk-to-VoIP in the same chapter shows that the GTalk protocol can be translated into SIP. MSN and Yahoo already use SIP. This leads to the possibility of interoperability between Skype, Gtalk, MSN and Yahoo by using SIP.

The reason why the approach of translating protocols to SIP is not chosen, is because of the desire to be independent of protocols. Furthermore, building something that already exists is a waste of time. Using PSGw, Uplink or GTalks2VoIP does not support the ICQ system. Furthermore one overall solution is better than several solutions combined.

What are the characteristics, differences and similarities of the VoIP systems and their accompanying protocols?

Chapter 3 and Appendix B shows the overview of five VoIP systems and their protocols. The discussed VoIP systems are MSN, GTalk, Yahoo, ICQ and Skype. The main differences are the protocols used by the VoIP systems. For calls, MSN and Yahoo uses SIP, GTalk uses Jingle, ICQ uses OSCAR and Skype uses Skype.

The five VoIP systems also have some similarities. The main similarities are the services offered by the VoIP system. For example, for a call, first the call has to be setup and accepted by the buddy before the actual conversation can take place. These similarities are also clarified in Chapter 4.

What are the requirements for a prototype to create interoperability between several VoIP systems?

The user requirements are almost all fulfilled. Just one client has to be used, and this can be the one the user is familiar with, which makes it user-friendly. Many features, like messaging and call, are supported. The only not fulfilled requirement is the use of another server. The Gateway buddy that has to be added to support interoperability is the Gateway (server) and thus will be noticed. This Gateway buddy is only needed for sending commands (currently only used for the buddy search and messaging command). The other requirements together weight heavier than this latter one, thus this approach works fine enough.

The Application Provider does not have to change their VoIP systems. The reliability and performance depends on the implementation of the gateway. For now, it seems possible to

provide reliability and performance, as long as these two have high priority in the implementation process. If this works out fine, the AP requirements are also fulfilled.

The interoperability provider requires scalability. Multiple instances of the client of the VoIP system take care of the scalability, although it is not clear whether the solution is scalable enough.

The designers and implementers requires a universal solution, which is done by this design. It supports five commonly used VoIP systems and is probably able to support many others, because it is based on events; it is protocol independent. Only the gateway has to be implemented and not the client, so it is few implementation work. It is easily extendable and has chances to succeed to be reliable, which means the designers and implementers requirements are fulfilled too.

What are the different solutions to create interoperability between the VoIP system?

To find out the best way to design the prototype for the interconnection of several VoIP systems, a design approach has to be defined. Chapter 7 discusses some possible approaches. The first approach is to use the existing VoIP systems and interconnect them with a Gateway. The second approach is to change the client of the existing VoIP systems and led those clients connect to the gateway. The third approach is to build a new VoIP system, which is interconnected to existing VoIP systems with a gateway. The fourth approach is a client that works as a peel around the existing VoIP systems; accounts for all existing VoIP systems are needed, but the VoIP clients do not have to be installed. The fifth approach equals the third, but the client is now a web client.

The first approach is chosen as the most interesting to continue working with. Chapter 8 provides a detailed description of this design solution and concludes it is indeed a good solution.

How is the interconnection modelled and realized?

Chapter 8 presented the architecture of the Interoperable VoIP Gateway. This design is realized by first studying the most commonly used VoIP systems (Chapter 3 and Appendix B) and defining their services (Chapter 4, second by studying related work on interoperability (Chapter 5) third by defining possible approaches (Chapter 7) and selecting the best suitable solution. This selected approach is modelled with sequence diagrams, defining scenarios (Chapter 8).

How can VoIP systems interoperate?

The main research question is already answered in the subquestions. The solution to let VoIP systems interoperate is to design a Gateway and use Proxies.

9.2.1 Solved problems

First we needed to find out the equalities and differences of the VoIP systems. We chose five commonly used VoIP systems (MSN, Yahoo, GTalk, ICQ, Skype), with Wireshark ex-

periments we sniffed the packets and we provided sequence diagrams of four features (Login, Buddy search, Messaging and Call).

We found out that it is impossible to understand the proprietary protocol packets. This let us to the conclusion it is not possible to create a solution for all VoIP systems when translating at protocol level. We decided to create a protocol independent solution.

Next, we needed to find out how to translate the data sent to the Proxies. The five VoIP systems provide APIs to create a plug-in to the existing Client. With the APIs it is possible to get data out and put data in the Clients. We looked closer to three VoIP systems and their APIs. We can conclude that for Skype, ICQ and GTalk we are able to get data out of the Client and translate it in our own language. The choice of these three VoIP systems is random. The audio forwarding can be done by virtual audio cables.

For the interconnection, every client has a representation, i.e. Proxy. Actually, it can have multiple proxies, one for each specific VoIP system. This means multiple instances of the client have to run at the Gateway. We found out that it is possible to create multiple instances for the five VoIP systems.

9.2.2 Advantages

Despite of the interconnection, the user can still communicate with a user from the same VoIP system, without using the Gateway. This concludes that it is possible to create interoperability between VoIP systems, at least for the five commonly used VoIP systems as presented in this report.

Furthermore, as already discussed in Paragraph 8.5, the five VoIP systems have the possibility to add plug-ins to handle the events and messages. Multiple instances of the Clients of the VoIP systems are needed, and this is also possible to get done. The audio forwarding can be done by using a virtual audio cable.

The advantage of this design is, that it does not matter what the underlying protocols of the VoIP system are, the solution is protocol independent.

The Gateway is able to handle commands and is now only used for the buddy search. In the future this is extendable for more commands if needed.

9.2.3 Disadvantages

This solution has also some disadvantages. First, for the user it is difficult to add a buddy from a different VoIP system to the contact list. In stead of adding a buddy, the user needs to send a command to the Gateway client. After adding the buddy, the user will not notice the interruption of the Gateway anymore.

Second, this solution needs multiple instances of the clients, to create the Proxies. This makes it unclear whether it is scalable enough.

9.2.4 Future work

This section suggests possibilities to continue the research presented in this thesis.

This thesis consists of the design of the Gateway to interconnect several VoIP systems. To prove the design, we need an implementation.

VoIP systems offer most functionalities for free. The implementation and maintenance is expensive, but is probably compensated with advertisements shown in the clients. A cost analysis of how much would cost the implementation of the Gateway was not addressed in this thesis. Furthermore, a cost analysis of the maintenance of the Gateway was not addressed in this thesis. Nevertheless, such an analysis would provide more information about the Interoperable VoIP Gateway chance of success from an economical point of view.

Appendix A

Additional information: State of the Art in VoIP

This Appendix shows additional information about the on-hook and off-hook operations, more Call Processing Protocols and the INVITE method of SIP.

A.1 On-hook and off-hook operations

Table A.1 shows the on-hook and off-hook operations.[82]

Name	Type	Direction	Meaning
Connect	Off-hook	→	Request service and hold connection
Disconnect	On-hook	→	Release connection
Answer	Off-hook	→	Terminating end has answered
Hang up	On-hook	←	Message complete
Delay start	Off-hook	←	Terminating end not ready for digits
Wink start	Off-hook	←	Terminating end ready to receive digits
Start dialing	On-hook	←	Terminating end ready for digits
Stop	Off-hook	←	Terminating end not ready for further digits
Go	On-hook	←	Terminating end ready for further digits
Idle trunk	On-hook	↔	
Busy trunk	Off-hook	→	

Table A.1: On-hook and Off-hook Operations

A.2 Call Processing Protocols

Four Call Processing Protocols have already be mentioned, H.323, MGCP, Megaco and SIP. Some other Call Processing Protocols are:

- Skinny Client Control Protocol (SCCP): Proprietary protocol of Cisco
- MiNet: Proprietary protocol of Mitel
- CorNET-IP: Proprietary protocol of Siemens
- IAX: Inter-Asterisk eXchange protocol for the open source PBX server Asterisk

- Skype: A proprietary peer-to-peer protocol used for the Skype application
- Jajah: A proprietary peer-to-peer protocol used in Jajah SIP and IAX compatible web-phones
- Jingle: Open peer-to-peer protocol based on XMPP (Jabber) and able to communicate with the Google Talk protocol

A.3 SIP INVITE method

Figure A.1 shows the INVITE method of SIP, and consists of the following events:

- Event 1: User 1 is trying to reach User 2. It sends an INVITE message with the URL of User 2.
- Event 2: The INVITE message is forwarded to the redirect server (Proxy Server) of the first user.
- Event 3: The Redirect Server looks up this URL in its database and asks the Proxy Server to directly contact the User Agent and provides the location information of the User Agent.
- Event 4: The Proxy Server now sends the INVITE message directly to the User Agent.
- Event 5: The User Agent alerts the called party with a ring tone.
- Event 6: The User Agent sends back OK message to acknowledge it.
- Event 7: The Proxy Server sends the OK message to the calling party.
- Event 8 and 9: User 1 sends an ACK message to User 2 (relayed by the Proxy Server).

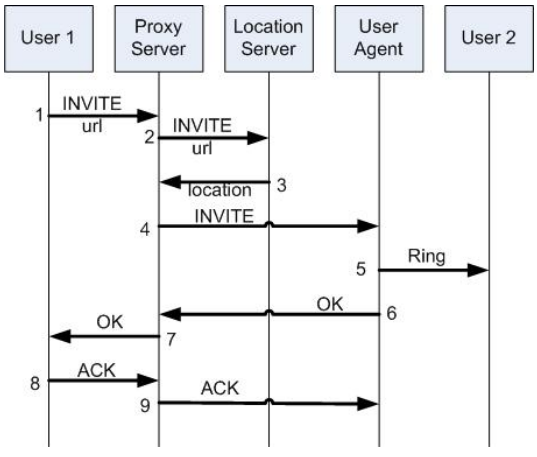


Figure A.1: SIP protocol flows of an INVITE

Appendix B

Additional information: Overview of VoIP systems

This appendix provide additional information about the VoIP systems. It gives a list of less known VoIP systems and the Skype IP domain. Furthermore, it provides additional information about the five VoIP systems used in this report, including the sequence diagrams based on the Wireshark results.

B.1 VoIP systems

Some open source alternatives for Skype are [47]:

- **Ekiga:** A free application that supports both H.323, SIP, audio and video. Ekiga was formerly known as GnomeMeeting. So far only works with various Linux based systems. No version for Microsoft Windows has been released yet, but there is a working snapshot available.
- **Kiax:** VoIP application based on IAX
- **PSI:** The current Beta version has protocol support for Google Talk
- **Switchboard:** Free VoIP applet which works from within a web browser. Works on Windows, Mac, Linux, and any other Java enabled platform. No installation necessary
- **Tapioca:** Includes support for Google Talk
- **WengoPhone:** A free VoIP application based on SIP open standard
- **Freetalk:** A command line jabber client with VoIP support. Has readline support and scheme as an extension language. This project is a part of the GNU Project.

Some closed source alternatives for Skype are [47]:

- **amiciPhone:** A secure peer-to-peer VoIP application
- **Gizmo Project:** A closed source VoIP application based on SIP open standard and uses SRTP between clients. Now offering free landline/cell calls to over 60 countries
- **iCall:** A closed source free VoIP application based on SIP open standard and providing free PC to Phone calling in the US and Canada.

- **Jajah:** Alternative where no headset, no download, no installation and no broadband connection is necessary. A VoIP call gets activated between two normal phones.
- **Secure Shuttle Transport (SST):** Free encryption and secure messaging software including VoIP and video. Works on PCs running Windows 98 or higher.
- **SightSpeed:** Free video and voice calling service supporting Mac and Windows. Also allows phone out and in calling.
- **Parlino:** A VoIP network based on open standard SIP-protocols, launched by Parlino S.A.
- **Vbuzzer:** A VoIP softphone and service as well as an active advocator of SIP open standard
- **VoIP Buster:** A VoIP application offering 300 minutes per week of free calls to landlines in many countries, including the EU, USA, Australia, etc.
- **VoIP Stunt:** A VoIP application offering 300 minutes per week of free calls to landlines in many countries, including the EU, USA, Australia, etc.
- **Zfone:** is a solution of Phil Zimmermann (inventor of PGP) to encrypt VoIP (SIP) sessions, protocol published as IETF draft.
- **TipicIM:** A free VoIP application, Videocalling based on XMPP/Jabber and Speex audio codec support
- **ClosedTalk:** is secure VoIP software free for Business/Personal use. Works on PCs running Windows 2000/XP. [ClosedTalk exposes 'man in the middle' attacks by displaying a short security message on both caller screens for comparison.
- **BT Communicator:** is a VOIP service from British Telecom (BT plc.)

Furthermore two multi-protocol applications exist: Trillan and Gaim. These applications include other application protocols and make it possible to have send messages to applications of different protocols.

B.2 Skype IP domain

Skype network According to RIPE, next ranges of IP addresses belong to the Skype company:

Skype-NL
212.72.49.128 - 212.72.49.159
213.244.170.64 - 213.244.170.127

Skype (LU)
80.92.83.128 - 80.92.83.255

According to ARIN, next range of IP addresses belongs also to the Skype company:

Skype (LU)
67.72.71.48 - 67.72.71.63

B.3 MSN

The protocol MSN uses for sending messages and control information is called Mobile Status Notification Protocol (MSNP). MSNP works over TCP and is an ASCII-based protocol and uses a series of UTF-8 encoded commands for the communication between servers and clients. These commands and their parameters are always transmitted as plaintext (i.e. unencrypted), and are displayed as human readable text. [91] The commands start with three characters, and are followed by the actual information regarding the command that is being sent. Examples of commands are TFR (transfer), USR (user), and CHG (change). One TCP/IP packet can have more than one command. The commands are separated using carriage-return/linefeed (CRLF, \r \l). [40]

Login

MSN uses the Mobile Status Notification Protocol (MSNP) to connect to the .NET Messenger Service. The .NET Messenger Service offers its service on port 1863 of messenger.hotmail.com. Microsoft disclosed version 2 (MSNP2) to developers in 1999 in an Internet Draft [91], but did not release newer versions to the public. For Windows Live Messenger MSNP13 has been used.

By default the MSN servers are waiting on port 1863 for clients to connect. If this port is blocked, clients will attempt connections on the very common destination port HTTP (80). MSNP is able to tunnel IM traffic via HTTP by embedding the packets into an HTTP POST request; this causes the packets to pass through security measures such as firewalls. [67]

Users attempting to access the MSN network must first contact the DS to request login, which then redirects the user to the .NET Passport Nexus Server for authentication. Communication with the DS occurs in plaintext and contain information such as the account name and the operating system and hardware platform. Communication with the Nexus Server occurs over an encrypted Secure Socket Layer (SSL) HTTPS connection, to prevent the password from being transmitted in plain text. If authentication is successful, the client may connect to the NS.

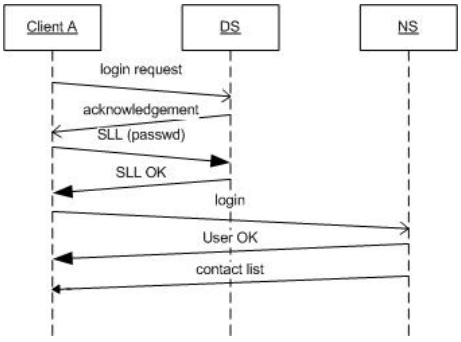


Figure B.1: Login sequence diagram

Figure B.1 shows the sequence diagram of the login process. Client A requests to login and the DS confirms. Client A will use the Secure SSL connection to send the password and the

DS responds over this secure SSL connection to confirm the login. Now client A can contact the NS. The NS will respond with an acknowledgement and the contact list.

During a connection to the MSN presence service, the client will synchronize now and then the data with the server. These data include account names of all contacts and their phone numbers. The synchronization process occurs in plaintext. After a successful authentication these kind of information is also synchronized with the server.

Buddy search
To expand the contact list, it is possible to search for a buddy. Figure B.2 show the sequence

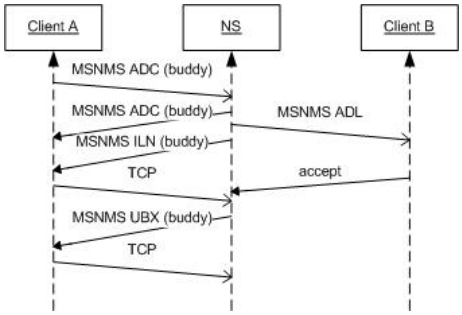


Figure B.2: Buddy search sequence diagram

diagram of the buddy search. First it sends a request to the NS with the msn name of the buddy to find. The NS sends a request to Client B to ask for permission. If Client B accepts, an acknowledgement will be send to the NS and now the NS is allowed to send the new buddy data to Client A, which is done by sending a new contact list. Messages are send by the MSN Messenger Service (MSNMS) protocol.

