FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
ENSINANDO E APRENDENDO

JAVA APPLICATION DEVELOPMENT FOR SOFTPHONE MESSAGE SENDING

GABRIELA MARGARETO ROCA

GABRIELA MARGARETO ROCA

JAVA APPLICATION DEVELOPMENT FOR A SOFTPHONE MESSAGE SENDING

Trabalho de Conclusão de Curso – TCC a ser apresentado e submetido à avaliação para conclusão do Curso de Engenharia Telecomunicações do centro de Ciências Tecnológicas da Universidade de Fortaleza.

Tuthor: M. Sc. Geneflides Laureno da Silva

Fortaleza - Brasil
May 2014

JAVA APPLICATION DEVELOPMENT FOR SOFTPHONE MESSAGE SENDING

Gabriela Margareto Roca

PARECER: ______________________________

Data: 28/05/2014

BANCA EXAMINADORA:

____

M. Sc. Geneflides L da Silva

Fortaleza - Brasil
May 2014

Fortaleza - Brasil
May 2014

ACKNOWLEDGMENT

I dedicate this project to all those people, family and friends, who have somehow helped me to get this far. All who shared with me this years of incredible experiences, which have enriched me so much from the educational side as well as the personal one.

In particular, I would like to thank some people because of their specific help and support in doing this thesis:
Barrulas, because he helped us from the very beginning and gave us a hand when we needed it most.
Prof. Geneflides, because he have always believed in us and have made the effort so we could finally get to Fortaleza to work in this project. He have been a great supervisor and I learnt so much from him.
Judah, Tarciso and Felipe, because they have walked with me during most of the time of this project and they help me out always I needed.

I would also like to thank my parents for supporting me in all my endeavors and giving me the wonderful opportunity to reach this point of my life.

And Aleix, for being here all this time.

Finally, I dedicate the project to Natalia, the best travel companion I could never have had.

Fortaleza - Brasil
May 2014

ABSTRACT

This scientific work consists in developing and improving a multimedia application created in Java. A Softphone created using SIP protocol is improved adding a message function, making possible to send mesages using the UDP protocol.
This project presents the basics of Voice over IP communnications, transmission and communication protocols in general and the SIP protocol in detail. The aim of this paper is to show the implementation steps of the application in Java language, bringing up the contribution of this software for the business area, the academy and the software community.

Fortaleza - Brasil
May 2014

LIST OF ABBREVIATIONS

AGI	Asterisk Gateway Interface
ATA	Analog Telephone Adapter
GPL	General Public License
IAX	Inter-Asterisk eXchange
IETF	Internet Engineering Task Force
IN	Intelligent Network
IP	Internet Protocol
ISDN	Integrated Services for Digital Network
ITU	International Telecommunications Union
MCGP	Media Gateway Control Protocol
MEGACO	Media Gateway Control
NAT	Network Address Translation
PBX	Private Branch eXchange
PSTN	Public Switched Telephone Network
RAS	Registration, Admission and Status
RFC	Request For Comments
RSP	Real-time Stream Protocol
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
SCTP	Stream Control Transmission Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SS7	Signaling System nº7
TCP	Transmission Control Protocol
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol

URL	Uniform Resources Locator
VNET	Virtual NETwork
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
XMPP	eXtensible Messaging and Presence Protocol

LIST OF FIGURES

LIST OF TABLES

CONTENTS

Fortaleza - Brasil
May 2014

1. INTRODUCTION

Since the human had been human, the communication had existed. In fact, communication is a human need in order to interact between individuals. The essence of communication lies in being sociable, belonging to a community, and implies the interpersonal reciprocity of transmitting, receiving and exchanging ideas, feelings, thoughts, opinions and general information, i.e. meaningful messages.

The history of communication dates back to prehistory, when human used signals and minimum language to communicate. Since, significant changes occurred. Human communication was revolutionized with speech approximately 100,000 years ago, symbols were developed about 30,000 years ago, and writing in the past few centuries. Nowadays, technology also has revolutionized communication. Connect to anyone in any place in the world or make a videoconference from one continent to another, is possible.

Hence, this project only pretends to provide a part in the evolution of the communication, expanding somehow the opportunity of interaction between human beings.

This project is divided in four chapters. The next paragraph, number two, introduces the VoIP concepts and the main protocols that are involved in this technology. The third chapter contains more specific details about the SIP (Session Initiation Protocol): its function, its components and elements and its messages format. The chapter number four explains the JAVA application, its properties and its uses.

2. VOICE over IP

The telephone was the evolution and successive improvements of the electrical telegraph.
It was in 1804, when the catalan scientist Francisco Salva Campillo constructed an electrochemical telegraph and, in 1832, when Baron Schilling created an electromagnetic telegraph, starting the telephone history.
During the second half of the 19th century inventors such as Charles Bourseul, Thomas Edison, Elisha Gray and Alexander Graham Bell tried to find ways of sending multiple telegraph messages simultaneously over a single telegraph wire by using different modulated audio frequencies for each message. Their efforts to develop acoustic telegraphy to reduce the cost of telegraph messages led directly to the invention of the telephone: the speaking telegraph.

Was at this time when appeared the firsts telephones: they were wired together in pairs so users who wanted to talk to different people had as many telephones as necessary.

Trying to improve this rudimentary mechanism, a bell for signaling and a switch hook were added and, following the telegraph structure, each telephone was wired to a local telephone exchange, which were wired together with trunks. In this way, networks were connected in a hierarchically spanning cities, countries, continents and oceans. This was the beginning of the Public Switched Telephone Network (PSTN). It consists of telephone lines, fiber optic cables, microwave transmission links, cellular networks, communication satellites and undersea telephone cables, all interconnected by switching centers. Originally, it was an analogical network but nowadays is almost entirely digital in its core. The use of PSTN network introduced the Signaling System nº7 (SS7), a set of telephony signaling protocols allowing set up and tear down telephone calls, number translation, local number portability, prepaid billing mechanisms or even short message service.

With the telephony evolution appeared the Integrated Services for Digital Network (ISDN), a circuit-switched telephone network system which also provides access to packet switched networks, designed to allow digital transmissions of voice and data over ordinary PSTN network. It offers transmission of voice, video, data and other network services to improve the call service. Digital telephony introduced the base for the VoIP telecommunication [1] [2] [3].

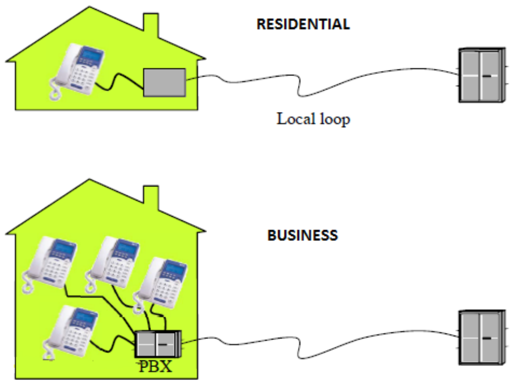


Figure 1: Example of ISDN simple network

What is the VoIP?

First of all, we need to differentiate between IP telephony and VoIP. Internet Protocol Telephony (IP Telephony) is a general term for the technologies that use the Internet Protocol's packet-switched connections to exchange voice, fax and other forms of information. Voice over Internet Protocol (VoIP) is a technology that converts analog voice signals into digital data packets and supports real-time transmission of conversations using the Internet Protocol (IP) [4].

Advantages and disadvantages

The most important reason for VoIP development was to provide voice communication in any place around the world, even where communication may be quite costly. Since

VoIP uses Internet as backbone, it offers substantial savings over traditional long distance telephone calls [5].

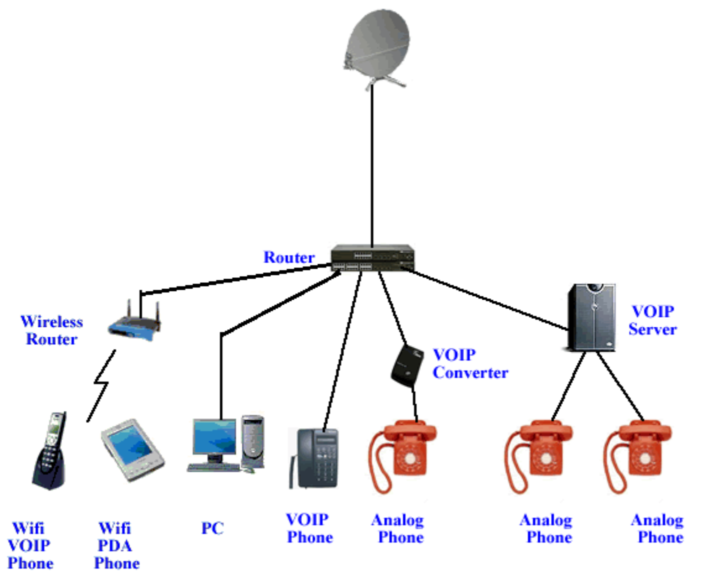


Figure 2: Example of VoIP network

In addition, to use VoIP for a call the only hardware you require besides your computer and an Internet connection, are a sound card, speakers and a microphone (which are quite cheap). But if using a computer for calling may seem uncomfortable, they exists VoIP telephones or even special VoIP devices (such as ATA) , which allows you to call through an ordinary telephone.

.Another improvement of this new technology is that VoIP compresses data packets during transmission, which causes more data to be handled over the carrier. This takes more advantages of the resources and more calls can be handled on one access line.

Rising VoIP also means benefitting from its abundant features: it permits the usage of application and features easy developed which improves the usage and exploitation of the call.

As we can see, there are lots of improvements regarding VoIP, but there are still some problems that need to be solved.

The main disadvantage of VoIP technology is the greater potential for dropped calls and degraded voice quality when the underlying network links are under heavy load. In fact, there are so many factors that could decrease of the Quality of Service (QoS): the broadband connection, the hardware, the service provided by the Internet company, the destination of the call…

VoIP technology also proves to be less robust than that of PSTN as Data, mainly voice, has to be compressed and transmitted and then decompressed and delivered. All this has to be done is a very short amount of time because if this process takes some milliseconds more, the quality of the call may suffer, giving rise to echo.

In the table below we analyze the differences between the use of PSTN lines and the use of VoIP technology [6].

Dedicated Lines.	All channels carried over one Internet connection.
Each line is 64kbps.	Compression can result in 10kbps.
Features such as call waiting, Caller ID and so on are usually available at an extra cost.	Features such as call waiting, Caller ID and so on are usually included free with service.
Can be upgraded or expanded with new equipment and line provisioning.	Upgrades usually requires only bandwidth and software upgrades.
Long distance is usually per minute or bundled minute subscription.	Long distance is often included in regular monthly price.
Hardwired landline phones (those without an adapter) usually remain active during power outage.	Lose power, lose phone service without power backup in place.

