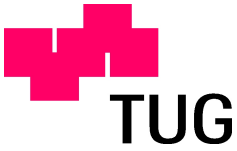
Diploma Thesis

Aspects of the Integration of New Generation Communication Services

JOHANN MADLBERGER

GRAZ, SEPTEMBER 2002

SUPERVISOR: O.UNIV.PROF. DR.PHIL. DR.H.C. HERMANN MAURER
THESIS ADVISOR: UNIV.ASS. DI. HARALD KROTTMAIER



INSTITUTE FOR INFORMATION PROCESSING AND
COMPUTER SUPPORTED NEW MEDIA
GRAZ UNIVERSITY OF TECHNOLOGY

Diplomarbeit aus Telematik

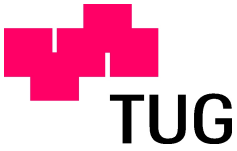
Aspekte der Integration von Kommunikationsdiensten der neuen Generation

JOHANN MADLBERGER

GRAZ, SEPTEMBER 2002

BEGUTACHTER: O.UNIV.PROF. DR.PHIL. DR.H.C. HERMANN MAURER

BETREUER: UNIV.ASS. DI. HARALD KROTTMAIER



INSTITUT FÜR INFORMATIONSVERARBEITUNG UND
COMPUTERGESTÜTZTE NEUE MEDIEN
TECHNISCHE UNIVERSITÄT GRAZ

SUBMITTED TO THE INSTITUTE OF INFORMATION PROCESSING AND COMPUTER SUPPORTED NEW MEDIA, GRAZ UNIVERSITY OF TECHNOLOGY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DIPLOM INGENIEUR
(MASTER OF ENGINEERING SCIENCES)

IN THE FIELD OF COMMUNICATION ENGINEERING AND COMPUTER SCIENCES.

GRAZ, SEPTEMBER 2002

This Thesis is a Cooperation of the Graz University of Technology and Infonova Information Technology, Head Office Seering 6, A-8141 Unterpremstätten, Austria, Department for IP Telephony, Planning & Carrier Aspects Group. The supervision and assistance of this thesis at Infonova was done by Gerhard Zweimüller.

I hereby certify that the work reported in this thesis is my own and that work performed by others is appropriately cited.

Signature of the author:

Ich versichere hiermit wahrheitsgemäss, die Arbeit bis auf die dem Aufgabensteller bereits bekannte Hilfe selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten Anderer unverändert oder mit Abänderungen entnommen wurde.

Abstract

This diploma thesis deals with the different kinds of communication on the Internet. Text-based communication services like email, news, Internet Relay Chat (IRC) or instant messaging are explained as well as real-time services. Examples for real-time services are the Internet Protocol (IP) telephony or video over IP. Unified messaging systems integrate different communication services. They enable users to use various devices for accessing their received messages. The thesis focuses on the architecture and the protocols of the communication services. In particular signaling protocols like H.323 or the Session Initiation Protocol (SIP), and transport protocols such as the Real-Time Transport Protocol (RTP) are explained in detail. Also SIMPLE, an instant messaging protocol which is still in the stage of development, is described. The practical part of the thesis was done in co-operation with Infonova GmbH. The task was to implement a client for an instant messaging system. The user handbook, as well as the description of the architecture are also included in this thesis.

Zusammenfassung

Die vorliegende Diplomarbeit beschäftigt sich mit den verschiedenen Arten der Kommunikation, welche im Internet möglich sind. Es werden sowohl text-basierte Kommunikationsdienste wie zum Beispiel Email, News, Internet Relay Chat (IRC) oder Instant Messaging als auch Echtzeitdienste wie Internet Protocol (IP) Telephonie oder Video over IP behandelt. Unified Messaging Systeme integrieren verschiedene Kommunikationsarten und ermöglichen von verschiedenen Geräten aus den Zugriff auf die erhaltenen Mitteilungen. Im Mittelpunkt der Arbeit stehen die Architektur und Protokolle der jeweiligen Kommunikationsformen. Besonders Signalisierungsprotokolle wie H.323 oder das Session Initiation Protocol (SIP), sowie Transportprotokolle wie das Real-Time Transport Protocol (RTP) werden ausführlich erläutert. Auch das gerade in der Entwicklungsphase stehende SIMPLE Protokoll für Instant Messaging Systeme wird behandelt. Im praktischen Teil der Arbeit wurde in Zusammenarbeit mit der Firma Infonova ein Client für ein Instant Messaging System entwickelt. Das Benutzerhandbuch sowie eine Beschreibung der Architektur dieser Anwendung sind ebenfalls in dieser Arbeit enthalten.