Messaging

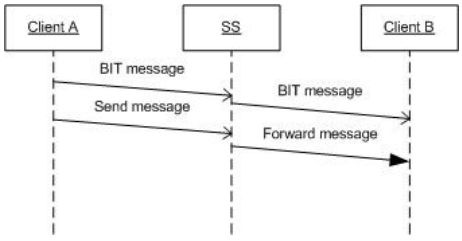


Figure B.3: Messaging sequence diagram

Sending a text message occurs in plaintext. Messages are sent to the SS, the MSNP messaging server.

Figure B.3 shows the sequence diagram of text messaging. Client A sends a message to the SS, which forwards the message to Client B. Client B might respond with an own message, and sends it to the SS, which forwards it to Client A again.

Filetransfer

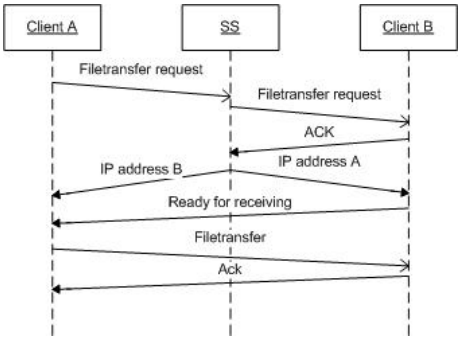


Figure B.4: File transfer sequence diagram

Figure B.4 shows the sequence diagram of file transfer. If Client A wants to send a file, it has to sent a request to the SS. The SS forwards this request to Client B. If client B accepts, it sends an acknowledgement to the SS. Now the SS is allowed to distribute the IP addresses to both clients, so they can setup a peer-to-peer connection. When client B receives this message with the IP address of Client A included, it will send a message to Client A to inform it is ready to receive. Client A will receive this message directly form Client B, because it has received the IP address of Client B. Client A will now send the file and Client B will acknowledge it afterwards.

Call

Figure B.5 shows the sequence diagram of a call. If Client A wants to have a conversation with Client B, it has to send a request to the SS, which forwards it to Client B. If Client B accepts, the SS is allowed to distribute the IP addresses to both clients. Client B sets up a SIP connection. It sends an invite to Client A, which responds with a SIP Trying, Ringing and OK message. If everything is okay, Client B responds with an acknowledgement. Some RTCP data will be send to maintain the connection and the conversation between the two clients is possible. To end the call, Client A sends a SIP Bye and a RTCP Goodbye. Client B acknowledges the end of the call by a SIP OK and a RTCP Goodbye.

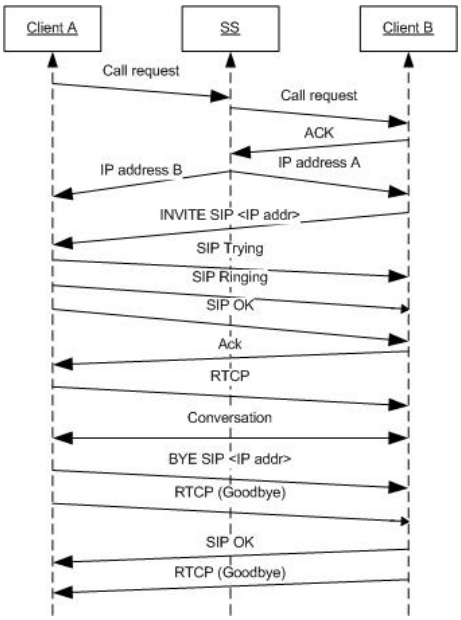


Figure B.5: Call sequence diagram

B.4 GTalk

Google Talk uses the Extensible Messaging and Presence Protocol (XMPP) and thus this protocol will be described in this paragraph. XMPP is defines in the XMPP core (RFC 3920) and XMPP IM (RFC 3921) specifications by the Jabber Software Foundation to the Internet Standards Process, which is managed by the Internet Engineering Task Force (IETF) in accordance with RFC 2026. [88]

For VoIP, video and other P2P multimedia, GTalk uses Jingle. Furthermore GT claims to support SIP in a future release.

A GTalk (XMPP) message may possess the following attributed [36]:

- **To:** Specifies the intended recipient of the message
- **From:** Specifies the sender of the message
- **ID:** An optional unique identifier for the purpose of tracking messages
- **Type:** An optional specification of the conversational context of the message
- **Body:** The textual contents of the message
- **Subject:** The subject of the message
- **Thread:** A random string that is generated by the sender and may be copied back in replies; it is used for tracking a conversation thread
- **Error:** An error code

Login
A Jabber client connects to a Jabber server on a TCP socket over port 5222. This connection is "always-on" for the life of the client its session. [23] [25]

When a client connects to a server, it opens a one-way XML stream from the client to the server, and the server responds with a one-way XML stream from the server to the client. The advantage of the use of XML streams is the low bandwidth-use caused to the exchange of simple and small lines of XML.

Figure B.6 shows the sequence diagram of the login process. Client A starts a secure SSL connection to the Login Server with a 'Client Hello' message to announce itself. The Login server responds with the secure TLS message and sends a certificate. Client A and the Login server will exchange encryption information, like key and afterwards Client A is able to connect to the Contact server by sending a Jabber request and receiving a Jabber response. Jabber messages are in XML.

Buddy search
Client A sends a Jabber request to the Contact Server, which responds with a Jabber response. Client B is not asked for acceptance, but is just added to the contact list of Client A. This is the main difference with the buddy search process as presented in Figure 3.4, where Client B has to accept. Furthermore a user does not receive a list of potential buddies while using GTalk.

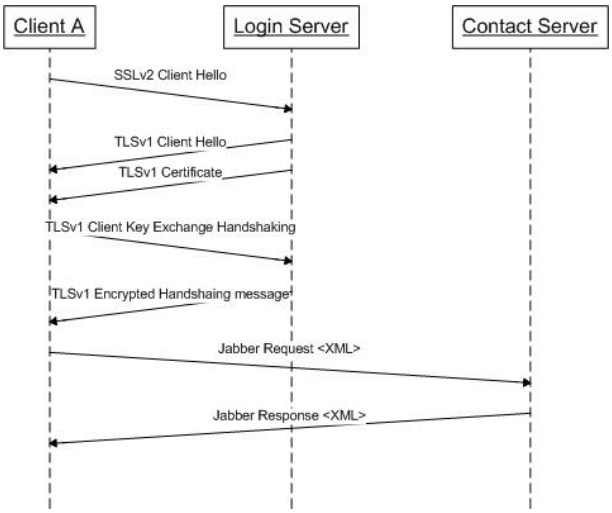


Figure B.6: Login sequence diagram

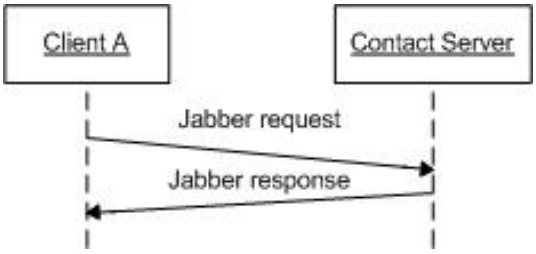


Figure B.7: Buddy search sequence diagram

does not have to accept the invitation to join the contact list.

Messaging
Figure B.8 shows the messaging sequence diagram. When Client A wants to send a message to Client B, it sends a Jabber request (the message) to the Contact server, which forward it to Client B. Client B acknowledges this message to the Contact Server, which forwards it to Client A. Client B might send a message back (via the Contact Server) to Client A, which responds (via the Contact Server) to Client B with an acknowledgement.

Filetransfer

Figure B.9 shows the sequence diagram of file transfer. First, Client A sends a Jabber request to the call server to request for file transfer. The Call server forwards this request to Client B, which might accept it (and respond to Client A via the Call server). Both clients do a HTTP get to the Call server and get a HTTP OK respond. To avoid NAT problems, STUN binding messages are sent to the Call server and between the two clients. Afterwards Client A is allowed to send the file to Client B, which responds with an acknowledgement.

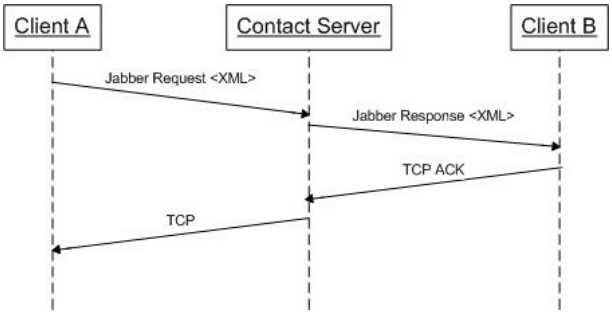


Figure B.8: Messaging sequence diagram

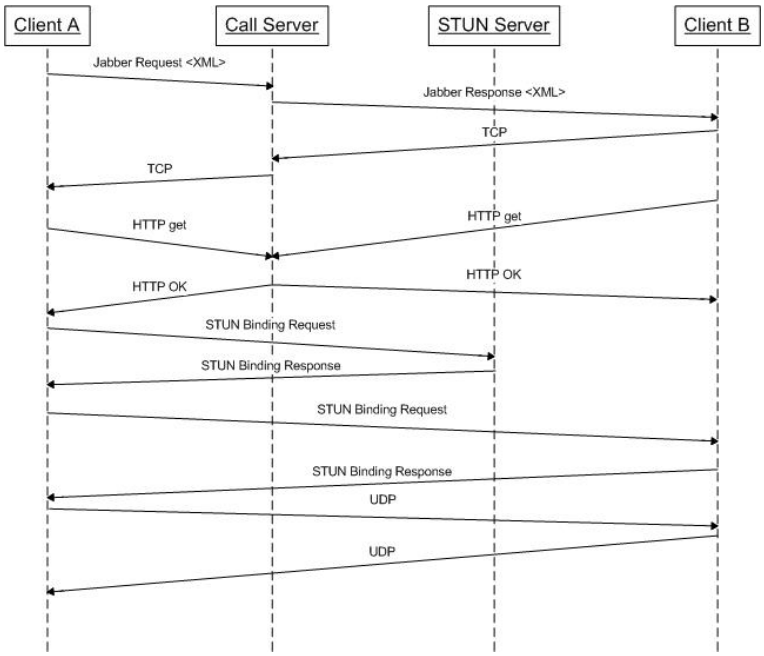


Figure B.9: Filetransfer sequence diagram

Call

XMPP has an extension to make it possible to do more then only IM, like video and audio communication. This extension is called Jingle. [88] [89]

With the extension Jingle it is possible for Jabber to create a Peer-to-peer connection to make voice communication possible. The Jabber Software Foundation published on the 15th of December 2005 the specification about "Jingle Signaling" and the specification "Jingle Audio" for VoIP. The same day Google published the source code for the library of Jingle: "libjingle" to implement the specification. [24]

Libjingle also specifies the RTP protocol for transmitting and receiving media across the network. RTP was improved by the Libjingle developers, STUN and ICE protocols were implanted to support a higher interoperability through various NAT and firewall configurations.

[52] found out that GTalk uses the PCMU/G.711u codec by default, but many more codecs are implemented, like GSM and iLBC. Google claims they are going to support a series of Speex codecs, which have a good voice quality with good bit rate and are license and patent free.

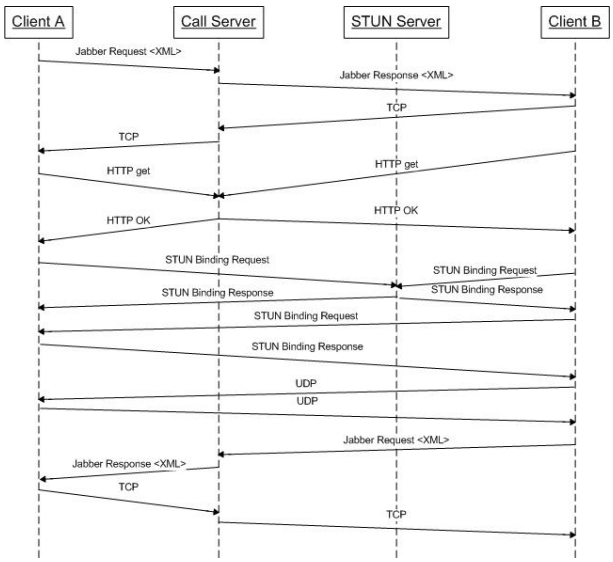


Figure B.10: Call sequence diagram

Figure B.10 shows the sequence diagram of a call. It is almost the same as Figure B.9. Client A sends a request for call, Client B accepts, HTTP get and OK messages are exchanged and STUN Binding messages are exchanged. The difference between call and filetransfer is that for the call both Client A and B needs to send STUN Binding messages to the STUN server and that Client B sends a STUN Binding request to Client A, in stead of Client B sending to Client A.

Afterwards it is possible to have the voice conversation. If Client B ends the call, it will send

a request through the Call server to Client A, which acknowledges this request. Of course, Client A is also allowed to end the call.

B.5 Yahoo

YMSG is the Yahoo! Messenger Protocol. Some parts of YMSG rely on other protocols. For example, file transfer is initially negotiated using YMSG, but the actual transfer of the file is done via HTTP. Regular webcam connections use H.323 and a PC-to-PC call uses SIP. The calls are handled indirectly by the Yahoo! Servers, so the client does not have direct access to it. [62]

A YMSG packet starts with the letters YMSG. If the client sends the packet to the server, the byte that follows is the version of the protocol. If the server sends it, the following byte is zero.

Appendix Aoverview describes the header for YMSG packets.

Login

YMSG communicates between clients and the server, using a TCP connection to cs.yahoo.com on port 5050 by default. Other ports may be used if this port is blocked. If other ports are also blocked, the alternative is a HTTP route.

The client remains logged in for as long as the TCP connection is still open. In case of the HTTP connection, the client remains logged in until the client fails to send a request for some time ("ping" messages are sent about every thirty seconds). [63]

The login for YMSG is quite complex. First, the client introduces itself with a message containing its username. The response of the server is a 24-character string. In return, the client puts this into an algorithm, along with the password and sends the two 24-caracter strings back to the server. These two strings contain an MF5 crypt hash of the password as well as the encrypted user name. If these values match, the client is authorized and will get the data associated with the account (like the contact list). Once the server verifies the user, a random X-byte cookie is sent back to the client. The client can now use this cookie for various functions, such as checking when new email arrives. Once the cookie is received, the client changes the status to online and is ready for messaging. [87]

Figure B.11 shows the sequence diagram for the login. Client A first sends a message containing the username to the IM Server. The IM server responds to this. Next Client A will connect to the Login Server by sending the username and password. If this is correct, the Login server will respond with an okay message. Now the client is allowed to send a request to the IM server and receives the contact list in response. Furthermore, the Client has to register itself to the SIP server with a SIP REGISTER message. The response of the SIP server will be SIP OK, SIP SUBSCRIBE and SIP NOTIFY, all over the secure SSL connection. Also STUN binding messages are sent between Client A and the STUN server.

Buddy search

Figure B.12 shows the sequence diagram for the buddy search. Client A sends an add buddy request to the IM server. The protocol YMSG is used again. The IM server checks whether

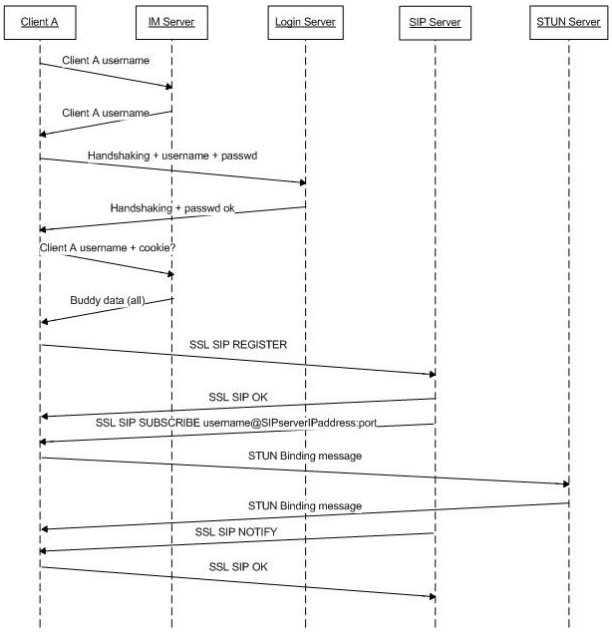


Figure B.11: Login sequence diagram

Client B accepts the invitation. If so, an acknowledgement and the contact information will be send to both Client A and Client B.