Table 1: PSTN Versus VoIP: A Feature Comparison

2.1 VoIP Protocols

Voice over IP has been implemented in various ways using both proprietary protocols (communication protocol owned by a single organization) and protocols based on open standards (standard that are public available).

Some examples of these protocols are H.323, Media Gateway Control Protocol (MGCP), Session Initiation Protocol (SIP), Media Gateway Control or H.248 (Megaco), Inter-Asterisk eXchange (IAX), Jingle XMPP or Skype protocol.

We will focus our attention in the VoIP more important protocols and the ones we will use [7].

2.1.1 H.323

H.323 is an International Telecommunications Union (ITU) standard provides specification for computers, equipment and services for multimedia communication over packet-based networks. It addresses call signaling and control, multimedia transport and control, and bandwidth control for point-to-point and multi-point conferences.
H.323 call signaling is suited for transmitting calls across networks using a mixture of IP, PSTN and ISDN. It is based on the Internet Engineering Task Force (IETF) Real-Time Protocol (RTP) and Real-Time Control Protocol (RTCP), with additional protocols for call signaling, and data and audiovisual communications [8] [9] .

Architecture

The H.323 protocol defines a protocol architecture based in different elements that work together in order to deliver rich multimedia communication capabilities. Those elements are listed below:

Terminals: Are the most fundamental elements in any H.323 system, they are the devices that users would encounter. They could be a simple IP phone, a softphone or a powerful high-definition videoconferencing system.

Gateways: devices that enable communication between H.323 networks and other networks, such as PSTN or ISDN networks. If one part in a conversation is using a terminal that is not an H.323 terminal, then the call must pass through a gateway in order to enable both parties to communicate.

Multipoint Control Unit (MCU): conference bridge capable of mixing or switching video, in addition to the normal audio mixing done by a traditional conference bridge. It is composed of two logical entities: the Multipoint Controller (MC) and the Multipoint Processor (MP).

Gatekeeper: optional component in the H.323 system used for admission control and address resolution. The gatekeeper may allow calls to be placed directly between endpoints or it may route the call signaling through itself to perform functions such as follow-me/find-me, forward on busy, control endpoint registration, address resolution, admission call control or user authentication. A zone is composed by a single Gatekeeper and all of the devices connected to it.

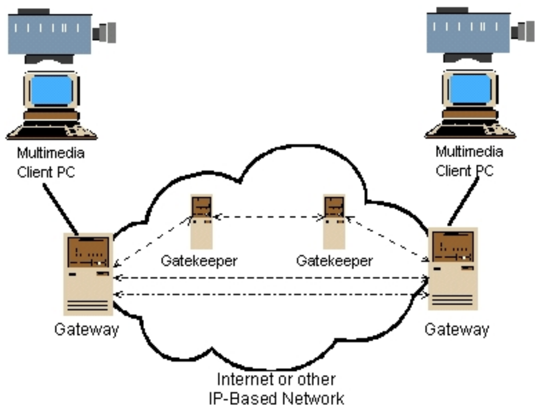


Figure 3: Example of an H.323 architecture

<u>H.323 communication</u>

H.323 protocol is based in other protocols that set up a communication call. The H.225.0 Registration, Admission and Status (RAS) is used between an H.323 endpoint and a Gatekeeper to provide address resolution and admission control services. Once the address is provided, the endpoint will use H.225.0 for call signaling in order to establish communication with the remote entity.

The H.245 protocol is used as a negotiator which interchange messages (request and answer) between both terminals to establish who will be the master and who the slave, the capacities of the participants and the audio and video codecs to be used. When the negotiation finishes the communication channel is opened (IP addresses, port).

Once the communication is settled, terminals start the communication using the RTP/RTCP protocol.

To stop the communication, both terminals can initiate the ending process using H.245 and then H.225 to confirm the connection is closed with the RELEASE COMPLETE message. Finally, the registration of the terminals in the gatekeeper are cleared using RAS protocol.

The Figure 4 that follow below demonstrates the steps the occurs during the communication, when we use the protocol H323**.**

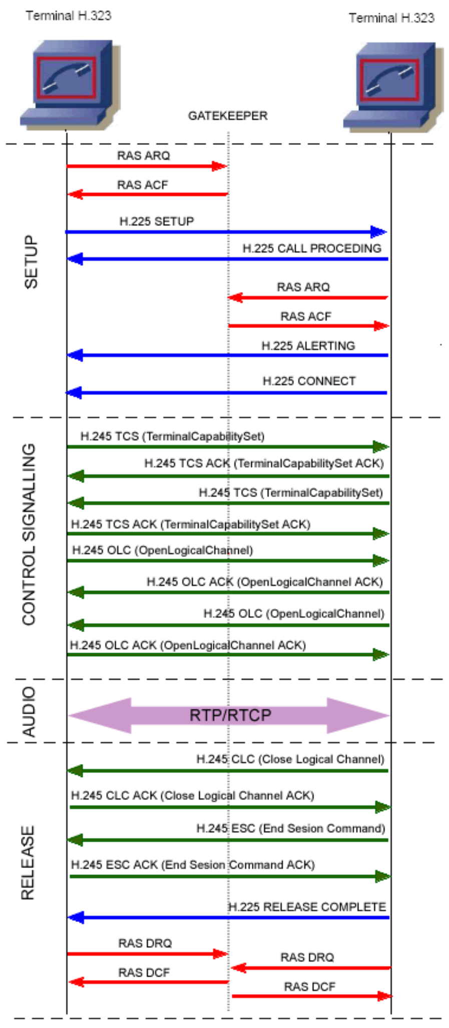

Terminal H.323
Terminal H.323
GATEKEEPER
SETUP
RAS ARQ
RAS ACF
H.225 SETUP
H.225 CALL PROCEDING
RAS ARQ
RAS ACF
H.225 ALERTING
H.225 CONNECT
CONTROL SIGNALLING
H.245 TCS (TerminalCapabilitySet)
H.245 TCS ACK (TerminalCapabilitySet ACK)
H.245 TCS (TerminalCapabilitySet)
H.245 TCS ACK (TerminalCapabilitySet ACK)
H.245 OLC (OpenLogicalChannel)
H.245 OLC ACK (OpenLogicalChannel ACK)
H.245 OLC (OpenLogicalChannel)
H.245 OLC ACK (OpenLogicalChannel ACK)
AUDIO
RTP/RTCP
RELEASE
H.245 CLC (Close Logical Channel)
H.245 CLC ACK (Close Logical Channel ACK)
H.245 ESC (End Sesion Command)
H.245 ESC ACK (End Sesion Command ACK)
H.225 RELEASE COMPLETE
RAS DRQ
RAS DCF
RAS DRQ
RAS DCF

Figure 4: H.323 Communication Example

2.1.2 IAX

The Inter-Asterisk eXchange (IAX) protocol is an application-layer control and media protocol for creating, modifying, and terminating multimedia sessions over Internet Protocol networks [10] [11].

IAX was developed by Mark Spencer for the open source community, to use it with Asterisk Private Branch Exchange (PBX) and is targeted primarily at VoIP call control, but it can also be used with streaming video or any other type of multimedia.

IAX uses the User Datagram Protocol (UDP) data stream on a static port (usually on port 4569) to communicate between endpoints and multiplex signaling and media flow. In addition, it greatly simplifies Network Address Translation (NAT), eliminating the need for other protocols to work around NAT, and simplifying network and firewall management. This is in contrast to SIP, H.323 and MGCP which use an out-of-band RTP stream to deliver information.

In the picture below, we can see the simplicity of a communication exchange using IAX.

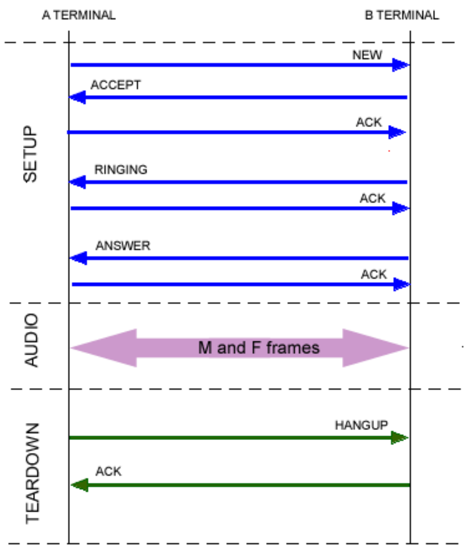


Figure 5: IAX communication example

IAX employs a compact encoding that decreases bandwidth usage and is well suited for Internet telephony service. In fact, IAX2 supports trunking: multiplexing channels over a single link. When trunking, data from multiple calls are merged into a single stream of packets between two endpoints, reducing the IP overhead without creating additional latency. This is advantageous in VoIP transmissions, in which IP headers use a large percentage of bandwidth.

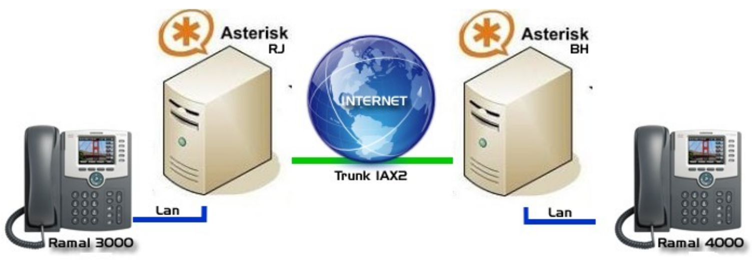


Figure 6: IAX example network

Resuming, the primary goals for IAX were to minimize bandwidth used in media transmissions, with particular attention drawn to control individual voice calls, and to provide native NAT transparency.

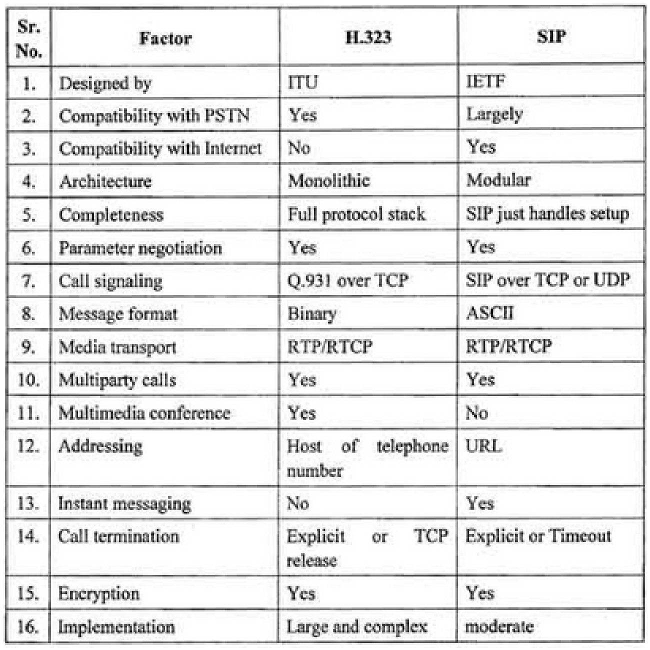
2.1.3 SIP

Session Initiation Protocol (SIP) is the protocol we used to implement our Softphone. It is the best option for what we intended to do. We will deep in SIP in the next chapter, just see now why this protocol has been chosen to implement this project.