Contents

List of Figures

List of Tables

Preface

In the late 1960s, the Advanced Research Projects Agency (ARPA) of the U.S. Department of Defense, U.S. universities and research organizations developed the first packet-switched network, called Advanced Research Projects Agency Network (ARPANET). The purpose of ARPANET was the correct transport of data packets. The main goal was reliability. That is, when a route between two hosts failed, the packets should automatically be transported via an alternative route. At that point, only a few scientists thought about Quality of Service (QoS) or the transmission of multimedia data.

The commercialization of the Internet and the resulting increase of users changed the communication behavior of those connected to this global network. Newsgroups, emails and Internet Relay Chats (IRCs) became more and more important and started to replace the traditional forms of communication like the telephone or the postal service. With the increase of available bandwidth, again, new communication applications were developed.

This thesis describes the different types of communication services. It also analyses, whether or not an integration of such services is possible and necessary. The main focus of this work is on Internet Protocol (IP) telephony and instant messaging, whereby the latter topic was also needed for the practical part of the thesis.

Chapter 1 is an introduction to communication services on the Internet. It tries to classify the different communication methods. It then outlines the architecture and the most important protocols of email, news and IRC.

The basics of IP telephony are presented in chapter 2. First, the Public Switched Telephone Network (PSTN) is described, followed by the advantages and problems of voice transmissions over the Internet. This includes an explanation of some audio codecs. Finally, Fax over IP (FoIP), video over IP and data conferencing is outlined.

Chapter 3 and chapter 4 explain two signaling protocols of IP telephony, namely the H.323 protocol family and the Session Initiation Protocol (SIP), respectively. These protocols are necessary to initiate and terminate telephone calls.

For the transmission of real-time data, the Real-Time Transport Protocol (RTP) as well as the Real-Time Transport Control Protocol (RTCP) were developed. Chapter 5 describes the requirements of real-time transmissions and the basics of RTP and RTCP.

Chapter 6 introduces instant messaging. This method of communicating was made popular by Mirabilis with their program ICQ. There are many instant messaging protocols available today. One of them is the SIMPLE protocol, which was developed by the Internet Engineering Task Force (IETF) and is based on SIP.

A possible integration of communication services is described in chapter 7. A Unified Messaging System (UMS) enables users to choose their preferred device or application for sending and retrieving messages. For example, a UMS makes it possible to retrieve emails with a normal mobile phone. This chapter contains a presentation of the benefits of unified messaging as well as a description of a sample UMS architecture.

The practical part of this thesis was to design and implement an instant messaging client for Infonova GmbH. The open source server 'Jabber' was chosen as instant messaging server. Chapter 8 specifies the requirements of the client as well as the architecture of the Jabber server. Then, the Jabber protocol is explained. Finally, the architecture of the client and its handling is described.

The last chapter sums up the results of this thesis and gives an outlook to the future of communication methods. Appendix A introduces the standardization organizations that specified the standards used in this thesis.

Chapter 1

Introduction

Computer networks and in particular the Internet strongly influenced the communication behavior of the people worldwide. The technical evolution in the last decades made it possible to communicate with other users in a way nobody ever thought before. While we probably use email, newsgroups and chats every day or at least once in a while other services are still in the development stage. For example even though instant messaging is used by many people, most of the existing instant messaging systems use different protocols. Therefore the interoperability between these systems is not possible. On the other hand there already exist standards for voice transmission over the Internet, but because IP offers no guaranteed QoS, the quality of voice transmission is often bad.