Messaging
Figure B.13 shows the sequence diagram for the messaging. Client A has been notified of the IP address before and is now able to send data directly to Client B. The moment the user of Client A starts typing, Client B will be notified of this, and might acknowledge it. When the message is sent, Client A sends a message it is having a chat session to the IM server. Next it is allowed to send the message to Client B. The IM server responds with a chat session ACK. And also Client B responds with an acknowledgement. Client B might send a message back. The moment the user of Client B starts typing, Client A receives a message of this and acknowledges it. When the message is sent, the IM server will be notified.

Filetransfer
Figure B.14 shows the sequence diagram for the filetransfer. If Client A wants to send a file to Client B, it sends a request to the IM Server. The IM server notices both clients about the file transfer. Client A is now allowed to send information about the file via the IM server to Client B. If Client B accepts, Client A sends the file using HTTP and Client B acknowledges it.

Call
Figure B.15 shows the sequence diagram for the call. Yahoo is using SIP with a secure SSL connection. Client A first sends a SIP INVITE via the IM server to Client B. Client B responds with a SIP Trying, SIP Ringing and SIP OK. Furthermore some STUN binding messages are needed. Now Client A is allowed to send the file to Client B, which acknowledges it. Closing the connection is done by SIP Bye messages, with a SIP OK response.

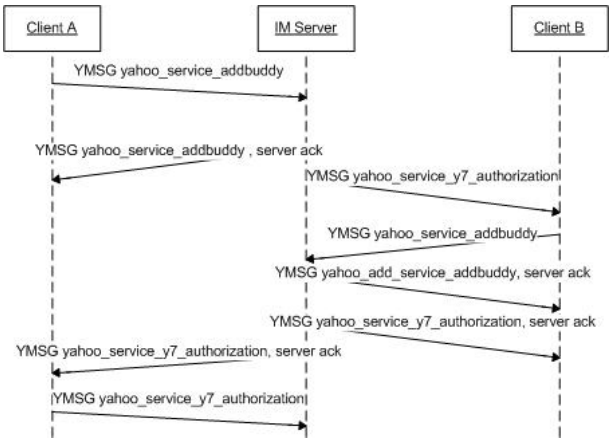


Figure B.12: Buddy search sequence diagram

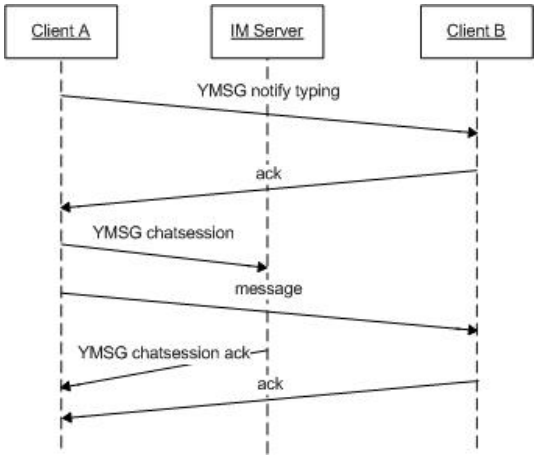


Figure B.13: Messaging sequence diagram

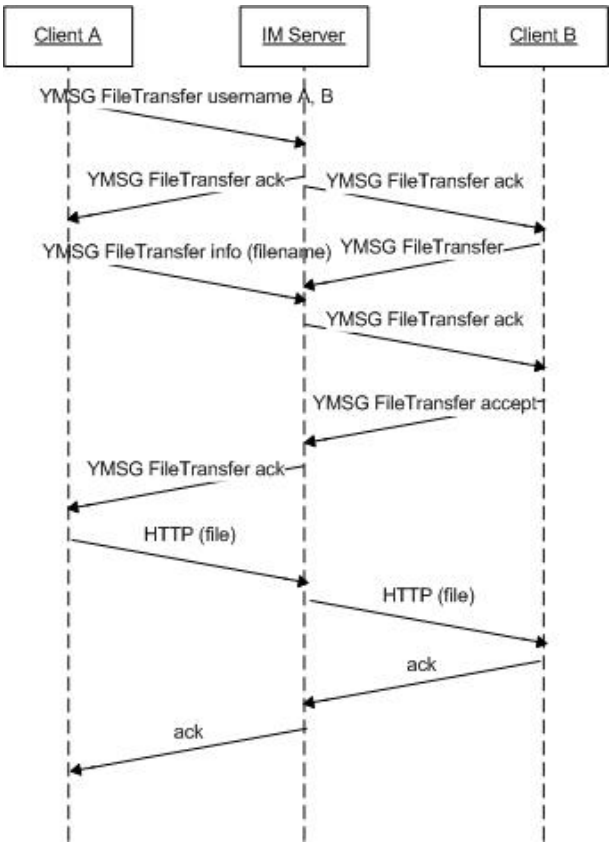


Figure B.14: Filetransfer sequence diagram

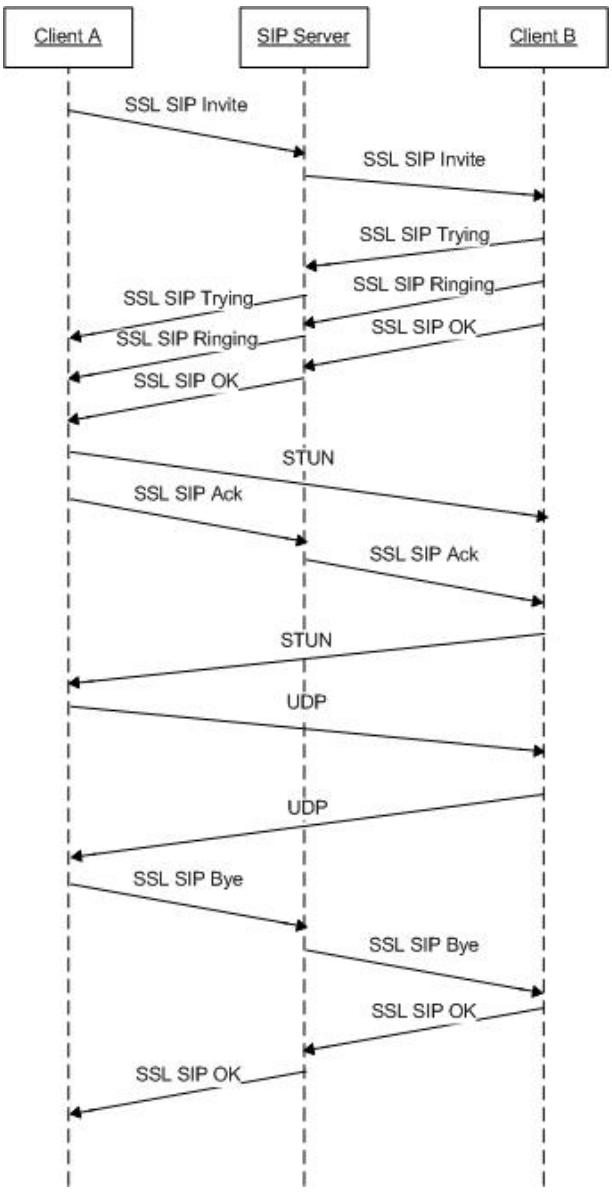


Figure B.15: Call sequence diagram

B.6 ICQ

The protocol OSCAR is used for the communication. OSCAR uses a binary/hexadecimal coded command syntax that is structured using a series of "frames". OSCAR frames are therefore not easily read, as commandos do not translate directly into meaningful text. OSCAR is used by both ICQ and AIM clients. [65]

OSCAR provides two methods for user authentication, referred to as "Channel 0x01" and "MD5-based" authentication. Both methods transmit a portion of information in plaintext, including client version, account name, and the country the user is located. Channel 0x01 authentication encrypts a password using a method called roasting. This method is not secure and an intercepted password can be easily decoded. MD5-based login uses an MD5 authentication key acquired from the login server. This key makes it possible to encrypt the password before sending it to the server. The MD5 authentication key is changed at each login and a password cannot be easily recovered as in the roasting method.

Packets are called FLAPs. Most of the commands are sent in SNAC packets that are contained within the FLAP packets. Each FLAP packet has a FLAP header. Appendix B presents these headers.

Login

During the login, OSCAR will first try to login on login.oscar.aol.com, with the default port 5190. If the default port of an OSCAR client is blocked by a firewall, the clients are able to use any number of ports to tunnel information. If the login.oscar.aol.com is blocked by firewalls, a HTTP proxy server (www.proxy.aol.com) can be used. In this case, an HTTP header will be added to all the packets.

Figure B.16 shows the sequence diagram of the login. The client first connects to the Login Server and sends the screen name and other associated information. The server will ask for the password. The client hashes the password and sends back a packet with the screen name, password, and client version information. If these data is correct, the server will respond with a random X-bit cookie and the IP address of the Basic Oscar Services (BOS) server.

Now the client disconnects from the Login Server and connects to the BOS server. The client will authenticate itself by sending the cookie it received from the Login server. The server also sends a package to the client telling how often it needs to send keep-alive packets to the server. If this rate is disobeyed, the server will invalidate the cookie and the client will be disconnected. If the client disconnects, the cookie will also be invalid.

Buddy search

Figure B.12 shows the sequence diagram for the buddy search. Client A sends a buddy search request to the BOS Server. Furthermore it sends an AIM SSI Grant Future Authorization to Buddy, an Edit Start and Add buddy request. The BOS server acknowledges these messages and Client A is now allowed to send an Authorization Request to the BOS. The BOS forwards this request to Client B, which might accept the invitation to join the contact list. The reply to the BOS will be forwarded to Client A, which will finalize the buddy search.

Messaging

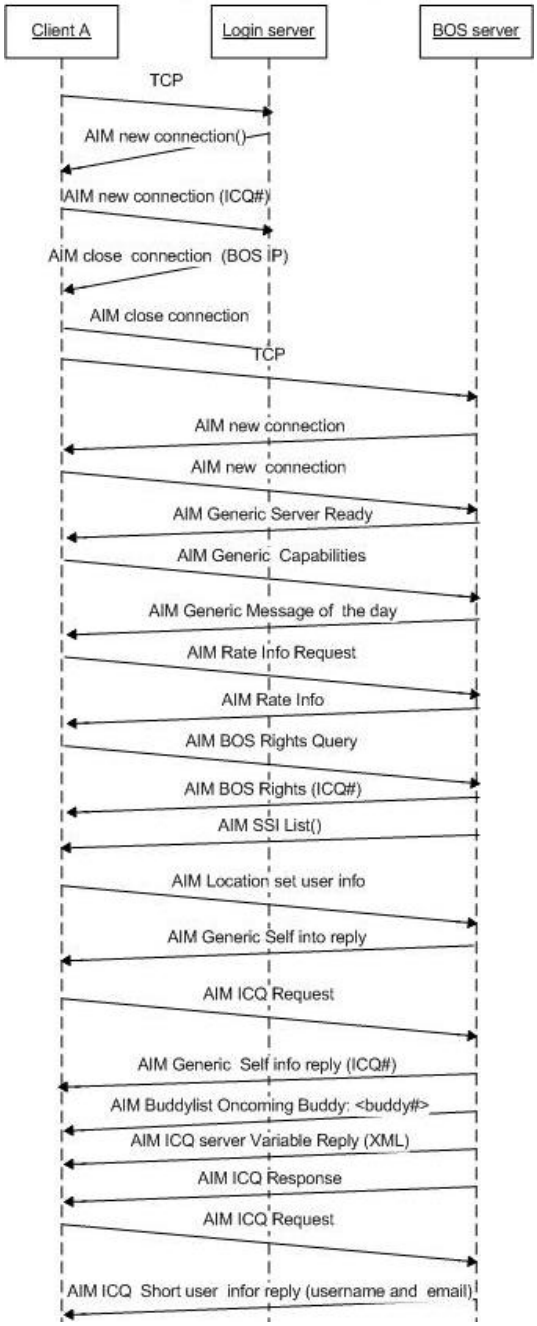


Figure B.16: Login sequence diagram

Figure B.13 shows the sequence diagram for the messaging. The moment the user of Client A starts typing, Client A sends a Mini Typing Notification (MTN) through the BOS Server to Client B, which acknowledges it. Now the real message can be send through the BOS Server to Client B. Client B sends an auto response to acknowledge the incoming message. Client A will receive the acknowledgement from the BOS server. When Client B wants to send an message to Client A, the opposite will happen.

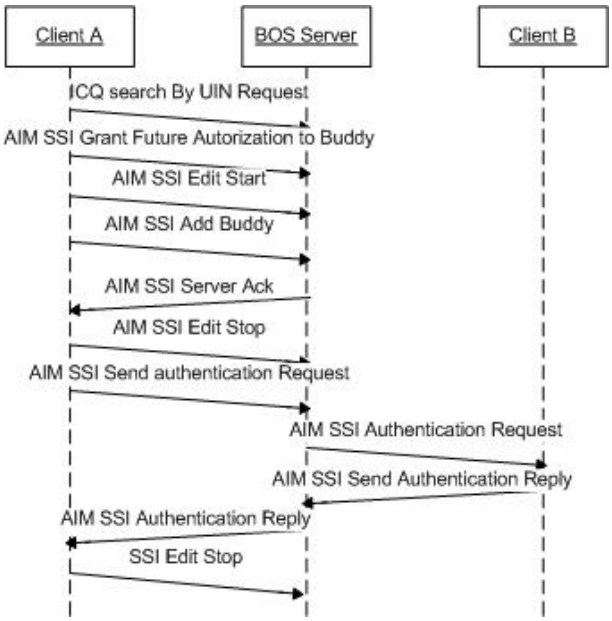


Figure B.17: Buddy search sequence diagram

Filetransfer

Figure B.14 shows the sequence diagram for the file transfer. When Client A wants to send

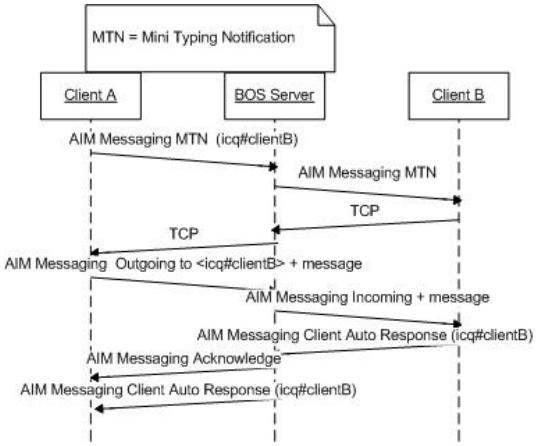


Figure B.18: Messaging sequence diagram

a file to Client B, it first requests Client B via the BOS Server. Client B acknowledges the incoming message. Afterwards Client A sends the file name to the File Server, which forwards this file name to Client B. Client B might accept the retrieval of the file and sends this acknowledgement through the File Server to Client B. When Client A receives this acknowledgement, it is allowed to send the actual file through the File Server to Client B. The file will be acknowledged.

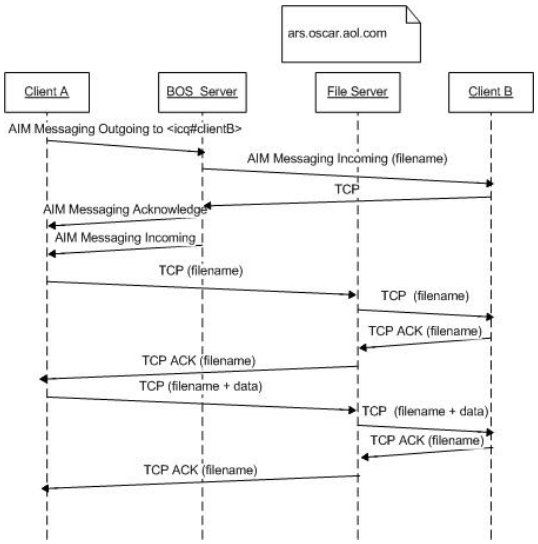


Figure B.19: Filetransfer sequence diagram

ICQ does not use a P2P connection for the file transfer. Furthermore, Client A first sends a message to Client B. Later on, Client A sends the request for the file transfer.

Call

Figure B.15 shows the sequence diagram for the call. When Client A wants to make a call to Client B, it first sends a message to the BOS Server, which forwards it to Client B. Client B acknowledges the incoming message. Afterwards Client A sets up the call through the BOS Server to Client B. Client B might accept the incoming call to let Client A and Client B have a conversation.