H.323 vs SIP:

There are quite differences between the two protocols. For example, as H.323 is more complex and needs many messages for similar functions, SIP protocols has logically numbered response and uses a smaller set of messages for the same functionality. H.323 is less flexible than SIP, since H.323 has to add specific fields or parameter values in the signaling while SIP, combined with SDP, provides this functionality transparently. Sip is also more compatibility among versions, while H.323 requires full backward compatibility.

Below, we can see a comparison between both protocols:



Sr. No.	Factor	H.323	SIP
1.	Designed by	ITU	IETF
2.	Compatibility with PSTN	Yes	Largely
3.	Compatibility with Internet	No	Yes
4.	Architecture	Monolithic	Modular
5.	Completeness	Full protocol stack	SIP just handles setup
6.	Parameter negotiation	Yes	Yes
7.	Call signaling	Q.931 over TCP	SIP over TCP or UDP
8.	Message format	Binary	ASCII
9.	Media transport	RTP/RTCP	RTP/RTCP
10.	Multiparty calls	Yes	Yes
11.	Multimedia conference	Yes	No
12.	Addressing	Host of telephone number	URL
13.	Instant messaging	No	Yes
14.	Call termination	Explicit or TCP release	Explicit or Timeout
15.	Encryption	Yes	Yes
16.	Implementation	Large and complex	moderate

Table 2: H.323 versus SIP

Resuming, even though H.323 has more enterprise oriented and campus scale products deployed, SIP provides long terms benefit which are related to and affect time to market, extensibility, multi-party service flexibility, ease of interoperability and complexity of development.

IAX vs SIP:

There are some differences between IAX and SIP protocol. For example, IAX is more efficient on the wire than RTP for any number of calls, any codec. It also is information-element encoded rather than ASCII encoded, which makes implementations substantially simpler and more robust to buffer overrun attacks. Also, as explained before, IAX's unified signaling and audio paths permit it to transparently navigate NAT's and provide a firewall administrator only a single port to have to open to permit its use, which is easer than working with SIP. IAX's authenticated transfer system allows to transfer audio and call control off a server-in-the-middle in a robust fashion so, if the two endpoints cannot see one another for any reason, the call continues through the central server.

In the other hand, SIP offers more Bandwidth indication and more extensibility. It also offers new codecs and Video telephone, which, for our application purpose, it is very important.

This time we are not going to implement a Video Conference Softphone but, as an open application, it can be implemented later. This is the main reason we used SIP instead of IAX.

3. SESSION INITATION PROTOCOL

The SIP protocol (Session Initiation Protocol), defined in the RFC 3261, is a signaling communication protocol, used to establish, modify or terminate two-party (unicast) or multiparty (multicast) sessions. It is complemented with other protocols such as SDP (Session Description Protocol), RTP (Real-Time Transport Protocol) or RSP (Real-Time Stream Protocol) [12] [13].

It is used to control multimedia communication sessions such as voice and video calls over Internet Protocol networks, video conferencing, streaming multimedia distribution, instant messaging, presence information, file transfer, fax over IP and online games.

SIP is an application layer protocol independent of the underlying transport layer. In fact, it can run on Transmission Control Protocol (TCP), User Datagram Protocol (UDP) or Stream Control Transmission Protocol (SCTP). It is a text-based protocol, which incorporates many elements of HTTP and SMTP.

SIP uses specific Uniform Resource Locators (URLs) which are close to an e-mail address, but with a different format.

- *sip:user@my_company.com*
- *sip:+1-972-555-1234@my_company.com; user=phone*

The user is also accessible through a gateway or in the Virtual Network (VNET).

- *sip:1234@my_company.com; user=phone; phone-context=VNET*

In fact, to use addresses close to e-mail formats and telephone numbers facilitates the interoperability between systems (telephony and Internet), which is a very important feature of this protocol.

3.1 Capabilities

As said, the Session Initiation Protocol is a signaling communication protocol, widely used for controlling multimedia sessions such as voice and video calls over IP networks. SIP can implement advanced services in order to improve and facilitate the communication such as *Call Hold* or *Call Waiting*. It can also provide IN (Intelligent Network) applications such as *Centrex* or *Voice VPN* (Virtual Private Network) or use user profiles to indicate how the calls should be treated by the network.
SIP has been designed to help establish a conference, making possible the use of multicast. It offers mobility facilities: it can be *terminal mobility*, the terminal is moving between sub-networks, *personal mobility*, the user can change from one terminal to another, or *service mobility*, possible to access the same service from different terminals. It also permits the use of presence status (*online, offline*), location (*office, home, mobile*), call status (*busy, idle*), availability (*available, in a meeting*) or preferred mean (*text, voice, video, e-mail, instant messaging*).

SIP Protocol offers instant messaging, which is the textual exchange of information in real time. The most part of this project is based in this part.

3.2 SIP Elements

The SIP protocol defines server network elements. Although two SIP endpoints can communicate without any intervening SIP infrastructure (the protocol is peer-to-peer), this approach is often impractical for a public service. RFC 3261 defines these server elements [14].

User-Agents: logical network end-point used to create or receive SIP messages and manage a SIP session. UA contains information about the User Agent Client (UAC), the logical entity originating the session. The UAC creates a new request and the User

Agent Server (UAS) receives it and generates a response. These roles of UAC and UAS only last for the duration of a SIP transaction.
A SIP phone is a SIP User Agent that provides the traditional call functions. SIP phones may be implemented as a hardware device or as a softphone but, as vendors increasingly implement SIP as a standard telephony platform, the distinction between hardware-based and software-based SIP phones is being blurred and SIP elements are implemented in the basic firmware functions of many IP-capable devices.

Proxy server: intermediary entity that acts as both a server (UAS) and a client (UAC) for the purpose of making requests on behalf of other clients. A proxy server primarily plays the role of routing: ensure that a request is sent to another entity "closer" to the targeted user. Proxies are also useful for enforcing policy or, if necessary, rewriting specific parts of a request message before forwarding it.

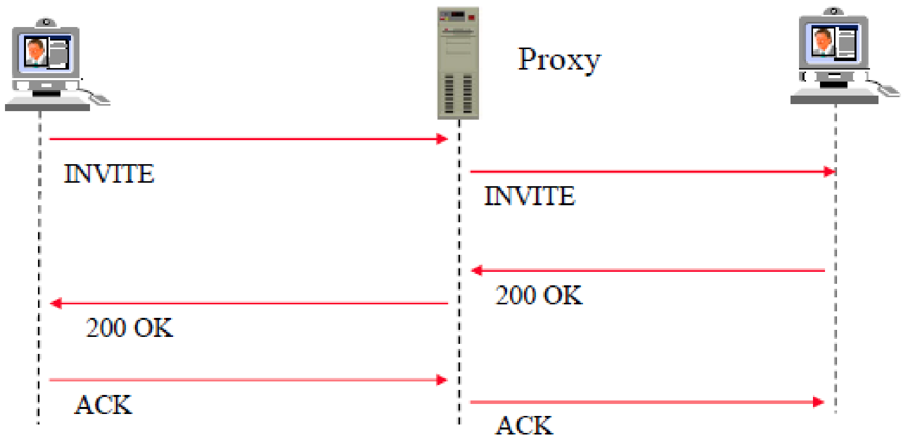


Figure 7: Example of a Proxy server working as a routing server

Registrar: SIP endpoint that accepts REGISTER requests and places the information it receives into a location service for the domain it handles. The location service links one or more IP addresses to the SIP URI of the registering agent. More than one UA can register at the same URI. SIP registrars are logical elements, and are commonly co-located with SIP proxies, but it is also possible and often good for network scalability to place this location service with a redirect server.

Redirect Server: when a User Agent Server generates answer a request with a *3xx* SIP Message (Redirection), the Redirect Server informs the client to contact an alternate set of URIs. The Redirect Server allows Proxy Servers to direct SIP session invitations to external domains.

Session Border Controller: serves as middle boxes between UA and SIP servers for various types of functions, including network topology hiding and assistance in NAT traversal.

Gateway: interfaces a SIP network to other networks, such as the PSTN, which uses different protocols or technologies.

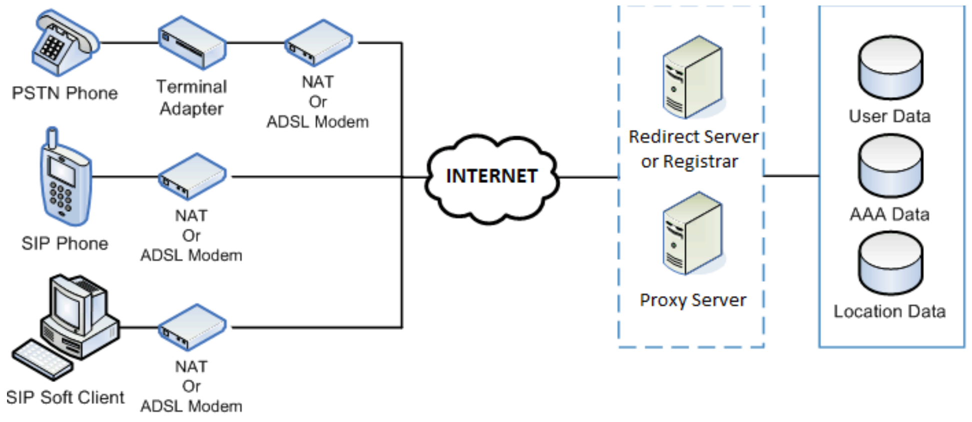


Figure 8: SIP Network example

3.3 Messages

There are two types of messages: the Request Messages (also called METHODS) and the Response Messages.

The first line of a SIP message indicates the method or response type and the SIP version.
Then there is a header that includes all the information required to send the packet, followed by an empty line that separates it from the message body.