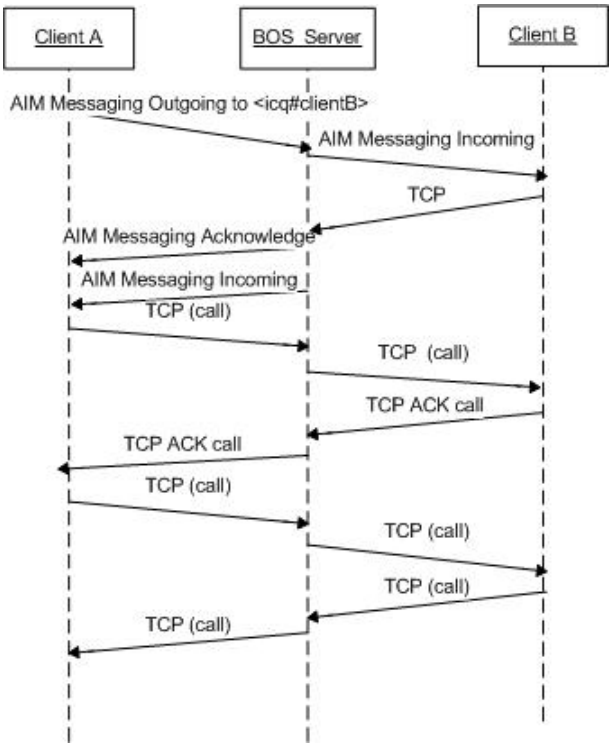


Figure B.20: Call sequence diagram

B.7 Skype

Skype uses its own proprietary protocol, named Skype.

Login

First of all, while trying to connect to the Skype network, the Skype client will search for available Super Nodes. Every node in the network has a list of super nodes, which is refreshed regularly. These list, also called Host Cache and named shared.xml is kept on the local disk (C:\Documents and Settings\username\Application Data\Skype) of the users computer and consists of up to 200 entries of Skype super nodes it knows from previous sessions. [64] As long as the list contains one valid entry (e.g. IP address and port number of an online Skype node), it is able to connect to the Skype network.

When it is the first time the Skype application is used, the Host Cache is empty. Therefore, Baset and Schulzrinne found out in [64] that Skype has some bootstrap super nodes hard coded in the software, which are only used when the client logs on for the first time.

According to [64], Skype is able to solve NAT and firewall traversal issues by using a variant of IETF's STUN protocol. Skype is also able to do hole punching [17], to open a port on the firewall to connect to this port. During the login process, Skype clients automatically informs all other nodes in the P2P networks of its presence and discovers other online nodes as well.

The steps in the process of connecting to a supernode have been figured out by Baset and Schulzrinne [64]. These steps are presented in Figure B.21 and discussed in the following text.

First, the Skype client will pick a super node from the Host Cache and tries to connect. This is done by sending an UDP packet to the IP address and port it finds in the Host Cache. If the node is not responding within five seconds, the node is probably offline and another node from the Host Cache will be used.

However, it is also possible that the node is behind a firewall that blocks all UDP traffic, so on the next attempt to connect to a Super node, TCP will be used.

When even a TCP connection to the addresses and ports of all entries from the Host Cache fails, it will try a TCP connection to port 80 (HTTP-port). The last attempt to get a TCP connection will be on port 443 (HTTPS-port). If all these attempts result in no connection, the Skype client gives up and reports a login failure.

If somewhere in this algorithm the connection to this algorithm is setup, the connection to the Super Node is successful and can continue the rest of the login process.

Figure B.22 shows the sequence diagram for the login. Client A first sends a UDP Request to the Super Node, which responds to it and provide the location of the Login Server. Now Client A can login to the Login Server sending its username and password. When these information is correct, the Login Server acknowledges it. Now Client A can ask for the online buddies to different Nodes. The Nodes respond with the online buddies. Also the Super Node will be asked to send information about the online buddies.

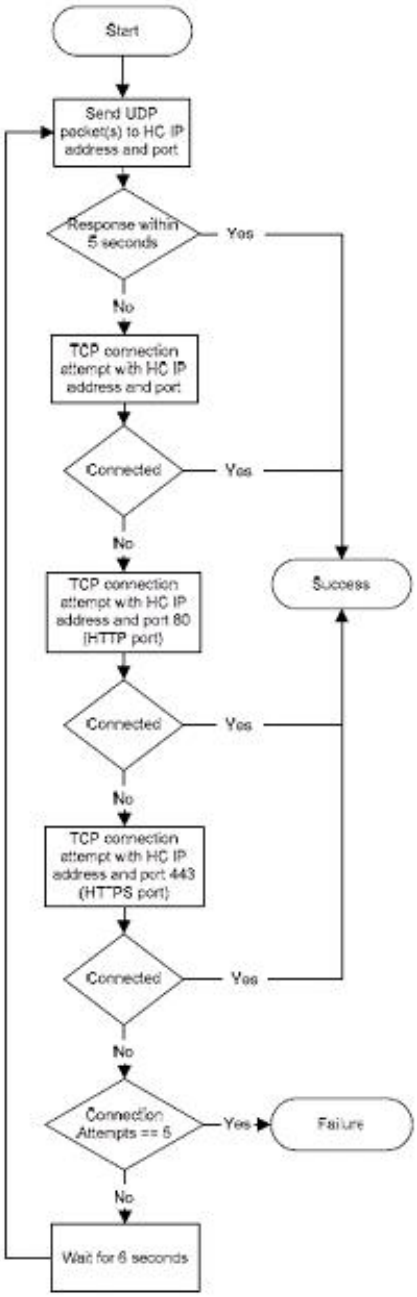


Figure B.21: Skype login algorithm

The Skype Client must authenticate the username and password with the Skype Login Server. This server ensures that Skype names are unique across the Skype name space. Baset and Schulzinne [64] found out during their experiments that the Skype Client always exchanged data over TCP with a node whose IP address is 80.160.91.11. This node is probably the Login Server and this server is hosted by an ISP based in Denmark. However, in my experiments

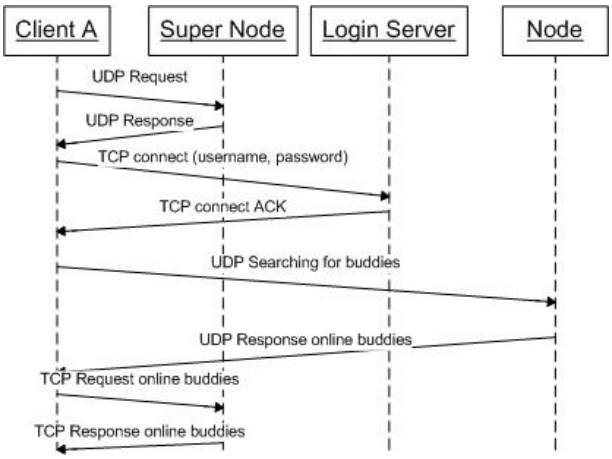


Figure B.22: Login sequence diagram

this IP address never shows up. According to the whois database of RIPE [37], this IP address is indeed located in Denmark, but there is no evidence that this IP address belongs to the Skype company. The IP ranges that belongs to the Skype network according to Ripe can be found in Appendix B.

A newer research, presented by Mirtic and Radusinovic [68] shows the Skype network has been grown, which means more Login Servers are used now, and thus more IP addresses can be used for the login process.

Buddy search
Skype has developed the Global Index technology [44]. This technology allows super nodes

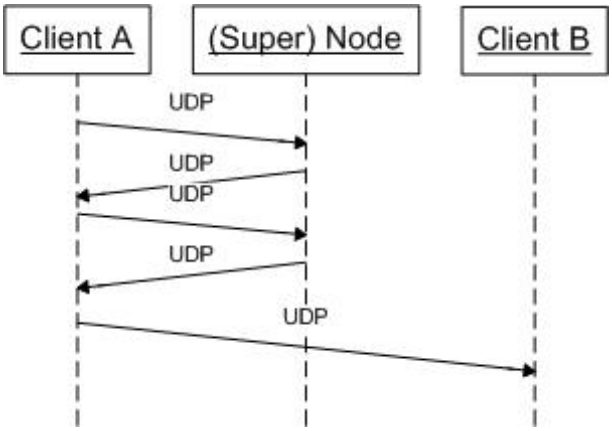


Figure B.23: Buddy Search sequence diagram

to communicate in such way that all other nodes in the network are aware of the available users. The Global Index technology is a multi-tiered network where super nodes communicate in such a way that every node in the network has full knowledge of all available users and

resources with minimal latency.

According to [66] to search for a user, a Skype client sends a request to its supernode, which apparently responds with four IP addresses with respective port numbers. Skype clients exchange TCP packets with its supernode and UDP packets with other nodes. It is unclear why Skype uses different types of packets with different nodes. As long as the Skype client does not find the user, it continues to request for new nodes to contact from its super node.

Figure B.23 shows the sequence diagram for the Buddy Search. Requests for a new buddy are send to several (super) nodes. If the buddy is found, Client A receives the IP address, and is able to contact Client B.

Messaging
Figure B.24 shows the sequence diagram for the Messaging. Client A first contact the Super

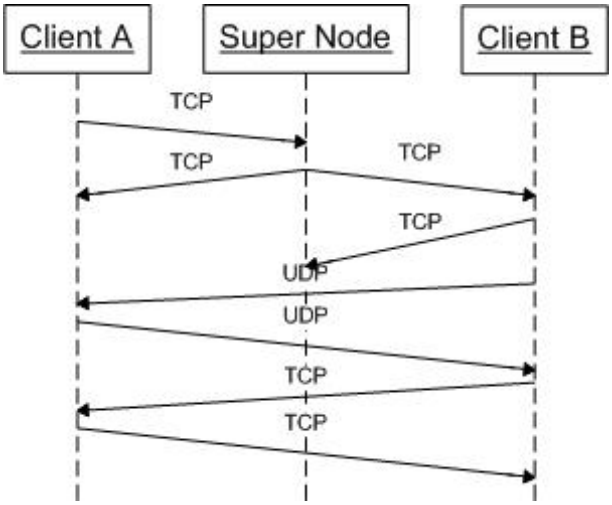


Figure B.24: Messaging sequence diagram

Node to find Client B. Client B will get the IP address of Client A to setup a peer-to-peer connection between the two clients, so Client A can send a message to Client B and opposite.

Filetransfer
Figure B.25 shows the sequence diagram for the Filetransfer. Again, Client A first contacts the Super Node, to find Client B. Client A will receive the IP address of Client B to setup a peer-to-peer connection to send the file to Client B. Client B will acknowledge the reception of the file.

Call
Skype always uses TCP to signaling, because the signaling is the most critical part of call establishment. In case the caller is behind a NAT, the caller first has to contact another online Skype node, which will forwards the packets to the receiver. This means, if a peer is behind a NAT, it will need assistance from other nodes in the network, which have either public IP or are not firewall restricted, to establish phone calls.

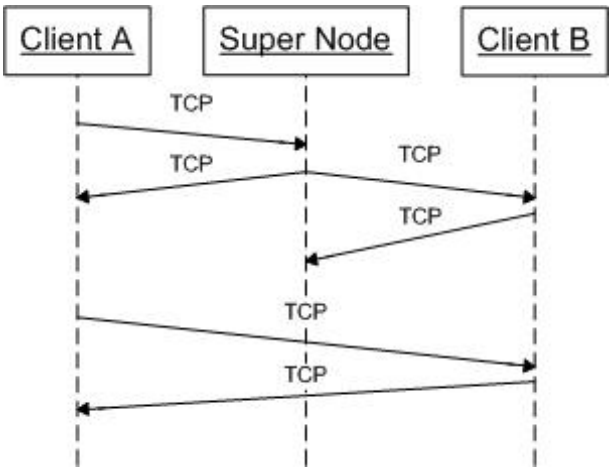


Figure B.25: Filetransfer sequence diagram

For the voice transmission, mostly UDP packets are used. TCP is used in case the caller or receiver is behind a NAT and UDP-restricted firewall. The voice messages in Skype have to be encrypted end-to-end, since the messages travels through other, possible unreliable nodes in the P2P network

[66] concludes that Skype works seamlessly behind most of the NAT and firewalls, calls are encrypted end-to-end since all the calls are routed through public Internet and Skype is not pure P2P system since login server is required.

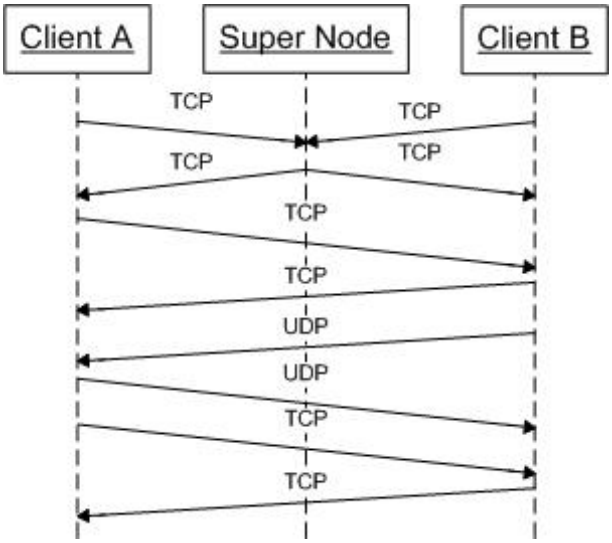


Figure B.26: Call sequence diagram

Figure B.26 shows the sequence diagram for the call. Again, Client A first connects to the Super Node to setup a peer-to-peer connection with Client B. Now it is possible to have a

conversation.

Appendix C

Additional information: Validation

This appendix validates the design. It gives a validation with sequence diagrams, where the VoIP systems are connected using several scenarios. Which events happen and what the responses are can be illustrated in such kind of sequence diagrams.

Because the login is done without the use of the Gateway, login does not have to be validated.

BIT message	MSN, ICQ, Yahoo
No BIT message	GTalk, Skype
Potential buddies	Skype, ICQ (partly)
No potential buddies	MSN, GTalk, Yahoo, ICQ (partly)
Buddy ack	MSN, Yahoo, ICQ
No buddy ack	GTalk, Skype

Table C.1: BIT messages and potential buddies

Table C.1 shows which VoIP system supports Buddy Is Typing (BIT) messages, which VoIP system provides a list of potential buddies after a buddy search and which VoIP system needs the acceptance of the buddy to add the buddy to the contact list.

C.1 Buddy search

Figures C.1 till C.13 show the sequence diagrams for the Buddy search. A buddy search is done as explained in Chapter 8, first a message with the buddy search request in it is sent to the Gateway, then a buddy search on the other side, afterwards a message back to the sender with the buddy name, and last a buddy search for the Proxy.

The sequence diagrams shows three differences. First of all, some VoIP systems send Buddy Is Typing (BIT) messages, before sending the actual message, others do not. Second, some VoIP systems provide a list of potential buddies, others only provide the matching entered buddy name. At the left side of the diagram, both the BIT message and the potential buddies can happen (or not). At the right side, only the potential buddies can happen. This means $2^3 = 8$ possible scenarios exists for these two differences. Third some VoIP systems ask for acceptance of the buddy before adding the buddy to the contact list, others do not.

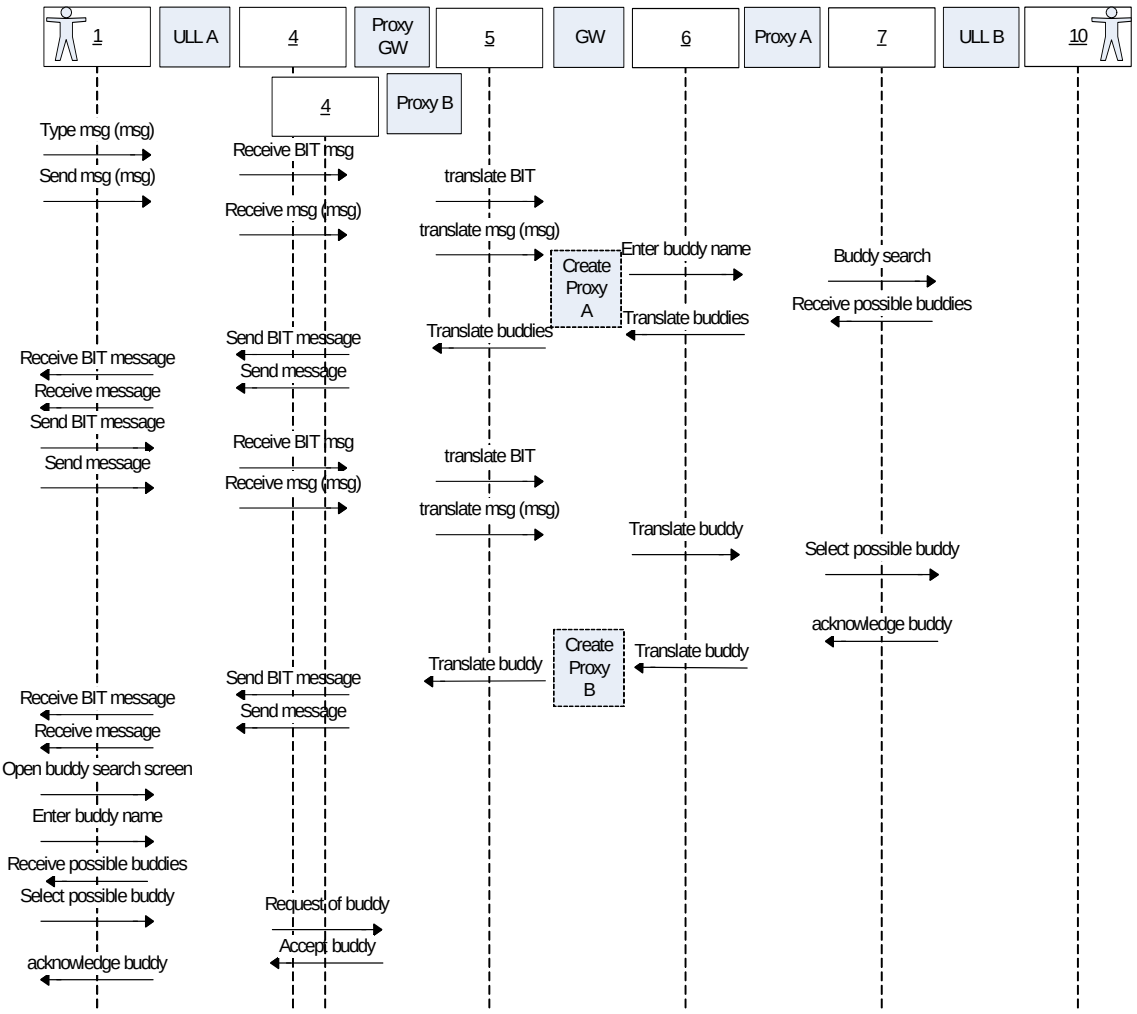


Figure C.1: ICQ → Skype

This makes five more scenarios exist. Summed up with the 8 possible scenario's, 13 possible scenario's are presented in this section.

Figure C.1 shows the sequence diagram for an ICQ application at the left side (VoIP system A) and a Skype application at the right side (VoIP system B). ICQ first sends a BIT message, before sending the actual message. At the Gateway the command "Search buddy" is activated and thus a buddy search happens at the network of VoIP system B. Skype first provides some potential buddies, so a message with the list of potential buddies is sent to the ICQ client. The user sends a message to the Gateway with the buddy it wants to select. Now the Gateway selects the buddy in the Skype network. The found buddy is added to the contact list, a Proxy B is set up (as a copy of this new found buddy) and the name of this Proxy is sent to the ICQ client. Because it first sends a BIT message, the user first sees (shortly) the BIT message. After receiving the buddy name, the user searches for the Proxy in its own network. As response, a contact list with the added buddy is sent.

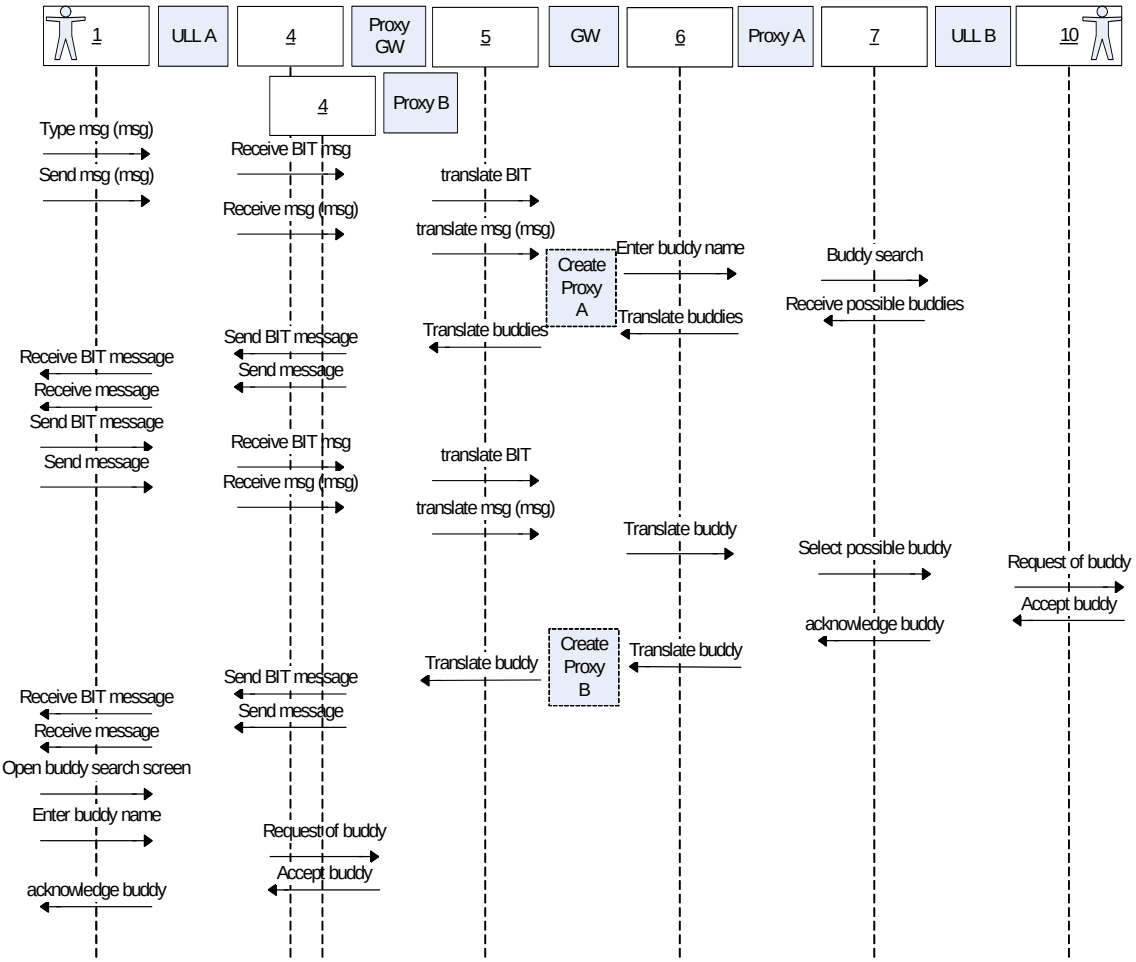


Figure C.2: MSN, ICQ, Yahoo → ICQ

The same idea happens for Figure C.2. Here the left side (VoIP system A) can be MSN, ICQ or Yahoo and the right side (VoIP system B) can be ICQ. This means MSN to ICQ and Yahoo to ICQ can be realized.

The same idea happens for Figure C.3, although in this figure Skype does not accept the buddy search request (and the buddy search request is successfully completed). Here the left side (VoIP system A) can be MSN, ICQ or Yahoo and the right side (VoIP system B) can be Skype. This means MSN to Skype, ICQ to Skype and Yahoo to Skype can be realized.

Figure C.4 shows the diagram with on the left side ICQ and on the right side MSN or Yahoo. Figure C.5 shows on the left side ICQ and on the right side GTalk. Figure C.6 shows on the left side MSN, ICQ or Yahoo and on the right side MSN, Yahoo or ICQ. figure C.7 shows on the left side MSN, ICQ or Yahoo and on the right side GTalk. Figure C.8 shows on the left side Skype and on the right side ICQ. Figure C.9 shows on the left side GTalk and on the right side ICQ. Figure C.10 shows on the left side GTalk and on the right side Skype. Figure C.11 shows on the left side Skype and on the right side MSN, Yahoo or ICQ. Figure C.12

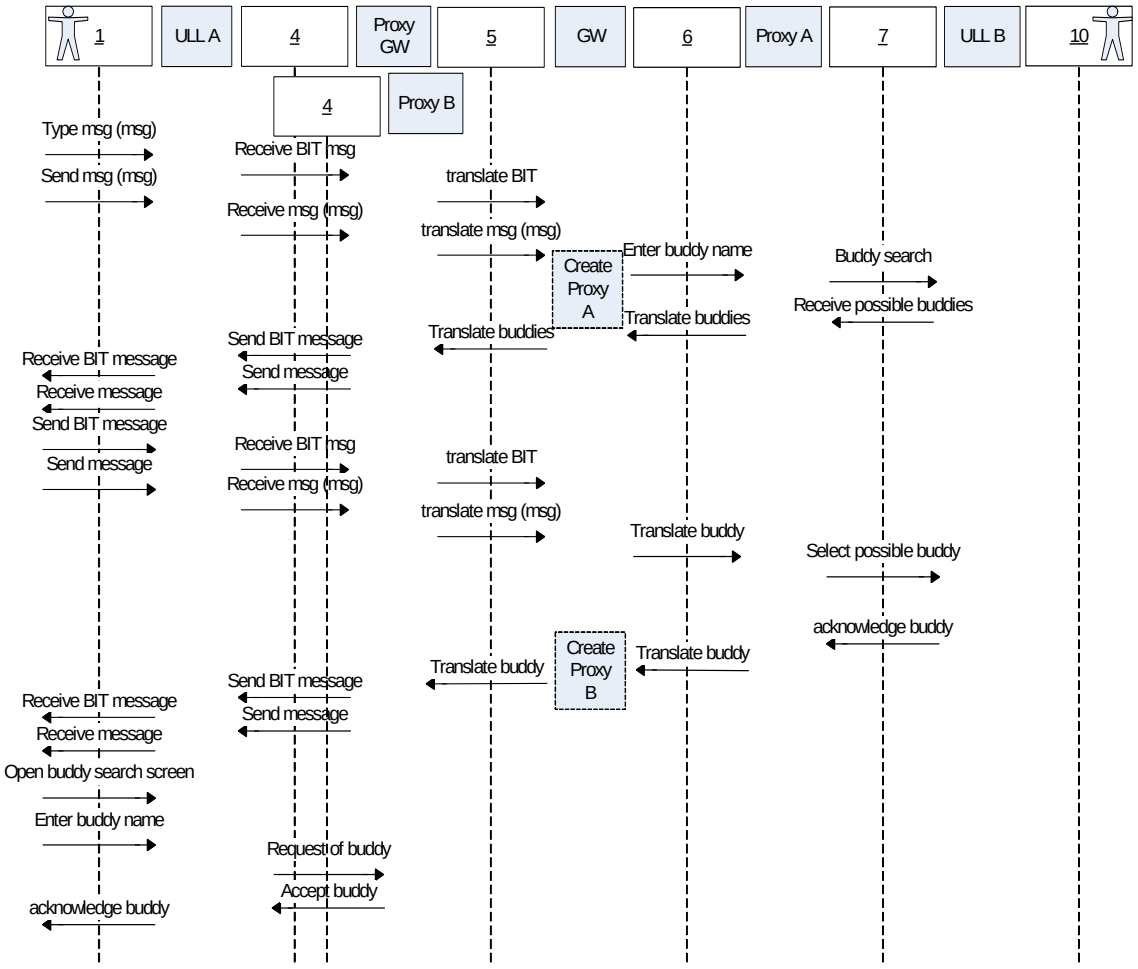


Figure C.3: MSN, ICQ, Yahoo → Skype

shows on the left side Skype and on the right side GTalk. Figure C.13 shows on the left side GTalk and on the right side MSN, Yahoo or ICQ.

All these figures combined show the connections between all five VoIP systems for the buddy search.