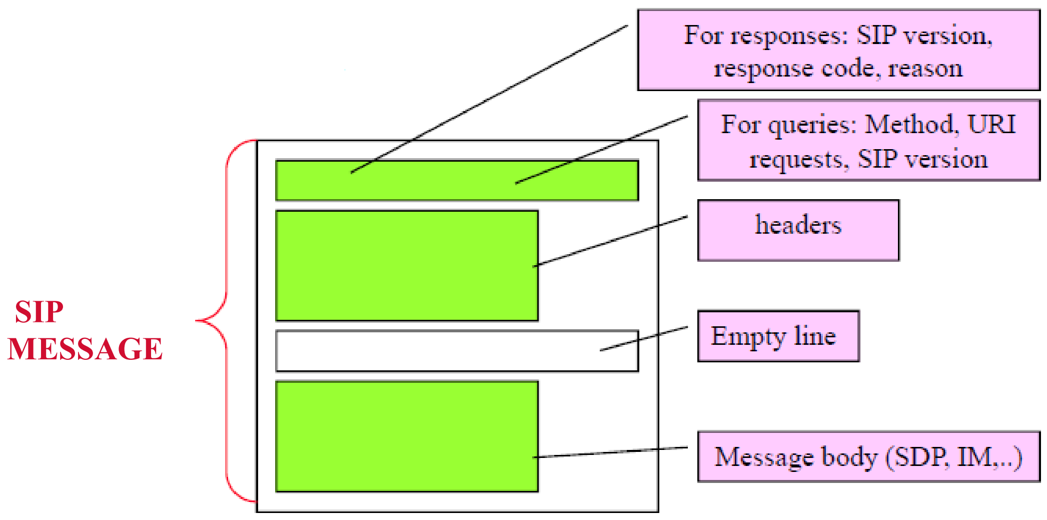
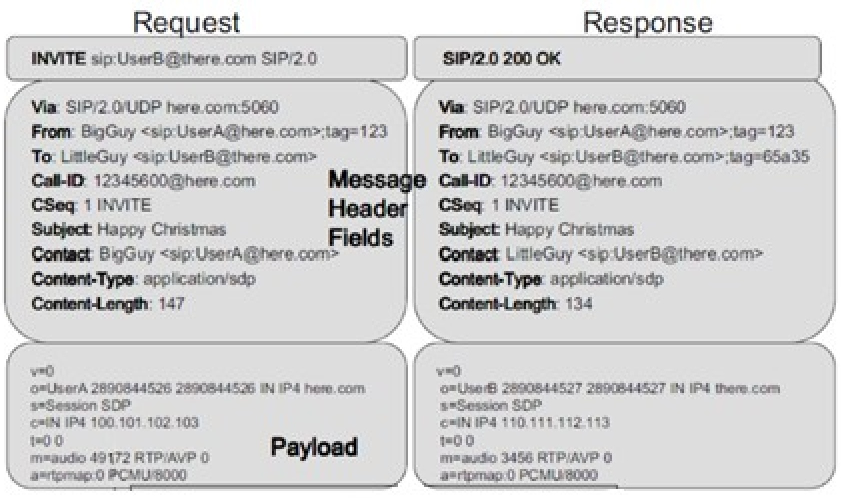

SIP MESSAGE
For responses: SIP version, response code, reason
For queries: Method, URI requests, SIP version
headers
Empty line
Message body (SDP, IM,..)

Figure 9: Parts of a SIP Message

In the picture above we can see all the parts of the message: the indication line, the header, the empty line and the payload used for an SDP message information.


Request
Response
Message Header Fields
Payload

```
INVITE sip:UserB@there.com SIP/2.0

Via: SIP/2.0/UDP here.com:5060
From: BigGuy <sip:UserA@here.com>;tag=123
To: LittleGuy <sip:UserB@there.com>
Call-ID: 12345600@here.com
CSeq: 1 INVITE
Subject: Happy Christmas
Contact: BigGuy <sip:UserA@here.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=UserA 2890844526 2890844526 IN IP4 here.com
s=Session SDP
c=IN IP4 100.101.102.103
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

```
SIP/2.0 200 OK

Via: SIP/2.0/UDP here.com:5060
From: BigGuy <sip:UserA@here.com>;tag=123
To: LittleGuy <sip:UserB@there.com>;tag=65a35
Call-ID: 12345600@here.com
CSeq: 1 INVITE
Subject: Happy Christmas
Contact: LittleGuy <sip:UserB@there.com>
Content-Type: application/sdp
Content-Length: 134

v=0
o=UserB 2890844527 2890844527 IN IP4 there.com
s=Session SDP
c=IN IP4 110.111.112.113
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Figure 10: Example of a SIP Message

SIP uses Requests and Responses to communicate and establish a call session. In the list below we can see the different types of SIP Requests and SIP Responses.

SIP Requests:

INVITE	= Establishes a session
ACK	= Confirms an INVITE request
BYE	= Ends a session
CANCEL	= Cancels establishing of a session
REGISTER	= Communicates user location (host name, IP)
OPTIONS	= Communicates information about the capabilities of the calling and receiving SIP phones

PRACK	= Provisional Aknowledgement
SUBSCRIBE	= Subscribes for Notification from the notifier
NOTIFY	= Notifies the subscriber of a new event
PUBLISH	= Publishes an event to the Server
INFO	= Sends mid-session information
REFER	= Asks the recipient to issue call transfer
MESSAGE	= Transports Instant Messages
UPDATE	= Modifies the state of a session

SIP Responses:

1xx = informational responses
2xx = success responses
3xx = redirection responses
4xx = request failures
5xx = server errors
6xx = global failures

Some examples of the Responses code are:

100 Trying
180 Ringing
181 Call is Being Forwarded
301 Moved Permanently
302 Moved Temporarily
400 Bad Request
401 Unauthorized
482 Loop Detected
486 Busy Here
500 Server Internal Error
600 Busy Everywhere

In a SIP message, there are different headers information fields that carry the information (destination, source, contact…). The most common headers are:

Via: tracks the request route in order to follow the same path in the response message. It indicates the SIP version, the transport protocol, the provider of the packet and the port it uses.

To: indicates the receiver direction

From: indicates the sender direction

Call-ID: contains a globally unique identifier for this call, generated by the combination of a random string and the softphone's host name or IP address.

CSeq: contains an integer and a method name. The CSeq increments by one for each RTP data packet sent, and the receiver uses it to detect packet loss and to restore packet sequence. The initial value of the sequence number should be random to make known-plaintext attacks on encryption more difficult.

Contact: provides a SIP or SIP URI used to contact that specific instance of the UA for subsequent requests.

Content-Disposition: describes how the message body or a message body part has to be interpreted by the UAC or UAS.

Content-Type: indicates the media type of the message-body. If the body is empty, it indicates that the body of the specific type has zero length.

Content-Encoding: modifies the "media-type". Its value indicates what additional content coding has been applied to the entity-body, and what decoding mechanisms must be applied.

Content-Length: indicates the size of the message-body.

Timestamp: reflects the sampling instant of the first octet in the RTP data packet.

3.3 SIP Communication

<u>How to start a SIP session</u>

To start a SIP session, the first thing we need to do is a request of REGISTER. The REGISTER method registers the SIP endpoint at the Registrar Server. It contains the IP address of the client and other information about the contact (mobile phone, address, work phone…). The register has to be updated periodically because if the IP address is not accessible, it is the way that others users can contact this specific user.

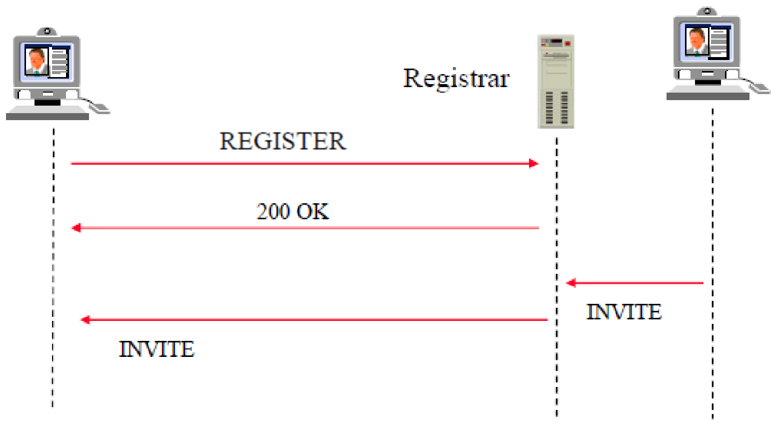


Figure 11: User register example

After the registration, the communication continues with an INVITE packet: it is a request to establish a call. There are different headers in the INVITE packet, the most important are:

Proxy-Authorization: stores the information of the authentication consisting of credentials containing the authentication information of the User Agent for the proxy.

Allow: lists the set of methods supported by the UA generating the message.

Supported: enumerates all the capabilities of the users.

Content-Type: contains a description of the message body.

The **Message Body** informs about the SDP Protocol. It also indicates whether it uses rtpmap (defines a mapping from RTP payload codes to a codec name, clock rate, and other encoding parameters), sendrecv (we can both send and receive media), or fmtp (defines parameters that are specific for a given format code).

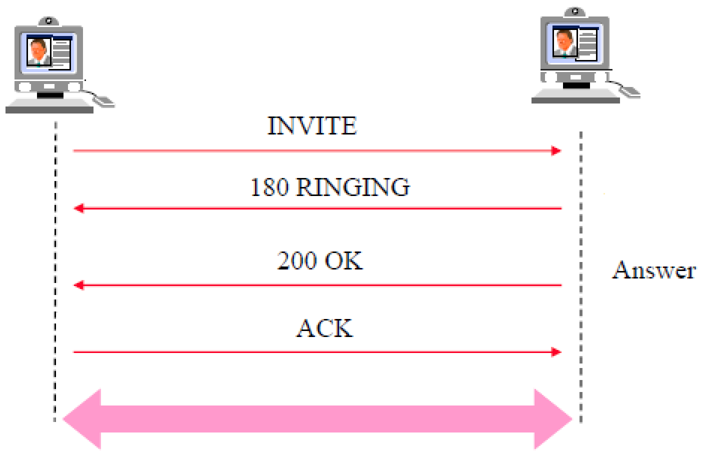


Figure 12: Simple example of a call

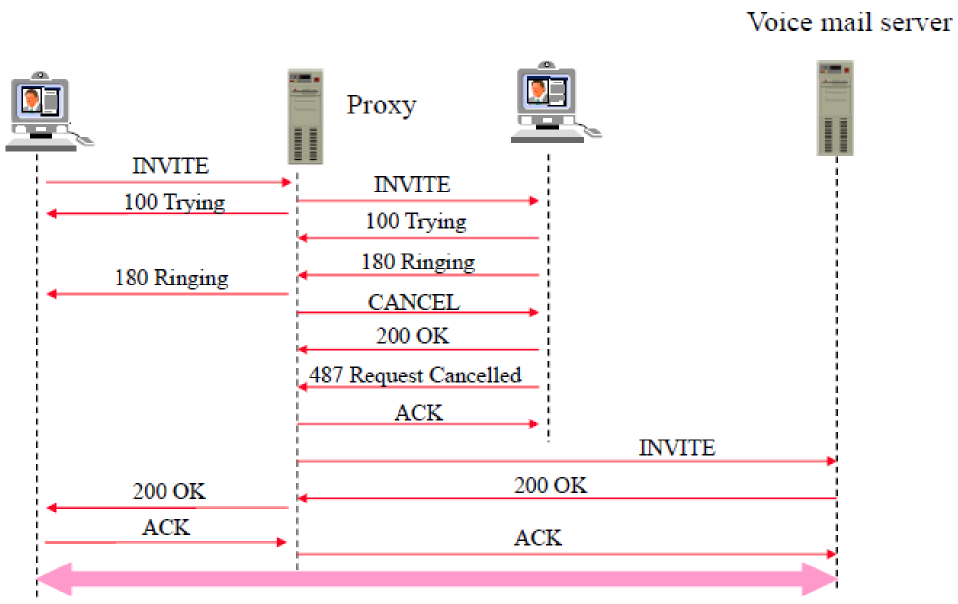


Figure 13: INVITE using a Proxy Server to redirect the message

How to end a SIP session

A SIP session can be closed with a BYE message or cancelled with a CANCEL message. When a call has been previously established with INVITE/200/ACK, it needs a BYE message to do an end-to-end termination.