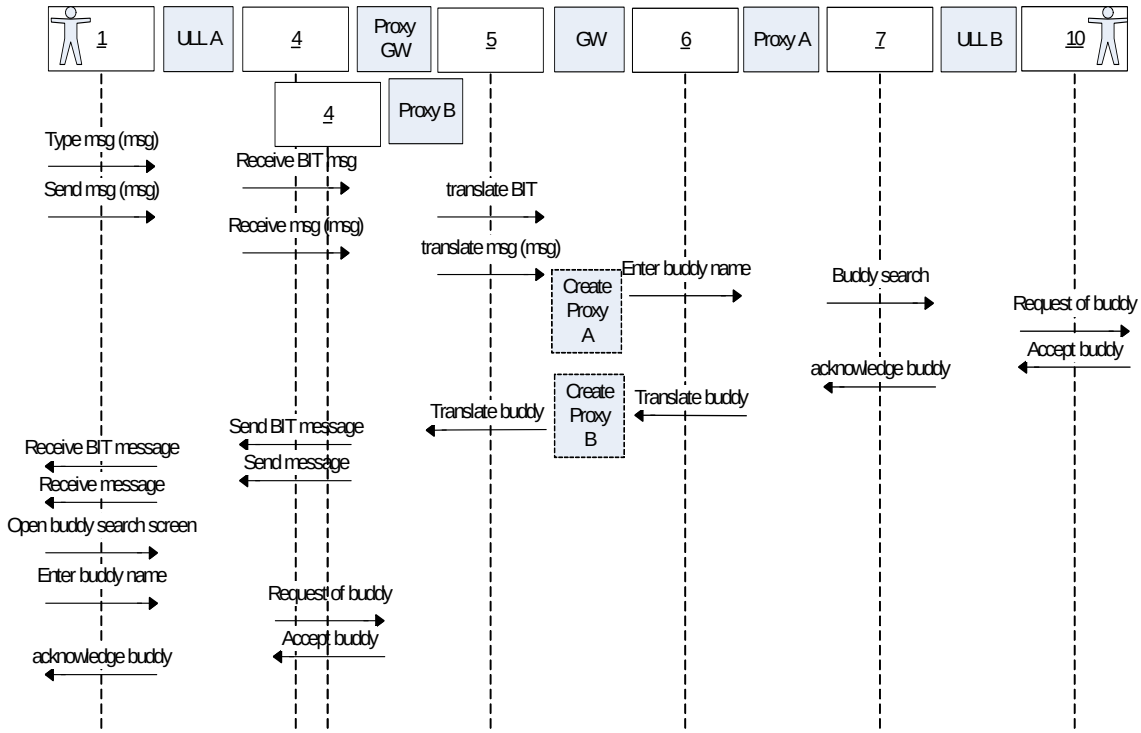


Figure C.4: ICQ → MSN, Yahoo

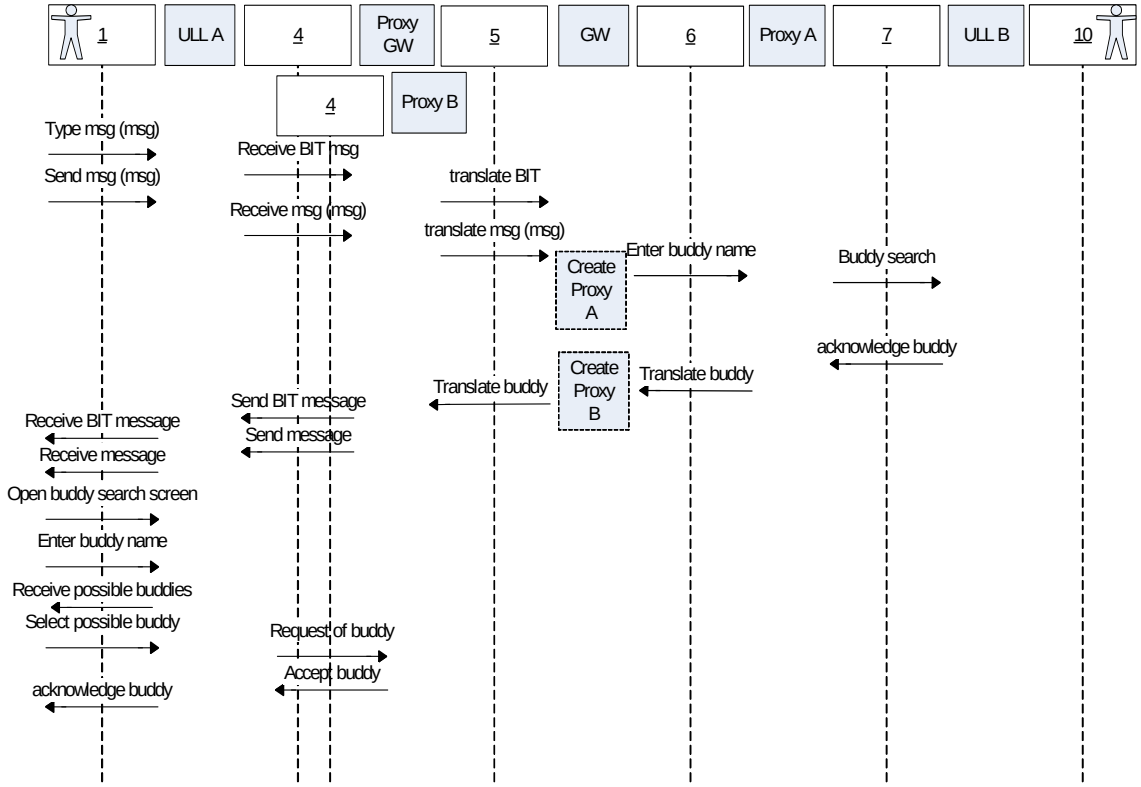


Figure C.5: ICQ → GTalk

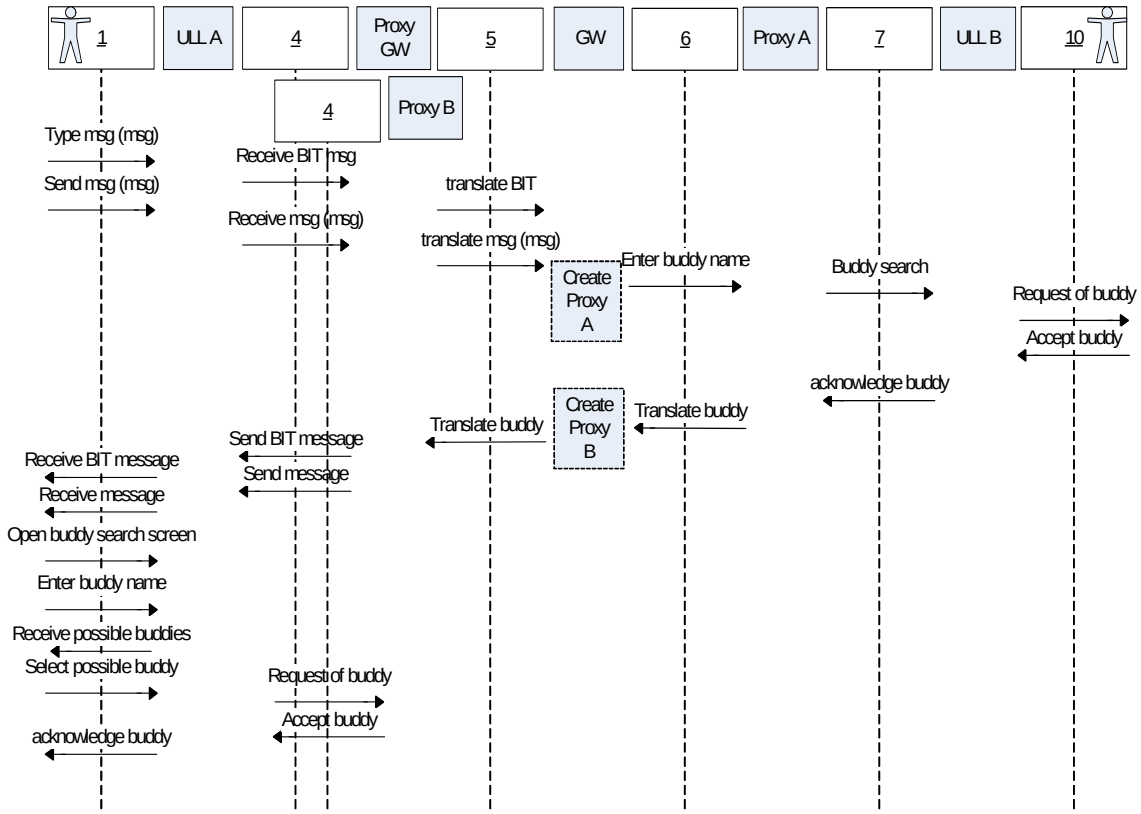


Figure C.6: MSN, ICQ, Yahoo → MSN, Yahoo, ICQ

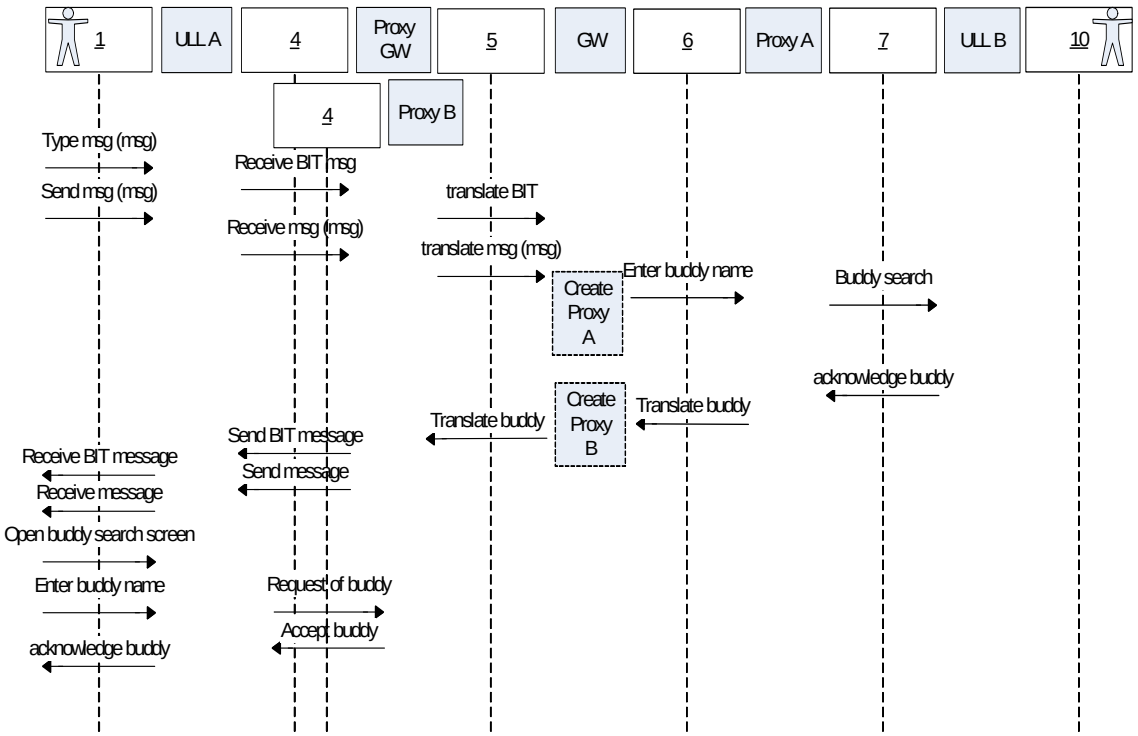


Figure C.7: MSN, ICQ, Yahoo → GTalk

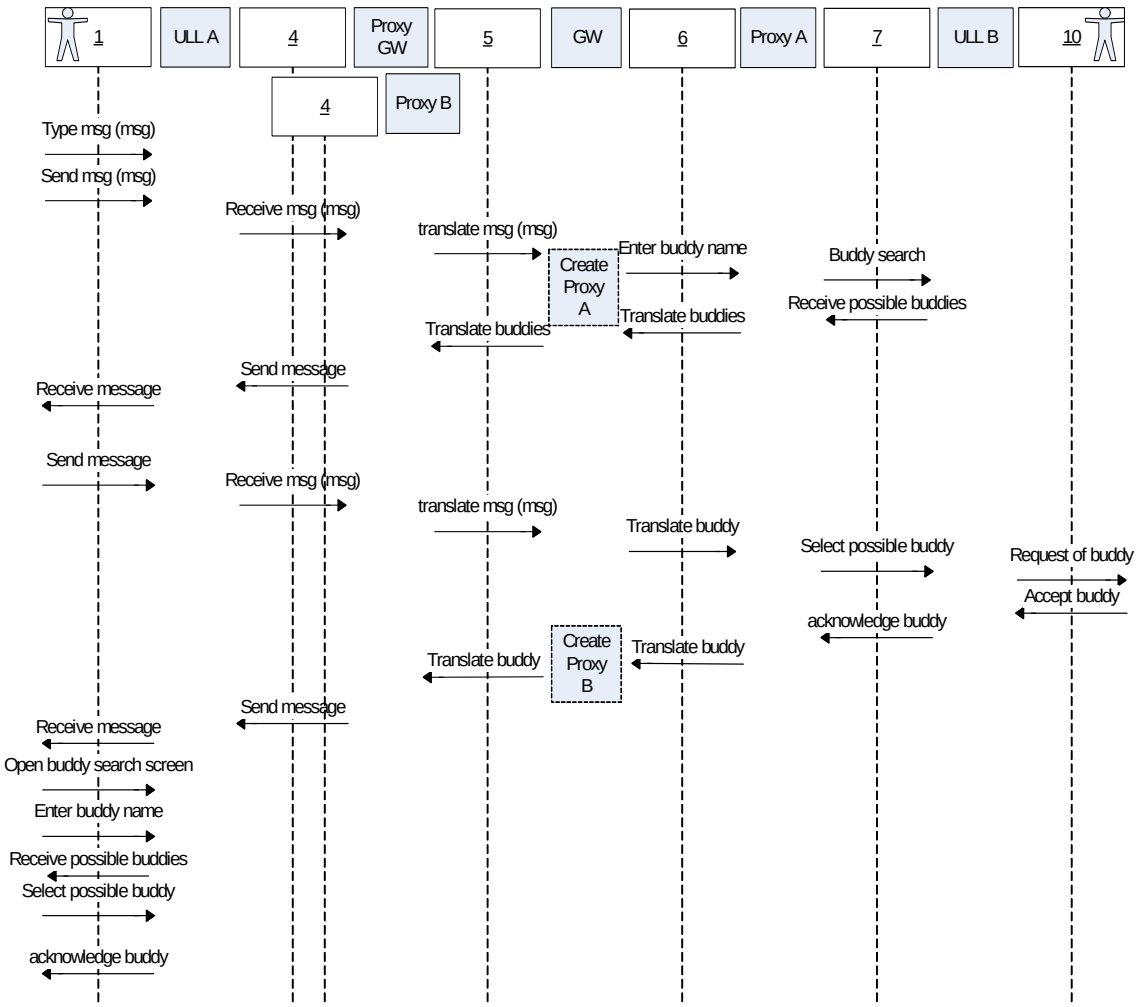


Figure C.8: Skype → ICQ

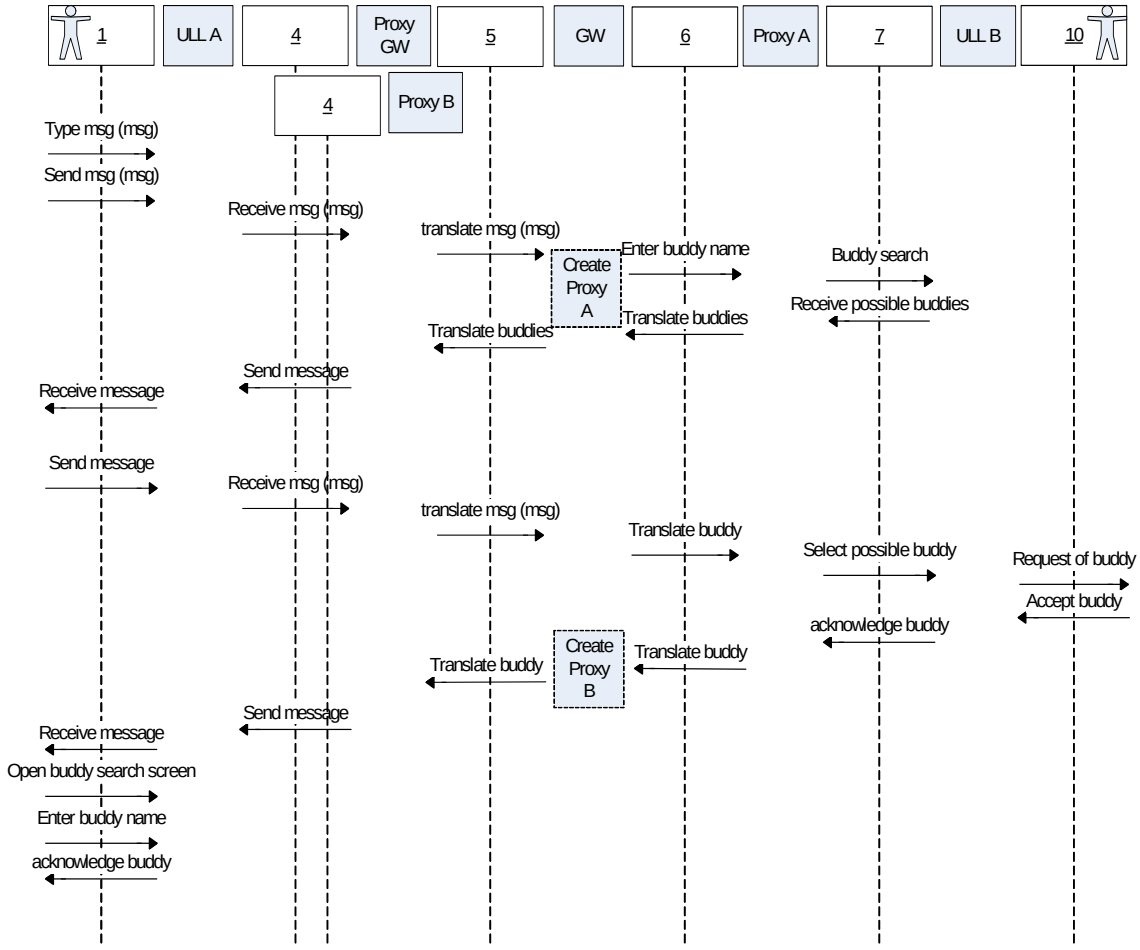


Figure C.9: GTalk → ICQ

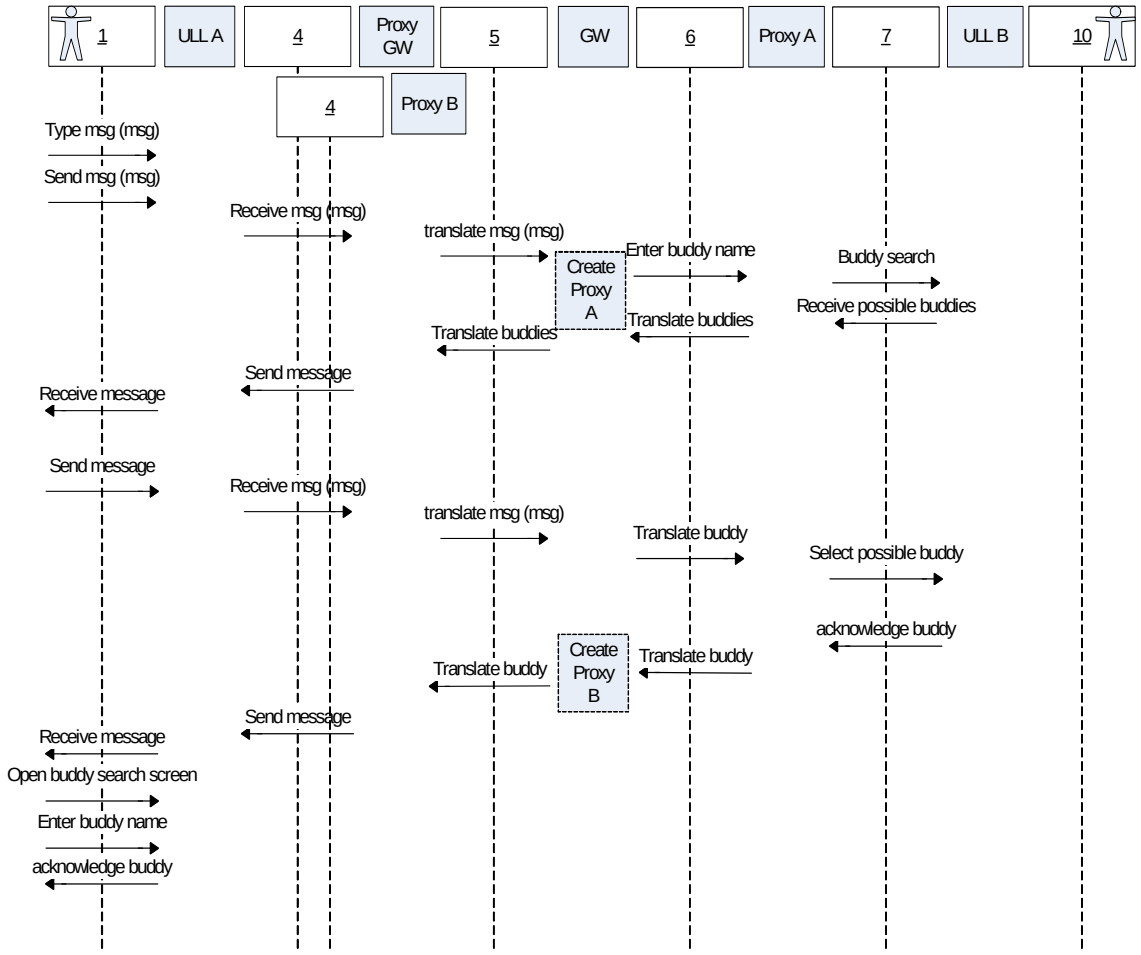


Figure C.10: GTalk → Skype

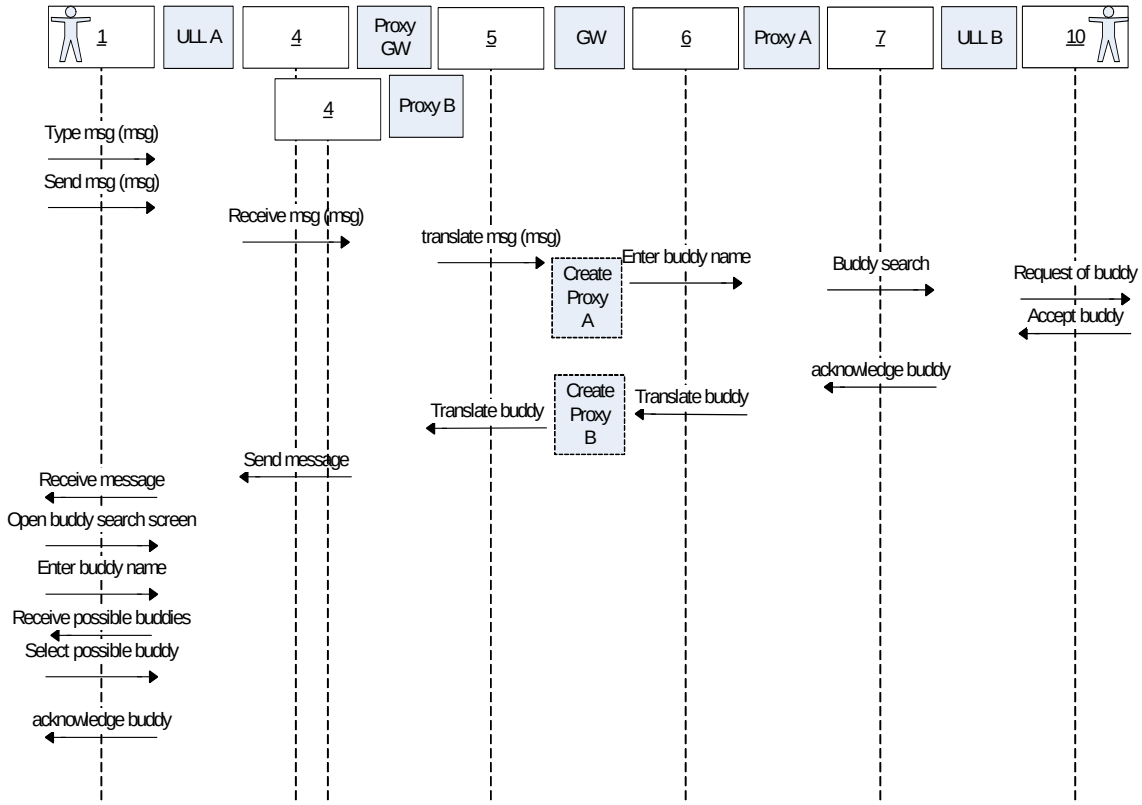


Figure C.11: Skype → MSN, Yahoo, ICQ

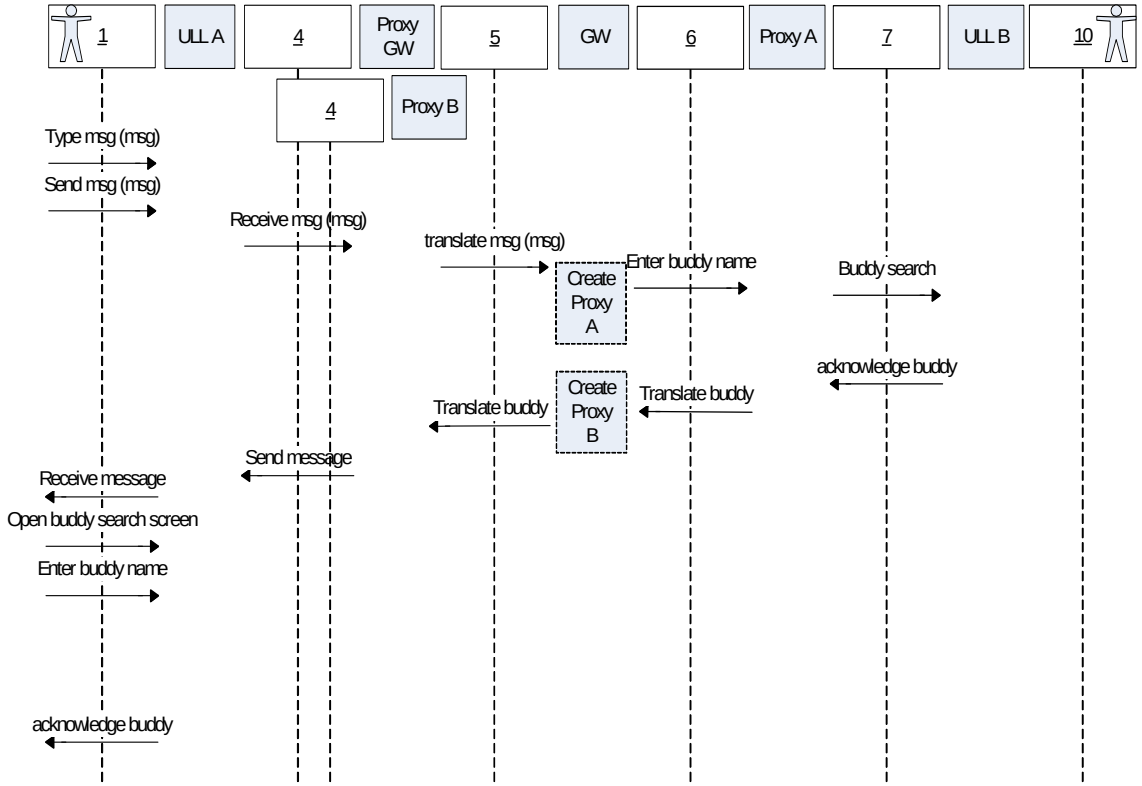


Figure C.12: Skype → GTalk

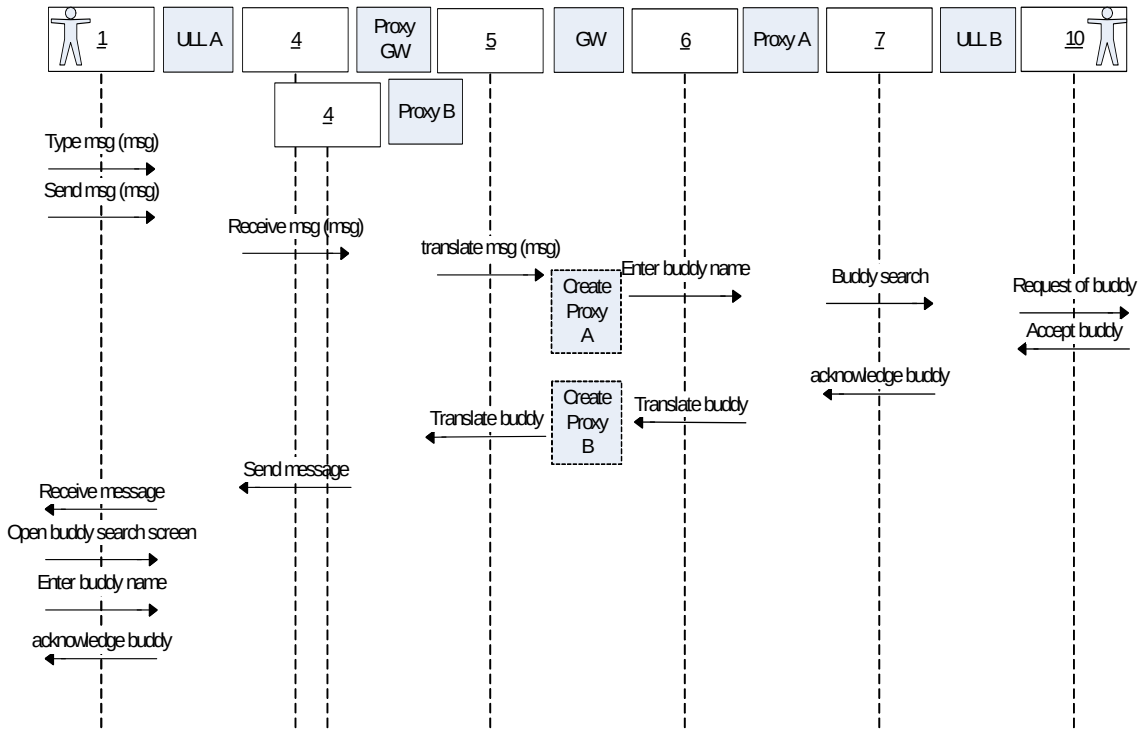