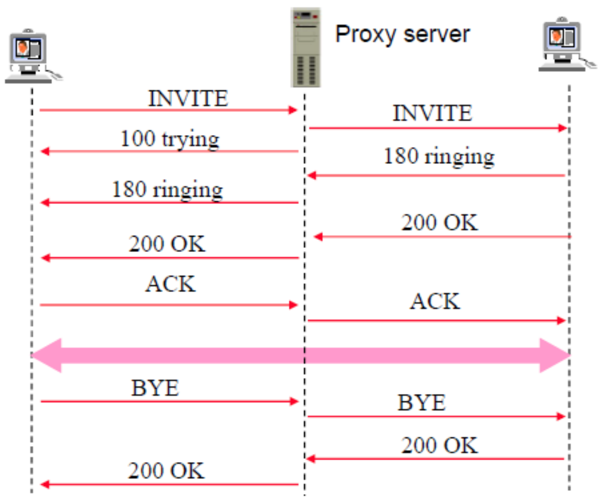


Figure 14: Session terminated with a BYE

If the session is ended with a CANCEL message, it will stop the session set-up. It can occur when a problem during the set is happening.

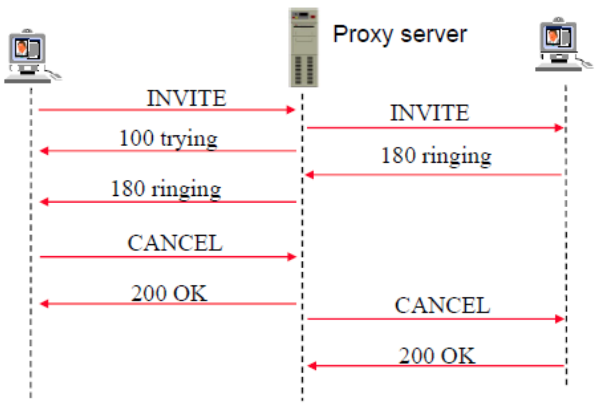


Figure 15: Session ended with a CANCEL

3.4 Support Protocols

As we already said, SIP Protocols uses other protocols to support a communication. It uses Transport Layer protocols such as TCP, UDP or SCTP; and Application Layer protocols such as RTP, SDP or SRTP.

3.4.1 SDP

The Session Description Protocol (SDP) is a protocol defined in RFC 4566 and intended for describing multimedia communication session for the purposes of session announcement, session invitation and parameter negotiation. It dos not deliver media itself, but it is used for negotiation between end-points of media type [15][16].

SDP uses INVITE or 200 OK as messages to propose a communication. The replay (200 OK or ACK) are the confirmation or rejection of the proposal. SDP is usually inside the SIP command.

SDP negotiates the codecs used in a communication. It sends a list of codecs the Client can use, and the server answers indicating which one is better for both UA.

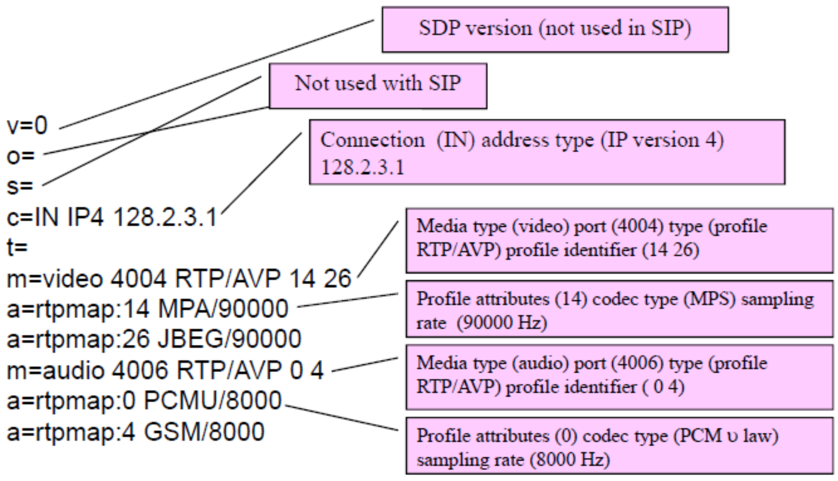

```
v=0
o=
s=
c=IN IP4 128.2.3.1
t=
m=video 4004 RTP/AVP 14 26
a=rtpmap:14 MPA/90000
a=rtpmap:26 JBEG/90000
m=audio 4006 RTP/AVP 0 4
a=rtpmap:0 PCMU/8000
a=rtpmap:4 GSM/8000
```
SDP version (not used in SIP)
Not used with SIP
Connection (IN) address type (IP version 4) 128.2.3.1
Media type (video) port (4004) type (profile RTP/AVP) profile identifier (14 26)
Profile attributes (14) codec type (MPS) sampling rate (90000 Hz)
Media type (audio) port (4006) type (profile RTP/AVP) profile identifier (0 4)
Profile attributes (0) codec type (PCM υ law) sampling rate (8000 Hz)

Figure 16: Example of an SDP Proposal Message

3.4.2 RTP

The Real-Time Transport Protocol (RTP) is an Application Layer protocol, defined in RFC 3550 and designed to deliver audio and video over IP networks [17][18].

It is used extensively in communication and entertainment systems that involve streaming media. It supports the Sip Protocol: when a SIP communication is settled, RTP delivers the information.

RTP is a protocol designed for end-to-end, real-time transfer of data. It provides facilities for jitter compensation and detection of out of sequence arrival in data. RTP also allows multicast communication: data transfer to multiple destinations.

The most RTP implementations are built on UDP because they normally favors timely delivery of information over some packet loss.

3.4.3 TCP

The Transmission Control Protocol (TCP) is defined by IETF RFC 5681 and it is, along with UDP, one of the core protocols of the IP Protocol.
TCP is a Transport Layer protocol, oriented to connection. It provides communication service between an application program and the IP: when an application program needs to send a lot of data across Internet, it can it request to TCP and let TCP handle the IP details.
TCP was designed as a reliable and ordered protocol. It uses error-control and error-checked delivery between programs running on different computers. It corrects losses, duplication and out of order datagram. It even helps minimize network congestion to reduce the occurrence of the other problems. It uses a three way handshake so, for these reasons, TCP is useful for accurate delivery rather than timely delivery such as for real-time applications [20] [21].

3.4.4 UDP

The User Datagram Protocol (UDP), defined in RFC 768, sends datagrams on an IP network without prior communications to set up special transmission channels or data path. It uses a simple transmission model with a minimum mechanism, without handshaking dialogues or other communication set up methods: it has connectionless transport mechanism. With this characteristic, UPD is a fast transport protocol but, in opposition of this, it offers no guarantee of delivery, ordering or duplicate protection. UDP protocol is useful when error checking and correction are not necessary, but a real-time system is needed.
UDP is transaction-oriented, simple and stateless, using less resources. Furthermore, it has a lack of retransmission delays, which makes it useful for real-time streaming media applications. UDP works also well in unidirectional communication, being suitable for broadcast information [22] [23].

3.5 ASTERISK

Although I didn't mesh directly with Asterisk, to do this project it was important to me knowing what Asterisk is and how it works. In fact, SIP protocol communicates directly with an Asterisk server, which, explained in few words, recognize the messages and sends them to the phone company [24] [25].

Asterisk is a software implementation (open source) of a telephone private branch exchange (PBX). It was created in 1999 by Mark Spencer, when he released the initial code under General Public License (GPL) open source license.
Nowadays, Asterisk is a framework for building multi-protocol, real-time communications applications and solutions. As it powers IP PBX systems, Asterisk software includes many features: voice mail, conference calling, interactive voice response (phone menus) and automatic call distribution.

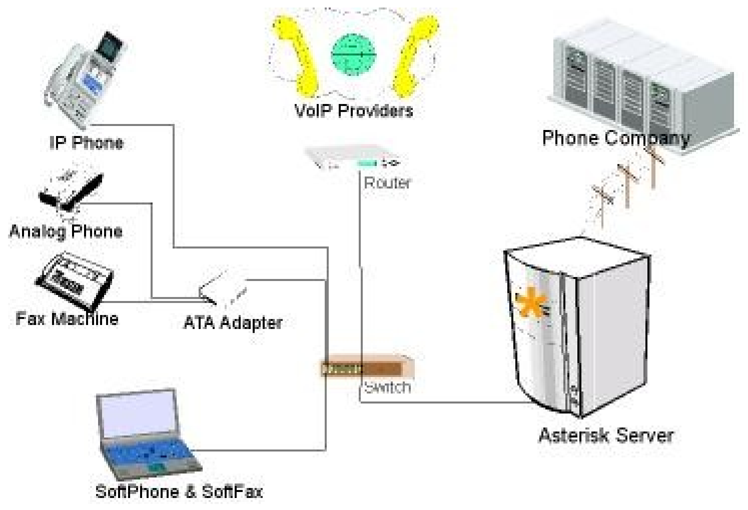


Figure 17: Example of an Asterisk PBX

Users also can create new functionality by writing dial plan scripts in several of Asterisk's own extensions languages, by adding custom loadable modules written in C or by implementing Asterisk Gateway Interface (AGI) programs using any programming language. As a matter of fact, it runs on Linux, BSD, Windows (emulated) and OS X and provides all of the features you would expect from a PBX and more. Asterisk does voice over IP in four protocols, and can interoperate with almost all standards-based telephony equipment using relatively inexpensive hardware.

It powers IP PBX systems, VoIP gateways, conference servers and other custom solutions. In fact, it can be the basis for a complete phone system, or used to enhance or to extend an existing system.

G4Flex uses Asterisk to create PBX and VoIP systems. The main reason is that Asterisk is open source and free to use, so you can get in it, see how it works and, if you want or you need, make some changes. Furthermore, Asterisk is flexible and lets you define the solution that truly fits your requirements.

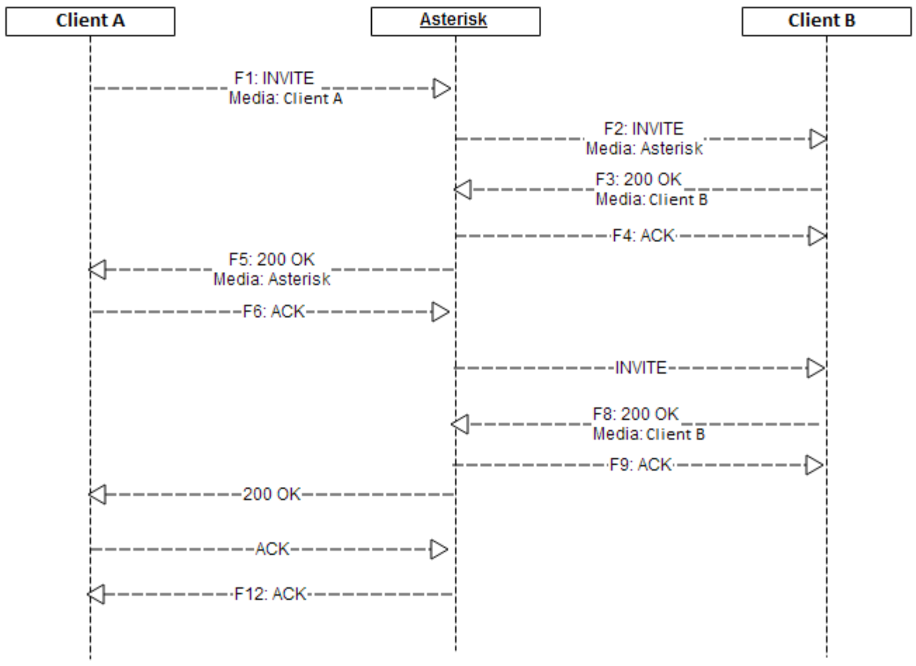


Figure 18: Asterisk exchange message

4. UDP MESSAGE FOR A SIP JAVA SOFTPHONE

G4Flex Phone is a softphone implemented by Renan Pinto de Paula in his Computer Science Final Project *'Desenvolvimento de aplicação multimídia com utilização do protocolo SIP'* [26], during winter 2012 in UNIFOR, in collaboration with G4Flex Business Services. The objective was to create an own Softphone which was perfectly adapted to the willing of the G4Flex enterprise. As the enterprise works with Asterisk servers, the softphone was created using SIP protocol, in order to communicate directly with the servers.

G4Flex, located in Rua Carlos Vasconcelos, 1706, Meireles, Fortaleza (Ceará – Brasil), is an enterprise that uses the technology and the modernization of technology communication in order to increase the productivity and uses of the resources of their clients. It adopts the VoIP technology and tries to expand it over their clients. For these reason, G4Flex Phone offers a personal and distinguished VoIP service to their clients.

Once the work [26] was finished, the G4Flex Phone was able to establish and do a call between two terminals connected to an Asterisk server. The design, as we can see in Figure 19, is simple, but is very useful. It has three windows: *Teclado*, one used to call; *Histórico*, to see the call's history; *Contatos*, one to see the list of contacts.

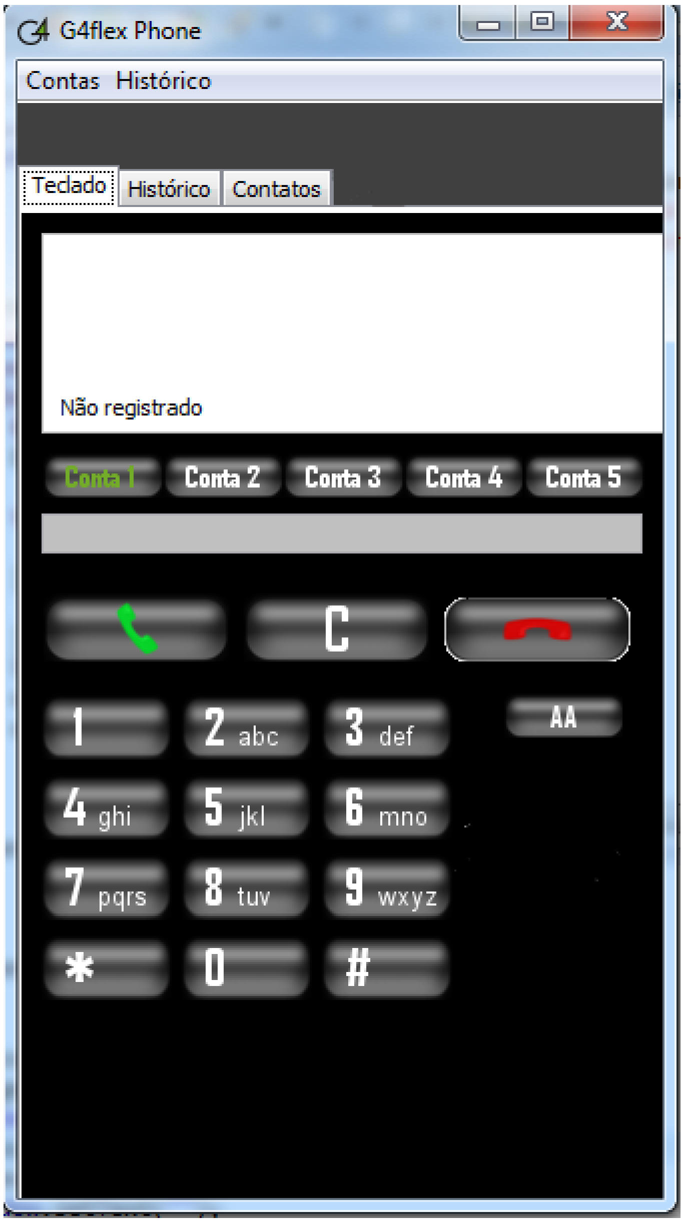


Figure 19: Renan's G4Flex Phone

My project, after receiving the G4Flex Phone [26], consisted to add a new feature to the softphone: is should be able to send a message through the users connected to the same network. The first idea was to create a chat using SIP Protocol, but after some investigations and the impossibility of using Asterisk servers in the university's lab, we decided to do the connection for the text message with UDP, as it is compatible with Asterisk and they can work well together.

So finally, we created a softphone that send UDP messages working in parallel with the SIP voice connection.
The application was created using Eclipse, as the previous softphone was already created using it. I used JavaScript for the programming part [27] [28] [29] [30] and the Eclipse Window Builder for the design [31].

4.1 DESIGN

The first thing I had to deal with was studying and understanding how G4Flex Phone [26] worked. As I didn't do this softphone from the beginning, I had to work always dealing with this work [26].
We need to change the Softphone's design in order to add a *messages workout*. The best option we found, was to create a new button which opened a new window from where all the message work was done. It was the best solution we found if we didn't want to change all the softphone design, as it was not designed to send messages.

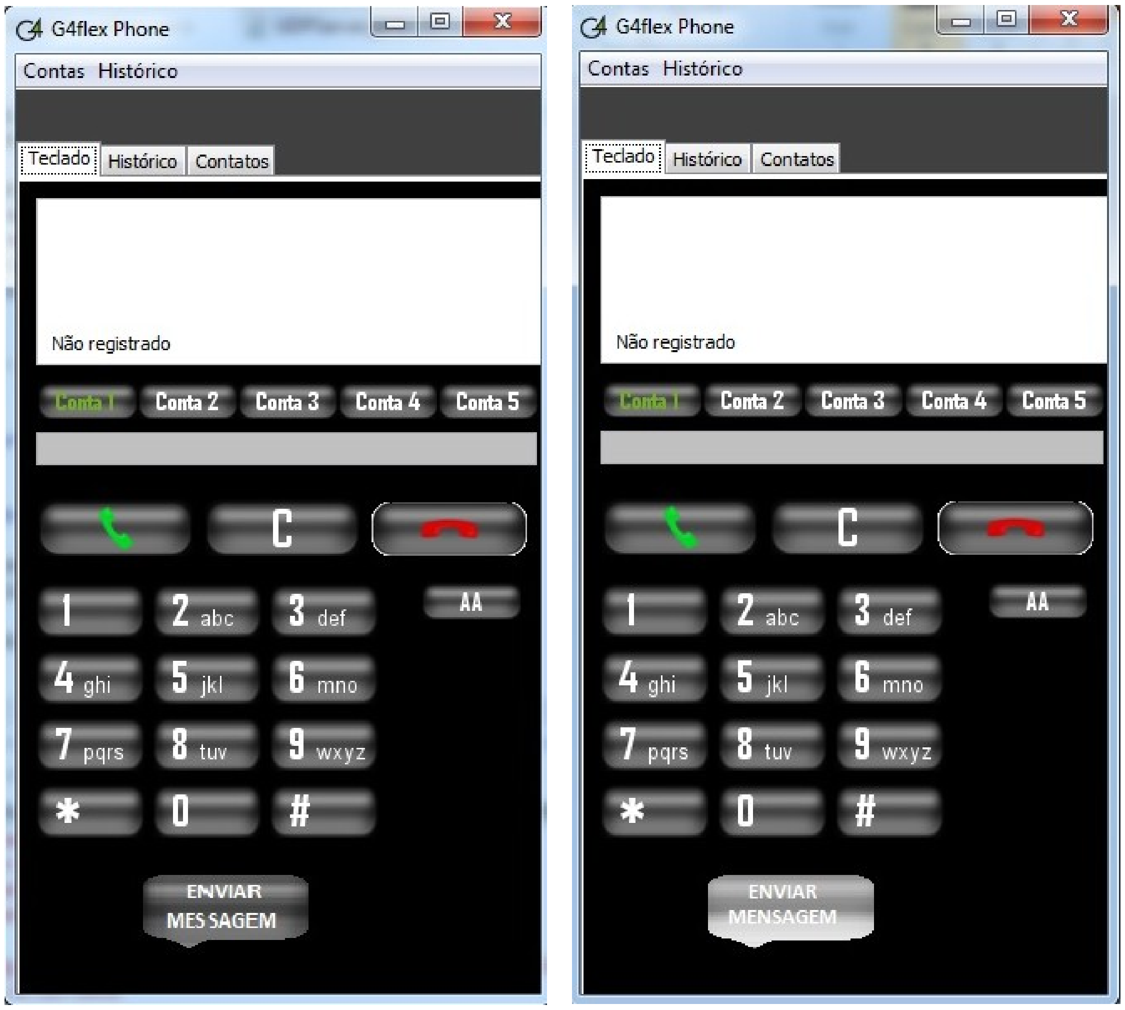


Figure 20: redesigned G4Flex Phone

As we can see in the right picture, when the *Enviar Messagem* button is clicked with the mouse, the button changes.

When the button is clicked, a new window is opened, where the user writes the user destination address. The port for the UDP connection can be also specified, otherwise it uses the 9999 port.

Once the first user writes the user's address and the port and the Conetar button is clicked, the Connect Window is closed and a new Chat Window opens. At this point, it sets up the communication between both clients and the Chat session starts.