Figure C.13: GTalk → MSN, Yahoo, ICQ

C.2 Messaging

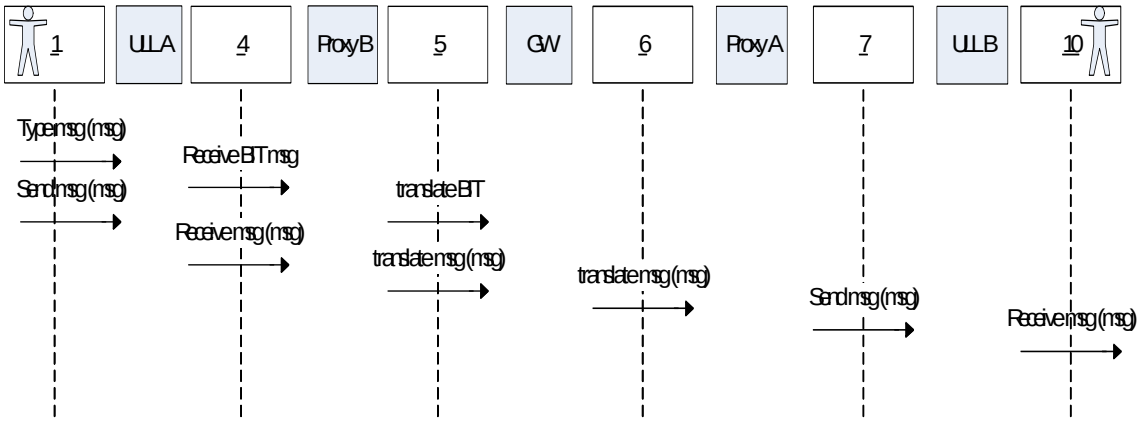


Figure C.14: MSN, Yahoo, ICQ → GTalk, Skype

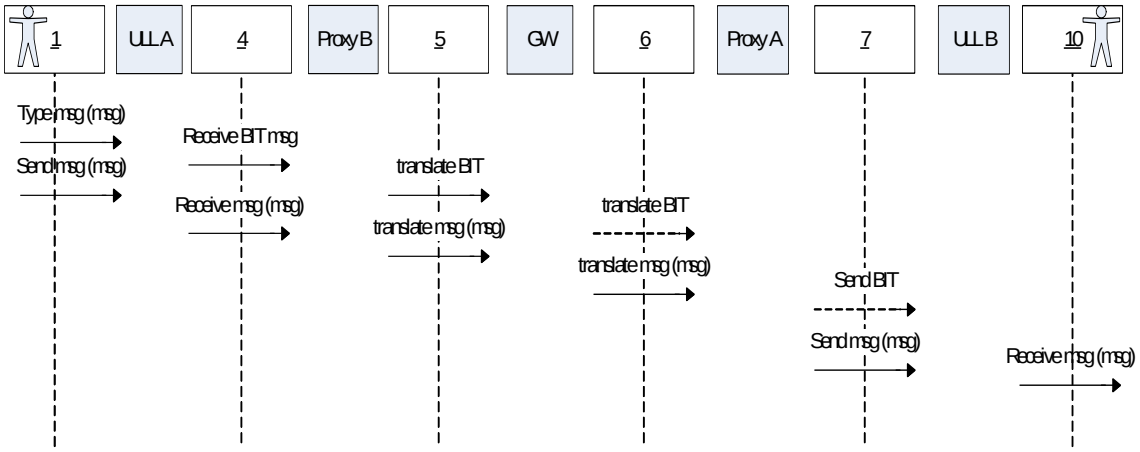


Figure C.15: MSN, Yahoo, ICQ → MSN, Yahoo, ICQ

Figure C.14 till C.17 show the sequence diagrams for the Messaging. The differences are the BIT message, a VoIP system offers it or not. This provides four possible diagrams. Figure C.14 shows on the left side a BIT message, but not on the right side. Figure C.15 shows on both sides the BIT messages. Figure C.16 shows on both side no BIT messages and Figure C.17 shows only on the right side BIT messages.

Again, all these figures combined show the connections between all five VoIP systems.

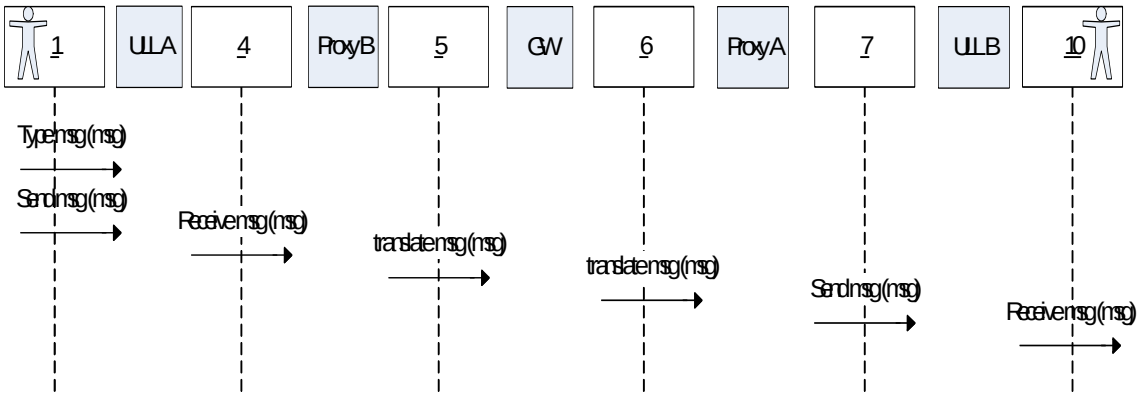


Figure C.16: GTalk, Skype → GTalk, Skype

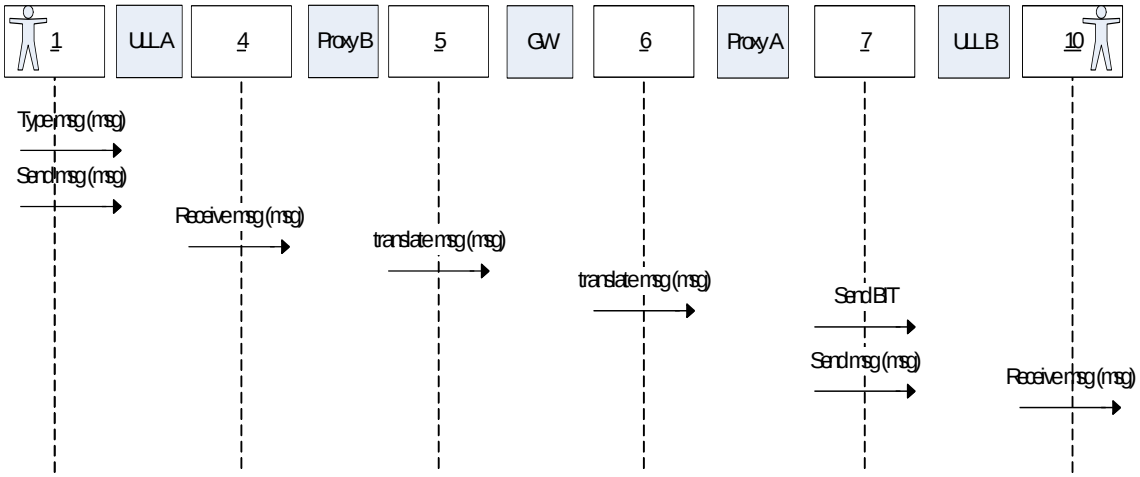


Figure C.17: GTalk, Skype → MSN, Yahoo, ICQ

C.3 Call

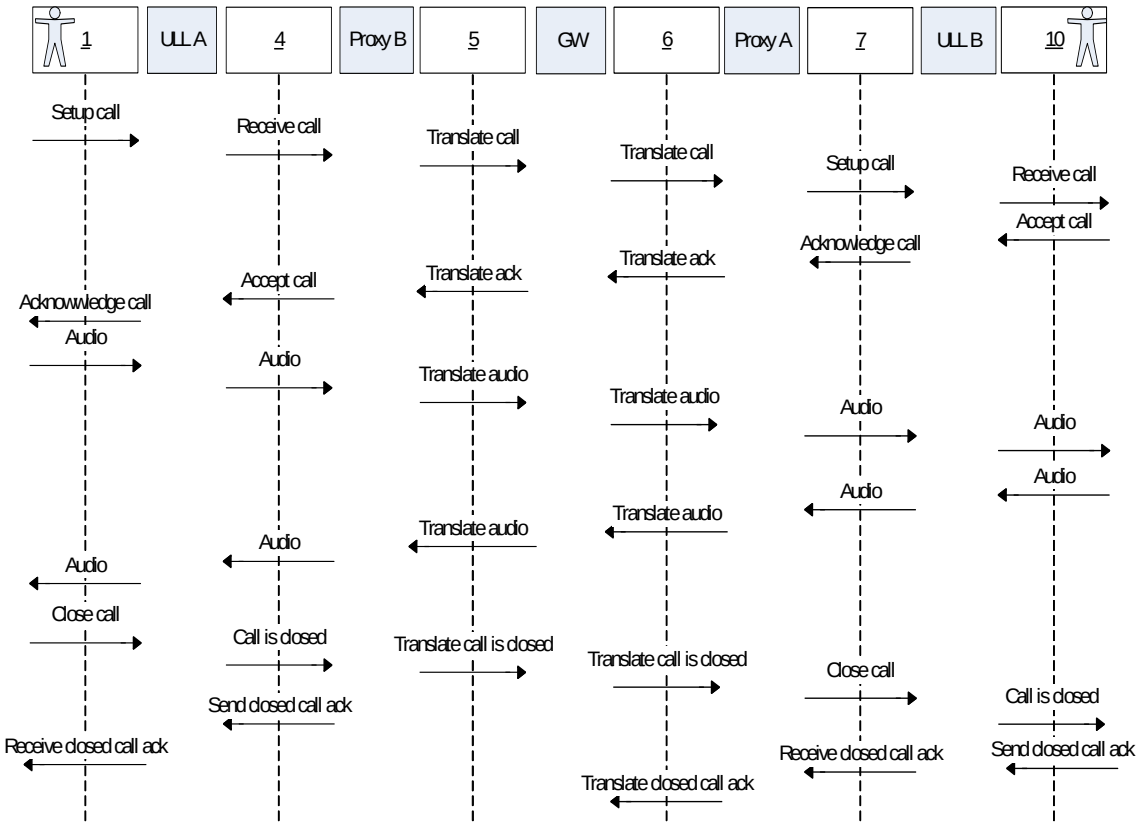


Figure C.18: MSN, Yahoo, GTalk, ICQ, Skype → MSN, Yahoo, GTalk, ICQ, Skype

For the call, only one sequence diagram show the connections between all five VoIP systems, because all VoIP systems handle the call (setup, tear down and the audio transfer) as the same way at this abstraction level. Figure C.18 shows this.

Bibliography

[1] Bbc - history, alexander graham bell (1847-1922), 12-09-2006. Available from: http://www.bbc.co.uk/history/historic_figures/bell_alexander.shtml.

[2] Cerulean studios: Creators of trillian and trillian pro instant messengers, 13-03-2007. Available from: http://www.ceruleanstudios.com/.

[3] Bolt, beranek and newman, 13-10-2006. Available from: http://nl.wikipedia.org/wiki/Bolt,_Beranek_and_Newman.

[4] History of voip - intertangent technology directory, 13-10-2006. Available from: http://www.intertangent.com/023346/Articles_and_News/1413.html.

[5] Osi model, 14-03-2007. Available from: http://en.wikipedia.org/wiki/OSI_model.

[6] Circuit switching, 14-11-2006. Available from: http://en.wikipedia.org/wiki/Circuit_switching.

[7] Comparison of instant messaging clients, 14-11-2006. Available from: http://en.wikipedia.org/wiki/Comparison_of_instant_messaging_clients.

[8] Comparison of instant messaging clients, 14-11-2006. Available from: http://www.answers.com/topic/comparison-of-instant-messaging-clients.

[9] Faq google talk, 14-11-2006. Available from: http://www.google.com/talk/about.html#privacy.

[10] The google blog, 14-11-2006. Available from: http://www.thegoogleblog.com/gtalk-tweaks/.

[11] Google talk, 14-11-2006. Available from: http://en.wikipedia.org/wiki/Google_Talk.

[12] Google talk and open communications, 14-11-2006. Available from: http://code.google.com/apis/talk/open_communications.html.

[13] Google talk poligamy patch, 14-11-2006. Available from: http://www.softpedia.com/get/Internet/Chat/Instant-Messaging/Google-Talk-Poligamy-patch.shtml.

[14] Google talk: Run multiple instances or login as multiple users, 14-11-2006. Available from: http://www.tech-recipes.com/google_tips975.html.

[15] Google talk to voip, 14-11-2006. Available from: `http://www.gtalk2voip.com/`.

[16] Gtalk website, 14-11-2006. Available from: `www.google.com/talk`.

[17] How skype & co. get round firewalls, 14-11-2006. Available from: `http://www.heise-security.co.uk/articles/82481`.

[18] Icq, 14-11-2006. Available from: `http://en.wikipedia.org/wiki/Icq`.

[19] Icq lite starter, 14-11-2006. Available from: `http://www.dirfile.com/icq_lite_starter.htm`.

[20] The icq story, 14-11-2006. Available from: `http://www.icq.com/info/icqstory.html`.

[21] Icq website, 14-11-2006. Available from: `www.icq.com`.

[22] Instant messaging, 14-11-2006. Available from: `http://nl.wikipedia.org/wiki/Instant_messaging`.

[23] Jabber, 14-11-2006. Available from: `http://xmpp.packetlabs.org/jabber`.

[24] Jabber, 14-11-2006. Available from: `http://de.wikipedia.org/wiki/Jabber`.

[25] Jabber entities, 14-11-2006. Available from: `http://micro-jabber.sourceforge.net/?What_is_Jabber%3F:Jabber_entities`.

[26] Jabber: het wat, het waarom, maar vooral het hoe..., 14-11-2006. Available from: `http://users.pandora.be/cobnet/ict/Jabber/jabber_vvv.html`.

[27] List of sip software, 14-11-2006. Available from: `http://en.wikipedia.org/wiki/List_of_SIP_software`.

[28] Media gateway control protocol, 14-11-2006. Available from: `http://en.wikipedia.org/wiki/Media_Gateway_Control_Protocol`.

[29] Megaco/h.248, 14-11-2006. Available from: `http://www.javvin.com/protocolMegaco.html`.

[30] Msn, 14-11-2006. Available from: `http://en.wikipedia.org/wiki/Msn`.

[31] Msn messenger 4&5 polygamy, 14-11-2006. Available from: `http://www.softpedia.com/get/Internet/Chat/Instant-Messaging/MSN-Messenger-Polygamy.shtml`.

[32] Msn messenger website, 14-11-2006. Available from: `get.live.com`.

[33] New google talk offers instant messaging & voice chat, 14-11-2006. Available from: `http://searchenginewatch.com/showPage.html?page=3529566`.

[34] Open mulitple instances of icq, 14-11-2006. Available from: `http://www.freevbcode.com/ShowCode.asp?ID=3253`.

[35] Packet switching, 14-11-2006. Available from: `http://en.wikipedia.org/wiki/Packet_switching`.

[36] Presence and awareness services, 14-11-2006. Available from: http://phoenix.labri.fr/documentation/sip/Documentation/Papers/Programming_SIP/Presentation/Liscano.pdf.

[37] Ripe website, 14-11-2006. Available from: http://www.ripe.com.

[38] Rsdevs psgw, 14-11-2006. Available from: http://www.rsdevs.com/psgw.shtml.

[39] Running multiple instances of skype, 14-11-2006. Available from: http://www.tipmonkies.com/2005/08/18/running-multiple-instances-of-skype/.

[40] Securing public instant messaging (im) at work, 14-11-2006. Available from: http://caia.swin.edu.au/reports/040726A/CAIA-TR-040726A.pdf.

[41] The session initiation protocol (sip): A key component for internet telephony, 14-11-2006. Available from: http://www.callcentermagazine.com/article/CTM20000608S0019.

[42] Signaling system 7, 14-11-2006. Available from: http://en.wikipedia.org/wiki/SS7.

[43] Sip phone extension for mozilla thunderbird, 14-11-2006. Available from: http://cockatoo.mozdev.org/.

[44] Skype explained, 14-11-2006. Available from: http://skype.com/products/explained.html.

[45] Skype website, 14-11-2006. Available from: www.skype.com.

[46] Skype website, 14-11-2006. Available from: http://www.skype.com.

[47] Skype wikipedia, 14-11-2006. Available from: http://en.wikipedia.org/wiki/Skype.

[48] Stun - wikipedia, 14-11-2006. Available from: http://en.wikipedia.org/wiki/STUN.

[49] Trillian (instant messaging client), 14-11-2006. Available from: http://en.wikipedia.org/wiki/Trillian_(instant_messenger).

[50] Uplink, 14-11-2006. Available from: http://www.nch.com.au/skypetosip/index.html.

[51] Virtual audio cable, 14-11-2006. Available from: http://spider.nrcde.ru/music/software/eng/vac.html.

[52] Voip application development: view from inside, 14-11-2006. Available from: http://www.gtalk2voip.com/article1_en.html.

[53] Website asterisk, 14-11-2006. Available from: http://www.asterisk.org/.

[54] Website ekiga, 14-11-2006. Available from: http://www.ekiga.net.

[55] Website gizmo project, 14-11-2006. Available from: http://www.gizmoproject.com/.

[56] What is mgcp?, 14-11-2006. Available from: http://www.tech-faq.com/mgcp.shtml.

[57] What is ss7 - isup?, 14-11-2006. Available from: http://www.asknumbers.com/Whatisss7isup.aspx.

[58] Y! multi messenger 8.0.0.508, 14-11-2006. Available from: `http://www.softpedia.com/get/Internet/Chat/Instant-Messaging/Yahoo-Multi-Messenger.shtml`.

[59] Yahoo, 14-11-2006. Available from: `http://en.wikipedia.org/wiki/Yahoo`.

[60] Yahoo! messenger, 14-11-2006. Available from: `http://en.wikipedia.org/wiki/Yahoo%21_Messenger`.

[61] Yahoo website, 14-11-2006. Available from: `messenger.yahoo.com`.

[62] Ymsg, 14-11-2006. Available from: `http://www.answers.com/topic/ymsg`.

[63] Ymsg, 14-11-2006. Available from: `http://en.wikipedia.org/wiki/YMSG`.

[64] *An analysis of the Skype Peer-to-peer Internet Telephony Protocol*, 2004.

[65] *Securing Public Instant Messaging (IM) At Work*, July 2004. Available from: `http://caia.swin.edu.au/reports/040726A/CAIA-TR-040726A.pdf`.

[66] *An analysis of Hybrid and Pure Peer-to-Peer Technologies for IP Telephony*, April 2005. Available from: `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1196353`.

[67] *Blocking MSN: A Case Study of Preventing the Abuse of IM*, October 2005. Available from: `http://ieeexplore.ieee.org/iel5/10412/33074/01554237.pdf?isnumber=&arnumber=1554237`.

[68] *The Principles of Speech Transmission Realization in Skype*, September 2006.

[69] Eventhelix.com - sequence diagram based system design tool, 22-10-2006. Available from: `http://www.eventhelix.com/`.

[70] Wireshark: The world's most popular network protocol analyzer, 22-10-2006. Available from: `http://www.wireshark.org/`.

[71] Gtalk plug-in, 24-03-2007. Available from: `http://code.google.com/apis/talk/open_communications.html`.

[72] Icq plug-in, 24-03-2007. Available from: `http://www.icq.com/webtools/app-develop.html`.

[73] Msn plug-in, 24-03-2007. Available from: `http://dev.live.com`.

[74] Skype-plugin, 24-03-2007. Available from: `https://developer.skype.com/`.

[75] Yahoo plug-in, 24-03-2007. Available from: `http://developer.yahoo.com/messenger`.

[76] Icq api, 29-12-2007. Available from: `http://joust.kano.net/docs/api`.

[77] Icq api sessions, 29-12-2007. Available from: `http://joust.kano.net/docs/api/net/kano/joscar/rv/RvSession.html`.

[78] Skype api, 29-12-2007. Available from: `http://share.skype.com/media/1.2_api_doc_en.pdf`.

[79] Skype api, 29-12-2007. Available from: `https://developer.skype.com/GettingStarted/KickStartGuide`.

[80] Voice chat applications, 29-12-2007. Available from: `http://code.google.com/apis/talk/libjingle/voice_chat.html`.

[81] M. Arango, A. Dugan, C. Huitema, and S. Pickett. *Media Gateway Control Protocol(MGCP)*. The Internet Society, 1.0 edition, 1999.

[82] U. Black. *Internet Telephony: Call Processing Protocols*. Prentice Hall PTR, 2001.

[83] U. Black. *Voice over IP*. Prentice Hall PTR, 2nd edition, 2002.

[84] I. M. A. Caballero. Secure mobile voice over ip. Master's thesis, June 2003.

[85] E. B. Fjellskl and S. Solberg. Evaluation of voice over mpls (vompls) compared to voice over ip (voip). Master's thesis, May 2002. Available from: `http://student.grm.hia.no/master/ikt02/ikt6400/g12/Evaluation-of-VoMPLS-compared-to-VoIP.pdf`.

[86] M. Handley and V. Jacobson. Sdp: Session description protocol. Rfc, The Internet Society, April 1998. Available from: `ftp://ftp.rfc-editor.org/in-notes/rfc2327.txt`.

[87] N. Hindocha. Threats to instant messaging. Master's thesis. Available from: `http://cnscenter.future.co.kr/resource/rsc-center/vendor-wp/symantec/ThreatsToIM.pdf`.

[88] S. Ludwig, J. Beda, P. Saint-Andre, R. McQueen, S. Egan, and J. Hildebrand. Xep-0166: Jingle. Xep, XMPP Standards Foundation, 1999-2007. Available from: `http://www.xmpp.org/extensions/xep-0166.html`.

[89] S. Ludwig, P. Saint-Andre, R. McQueen, and S. Egan. Xep-0167: Jingle audio via rtp. Xep, XMPP Standards Foundation, 1999-2007. Available from: `http://www.xmpp.org/extensions/xep-0167.html`.

[90] D. Minoli and E. Minoli. *Delivering voice over IP networks*. Indianapolis, IN, Wiley, 2nd edition, 2002.

[91] R. Movva and W. Lai. Msn messenger service 1.0 protocol. Rfc, Microsoft, August 1999. Available from: `http://www.hypothetic.org/docs/msn/ietf_draft.txt`.

[92] S. Norin and D. Stenman. Instant messaging skerhetshot och tgrder. Master's thesis. Available from: `http://dsv.su.se/en/seclab/pages/pdf-files/05-68.pdf`.

[93] R. Parhonyi. *MICRO PAYMENT GATEWAYS*. University of Twente, 2005.

[94] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. Sip: Session initiated protocol. Rfc, The Internet Society, June 2002. Available from: `http://www.apps.ietf.org/rfc/rfc3261.html`.

[95] H. Schildt. Sicherheitsaspekte von instant messaging. Master's thesis, July 2005. Available from: `http://archiv.tu-chemnitz.de/pub/2005/0091/data/IM-Aspekte.pdf`.

[96] C. Vissers, L. F. Pires, D. Quartel, and M. van Sinderen. *Design of Telematics Systems, Reader for the Design of Telematics Systems course.* University of Twente, 2003.