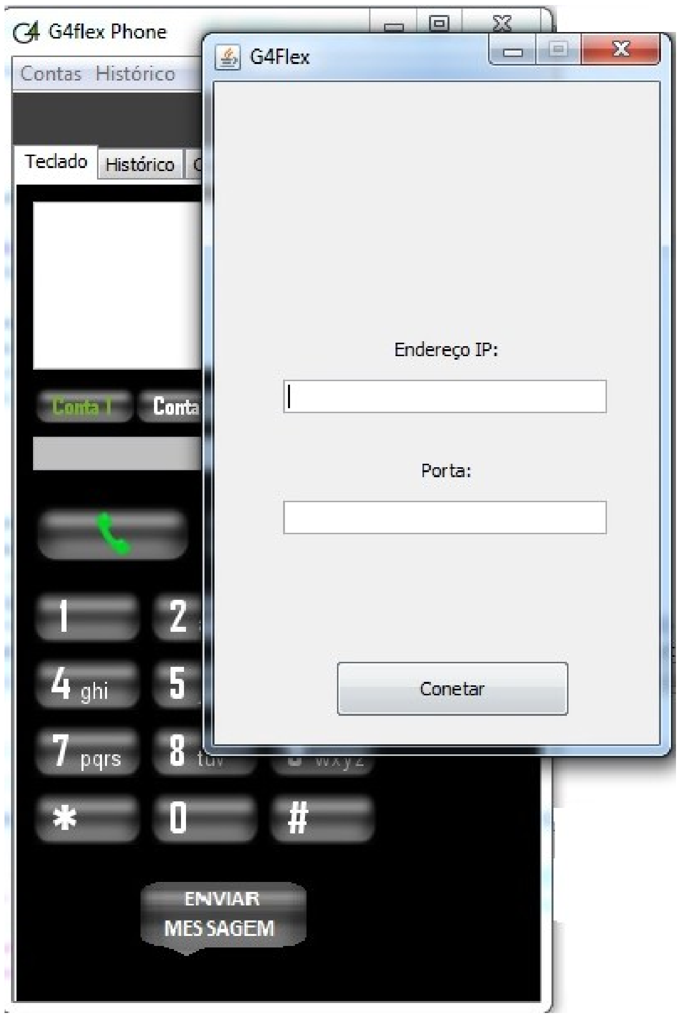


Figure 21: Connect Window

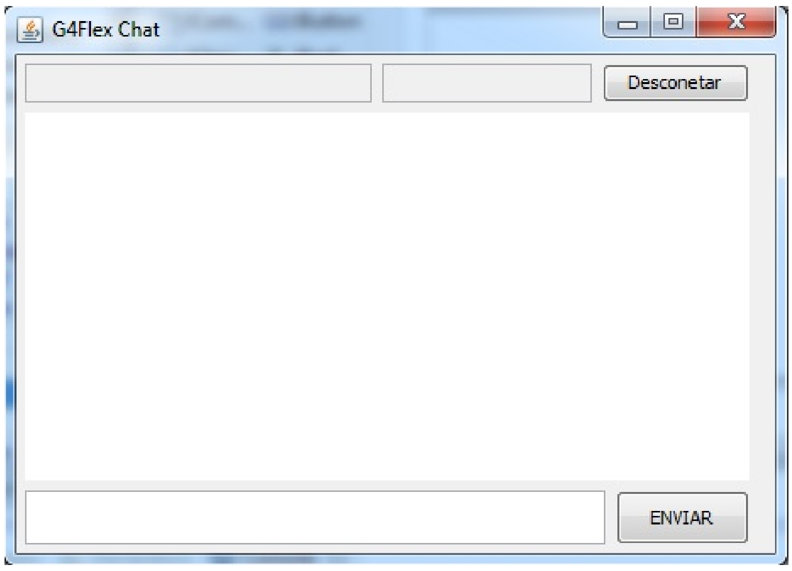


Figure 22: Chat Window

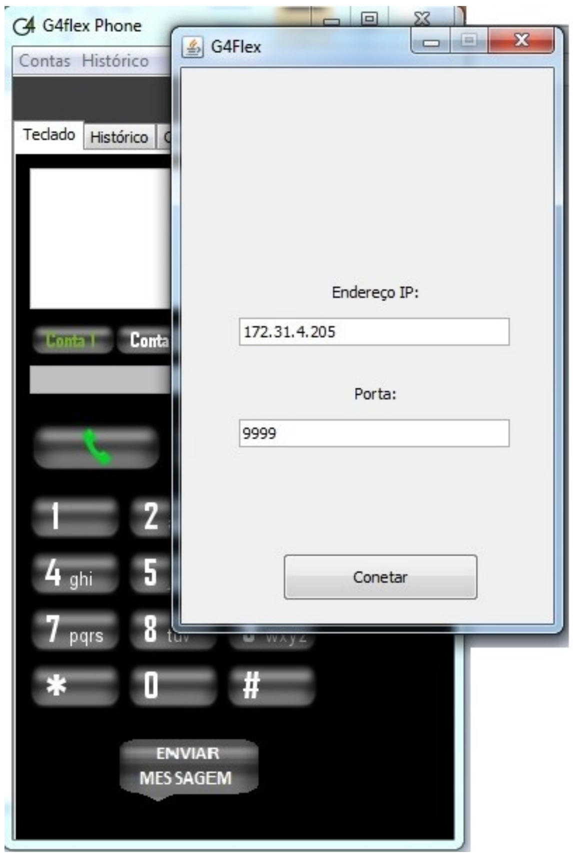
As we can see in the picture above, the Chat Window contents a text window where the user writes the message and the *Enviar* button, to send the message. As the messages are sent and received, they are written in order in the big white window in the center.
The Text Labels in the top of the Window indicates the User connected to and the port used. To close the window and finish the communication, the user only needs to click the *Desconetar* button.

This is the basic design we created for the Softphone. It is simple but clear, easy to understand for anyone who never meshed with any Softphone or messaging technology. In fact, it is the idea of this softphone, as it is intended to people that may be not used to this kind of technologies.

4.2 PERFORMANCE

The part of the G4Flex Phone created works with sockets. In fact, as it is UDP connection, it was decided to create a server and a client. In order to do a terminal-to-terminal connection, all users are both server and client. There are two different threads working in parallel: one running the server and the other running the client.

As the G4Flex Phone is initiated, the server starts to run and it is always listening for any message to arrive. When a user decides to send a message, he opens the Connect Window and write the IP of the other terminal, the one who will receive the message.



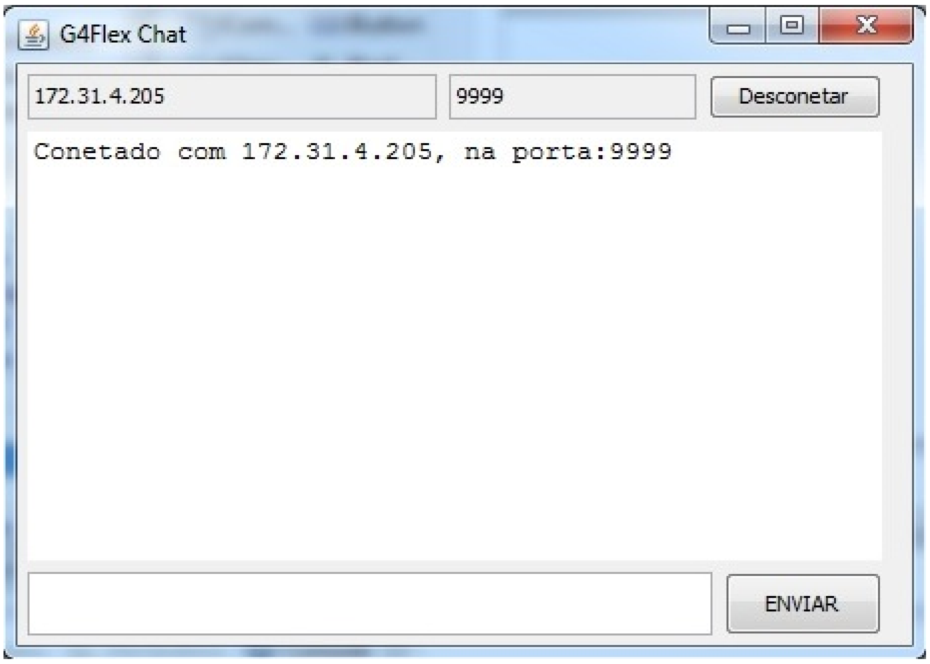
Endereço IP: 172.31.4.205

Porta: 9999

Figure 23: User connecting to 172.4.31.205

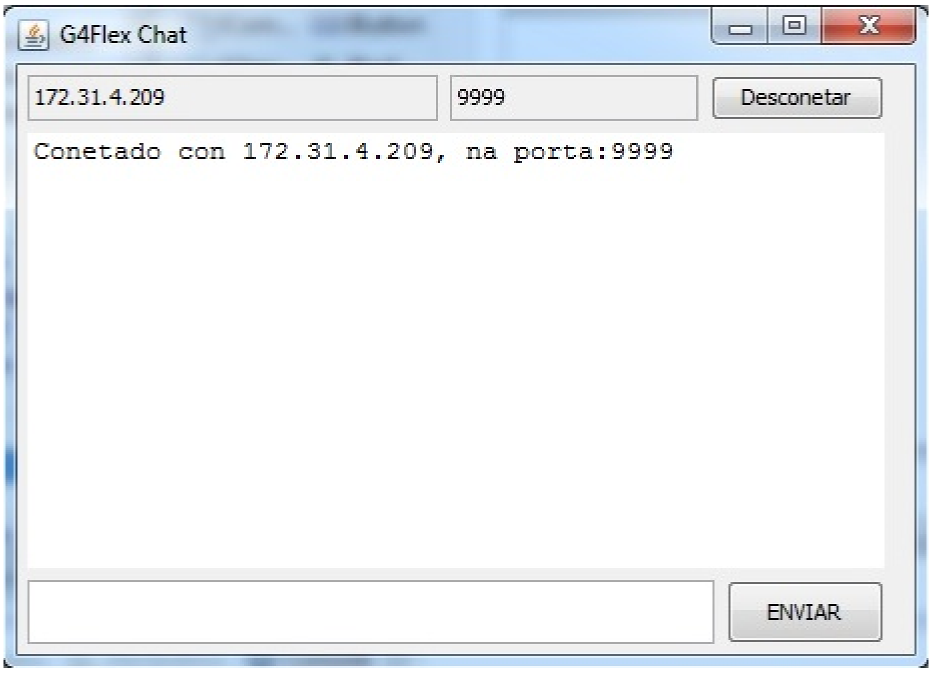
For example, the user *172.31.4.209* wants to connect to the user *172.31.4.205*, through the *9999* port. Once he writes the address and the port, and clicks *Conetar* button, the client-thread of the first user starts running, creating a socket with the server of the second user. At this time, the Chat Window of both users opens.

Notice that when the Chat Window of the receiving user opens, a notification sound also exist, to let he know that a message is arriving.



```
Conetado com 172.31.4.205, na porta:9999
```

Figure 25: Chat Window of first user (172.31.4.209)

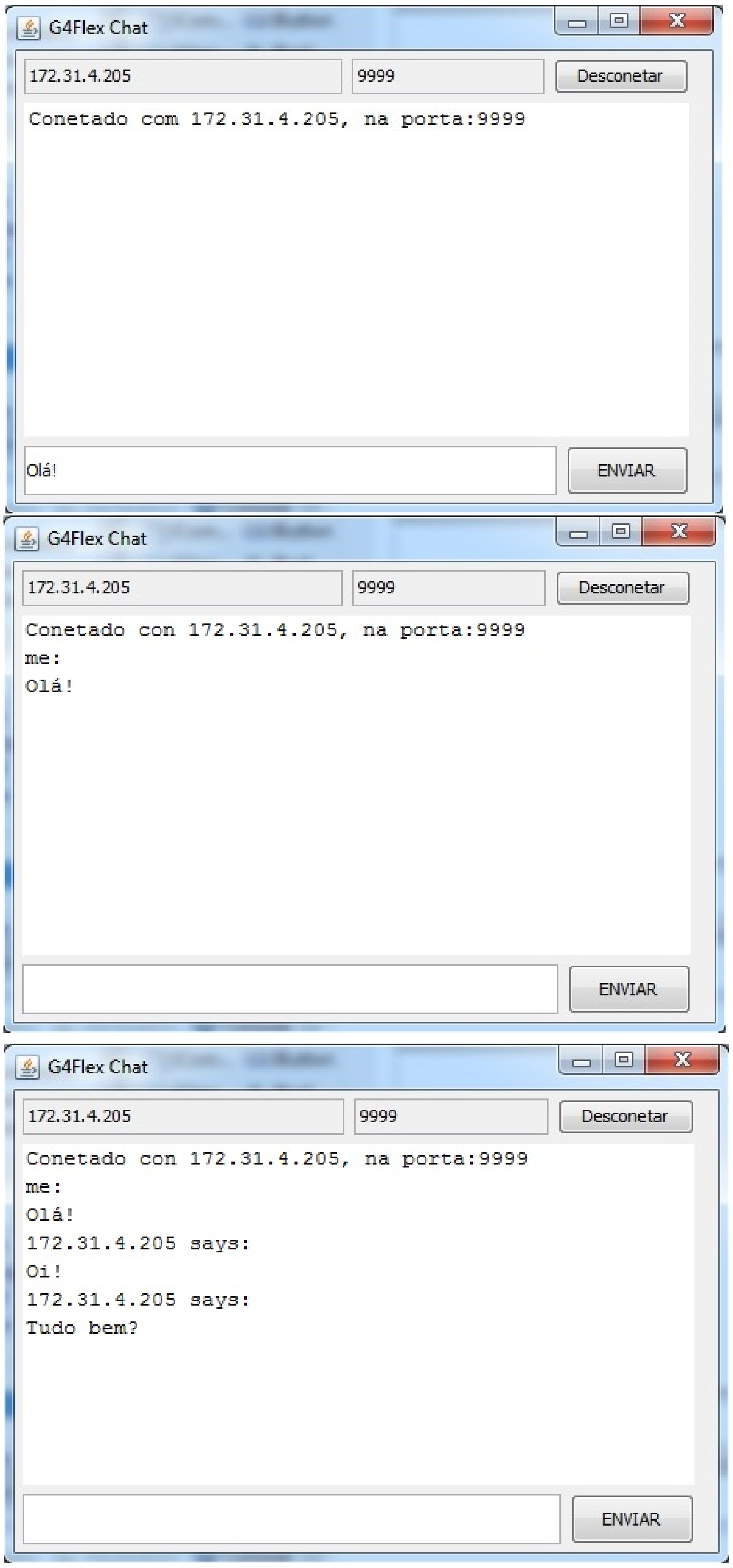


```
Conetado con 172.31.4.209, na porta:9999
```

Figure 24: Chat Window of second user (172.31.4.205)

Once the connection is done and sockets are created, the chat is done. As the client and server are running in different threads, both users can talk simultaneously.

In the pictures below, we can see an example of a message exchange.



```
172.31.4.205    9999
Conetado com 172.31.4.205, na porta:9999

Olá!
```

```
172.31.4.205    9999
Conetado con 172.31.4.205, na porta:9999
me:
Olá!
```

```
172.31.4.205    9999
Conetado con 172.31.4.205, na porta:9999
me:
Olá!
172.31.4.205 says:
Oi!
172.31.4.205 says:
Tudo bem?
```

Figure 26: Example of a message exchange

CONCLUSIONS

This project has been concluded with satisfaction. It has succeeded in making the application G4Flex Phone send messages between two users.
Even so, we found ourselves with various problems they did modify the initial idea. The fact of working on a softphone previously done and do not create it from zero, provocated some complications at the beginning. The little clarity of the program already done and the impossibility of contacting its author, meant to spend a lot of time to introduce myself in the project and be able to start to work on it.

The main idea, as we can see in the work [26], was to improve the already existing Java application by attaching the message transmission between users using the SIP protocol. But after studying it calmly and make several attempts, we did not find a solution. The problem was that the SIP protocol, together with the enrebassament of the program already created, was chaotic and confusing to attach the SIP message function. That is why we were forced to look for a possible alternative.

So, after assessing different options, we believed that the best option was to send messages using the UDP protocol to be fast and efficient, and, above all, to be fully compatible with Asterisk. In addition, being simple and basic, we could create our own code and join it to the previous program, without affecting its efficiency or functionality.

The results obtained demonstrate that the application finally works as we had expected.

Personally, working on this project has been very interesting because I have been able to delve into the field that most interested me during the career: the telematics. The fact of having to search for information and immerse myself in the IP technology and its applications, has opened a new world with a lot of options ahead.

As a final conclusion, this work has contributed to the implementation of the Brazilian company G4Flex's own softphone which, despite not having been made with the intention of being commercialized, it strengthens the brand of the company.

BIBLIOGRAPHY

1. LITWA, Mark; **The evolution of the telephone.** July, 2009. Available in: <http://www.slideshare.net/mjlitwa/the-evolution-of-the-telephone>.

2. History of Telecommunication. Buzzinbees, 2013. Available in: <http://www.buzzinbees.com/about-us/tutorial>.

3. History of Telecommunications. *Wikipedia, 2014.* Available in: <http://en.wikipedia.org/wiki/History_of_telecommunication>.

4. Voice over IP: A reference guide to VoIP. 2013-2014. Avaiable in: <http://www.voip-info.org/>.

5. RAMÍREZ, Tamara et DÍAZ, Jaime; **Aplicaciones sobre una red de telefonía IP.** UTFSM, Chile. Available in: <http://profesores.elo.utfsm.cl/~tarredondo/info/networks/Presentacion_voip.pdf>.

6. UNUTH, Naideem; **Voice over IP Cons:** VoIP Problems and Pitfalls. Philadelphia, 2011. *Available in:* <http://voip.about.com/od/voipbasics/a/voipproblems.htm>.

7. NETO, Renato de Freitas Bulcão; PIMENTEL, Maria da Graça Campos. **Multimídia sobre IP:** Protocolos e Padrões. Departamento de Computação e Estatística – USP, Brasil, 2001.

8. ITU-T Recommendation H.323: Packet-based multimedia communications systems. ITU-T, 2009. Avaiable in: <http://www.itu.int/rec/T-REC-H.323-200912-I/en>.

9. H.323 Protocols Suite. Protocols. Avaiable in: <http://www.protocols.com/pbook/h323.htm>.

10. Voice over IP: IAX. August 2012. Avaiable in:<http://www.voip-info.org/wiki/view/IAX>.

11. SPENCER, M. et al. **IAX: Inter-Asterisk eXchange Version 2**. IETF, 2006. *IETF draft-guy-iax-01.*

12. ROSENBERG, J. et al. **RFC 3261 - SIP: Session Initiation Protocol.** IETF, 2002. Available in: <http://www.ietf.org/rfc/rfc3261.txt.pdf>.

13. Voice over IP: Session initiation protocol. May, 2014. Available in: <http://www.voip-info.org/wiki/view/SIP>.

14. Session Initiation Protocol. Wikipedia. Available in: <http://en.wikipedia.org/wiki/Session_Initiation_Protocol>.

15. Handley, M. et al. **RFC 2327 - SDP: Session Description Protocol.** IETF, 1998. Available in: <http://www.ietf.org/rfc/rfc2327.txt>.

16. Session Description Protocol. Wikipedia. Available in: <http://en.wikipedia.org/wiki/Session_Description_Protocol>.

17. SCHULZRINNE, H. et al. **RFC 1889 - RTP: Real-Time Transport Protocol.** IETF, 1996. Available in: <http://www.ietf.org/rfc/rfc1889.txt>.

18. Real-Time Transport Protocol. Wikipedia. Available in: <http://en.wikipedia.org/wiki/Real-time_Transport_Protocol>.

20. POSTEL, J. **RFC 793 - Transmission Control Protocol.** Information Sciences Institute University of Southern California. IETF, September 1981. Available in: <http://www.ietf.org/rfc/rfc793.txt>.

21. Transmission Control Protocol. Wikipedia. Available in: <http://en.wikipedia.org/wiki/Transmission_Control_Protocol>.

22. POSTEL, J. **RFC 768 – User Datagram Protocol.** IETF, August 1980. Available in: <https://www.ietf.org/rfc/rfc768.txt>.

23. User Datagram Protocol. Wikipedia. Available in: <http://en.wikipedia.org/wiki/User_Datagram_Protocol>.

24. ASTERISK. *Available in:* <http://www.asterisk.org/>.

25. Voice over IP: Asterisk. April, 2014. Available in: <http://www.voip-info.org/wiki/view/Asterisk>.

26. PAULA, Renan Pinto de. Desenvolvimento de aplicação multimídia com utilização do protocolo SIP. Trabalho Conclusão do Curso. Departamento de Ciência da Computação, Unifor, Fortaleza, 2012.

27. The Java Tutorials. Oracle Corporation. Available in: <http://docs.oracle.com/javase/tutorial/>.

28. Java Development: Java Development User Guide. Eclipse Foundation, 2004. Available in: <http://help.eclipse.org/kepler/index.jsp?nav=%2F1>.

29. PROULX, Emmanuel. **An Introduction to the JAIN SIP API.** Oracle Corporation, 2007. Available in: <http://www.oracle.com/technetwork/articles/entarch/introduction-jain-sip-090386.html>.

30. MARTINEAU, Johann; Peers Java SIP Softphone: Simple telephony application. Available in: <http://peers.sourceforge.net/>.

31. Eclipse: Window Builder User Guide. Eclipse Foundation, 2004. Available in: <http://help.eclipse.org/kepler/index.jsp>.

Fortaleza - Brasil
May 2014