Trying to classify the different kinds of communication offers several possibilities. For example one can differ if a message is sent to one person or to a group of people. Another possibility would be the classification into synchronous and asynchronous communication. Table 1.1 shows the assignment of services into these categories. *Talk* is a UNIX tool that is similar to chat except that it is limited to one-to-one communication.

	one-to-one	**one-to-many**
asynchronous	email	news, mailing lists
synchronous	talk, voice over IP, video over IP	IRC, audio and video conferencing

Table 1.1: Classification of communication services on the Internet

The following sections outline the most widespread communication services, namely email, news and IRC. Further information of these topics can be found in [Tanenbaum, 1996] and [Feit, 1998].

1.1 Email

Electronic mail is the most popular communication service on the Internet. It is a simple, fast, reliable and cheap way to send messages to other people. An email system consists of two parts: a user agent (client application) and a message transfer agent (email server). The client helps the user to compose and send the message. It can manage an address book and include a spellchecker. The message transfer agent is responsible for the transfer and the storage of the messages.

1.1.1 Message Format

The format of an email message is specified by the Internet Engineering Task Force (IETF) in Request For Comments (RFC) 2822 [Resnick, 2001]. Such a message includes an envelope (specified in [Klensin, 2001]), some headers, a blank line and the message text. When the message is sent, the message transfer agent generates the envelope from some of the headers. A header field has following form:

Field-name: value

There are many headers, but for a normal email message, many of them are obsolete. The field *To* defines the DNS-address of the receiver. In the field *Cc* (Carbon Copy) the user can declare more recipients of the message and in *Bcc* (Blind Carbon Copy) receivers, that are invisible to other recipients, can be specified. The fields *From* and *Sender* declare who wrote and sent the message, respectively. Normally, these fields are identical, but an undeliverable message is always sent back to the person specified in the *Sender* field while the displayed sender is that in the from-field (important when for example a secretary sends a message for her boss). In the field *Reply-To*, an address other than that of the sender can be declared. The response to this message is then sent to the *Reply-to* address. The field *Subject* should describe the content of the message.

After the header, a blank line and the message follows. If this message should not just contain flat US-ASCII-text, an extension to RFC 2822, which is explained in the next section, must be used.

Multipurpose Internet Mail Extensions

Multipurpose Internet Mail Extensions (MIME) is specified in RFC 2045 [Freed and Borenstein, 1996a], RFC 2046 [Freed and Borenstein, 1996b], RFC 2047 [Moor, 1996], RFC 2048 [Freed et al., 1996] and RFC 2049 [Freed and Borenstein, 1996c]. It extends RFC 2822 by defining additional headers (see Table 1.2).

That makes it possible, that for example textual messages in character sets other than US-ASCII, non-textual message bodies (e.g. pictures, audio or video), multi-part message bodies or textual header information in character sets other than US-ASCII can be used.

Header	Description
MIME-Version	Identifies the MIME-version
Content-Description	Describes the content of the message
Content-Id	Unique identifier
Content-Transfer-Encoding	Specifies the encoding of the message-content
Content-Type	Describes the kind of message

Table 1.2: MIME header fields

The first header states which version of MIME is used. If no MIME-version header is specified, then it is a standard text-message without MIME. It is often very useful to know, what an image in a message represents. So it is easy to decide if it is worth to open the content. This description, which is optional, is specified in the *Content-Description* header. The *Content-Id* header contains a unique number for the MIME-content.

The Simple Mail Transfer Protocol (SMTP) (described in the next section) is using 7bit characters and limits the length of a line to 1000 characters. Because many media types use 8bit characters, a standard mechanism for encoding such data is necessary. The *Content-Transfer-Encoding* header specifies this mechanism.

The header *Content-Type* can be used to describe the media type and subtype of data in the body of a message. Type and subtype must be separated by a slash. RFC 2046 [Freed and Borenstein, 1996b] defines an initial set of seven top-level media types (text, image, audio, video, application, message, multipart). The splitting into type and subtype (e.g. image/jpeg) has the advantage, that a user agent can recognize the type (image), even if it does not know the specific subtype. This can be used to decide, whether or not the user agent should display the raw data of an unrecognized subtype.

1.1.2 Message Transfer

Figure 1.1 shows the way of an email from the originator to the destination host. When a user composes and sends a message, it is usually queued to a Message Transfer Agent (MTA). This MTA transmits the email to the destination host using the SMTP or the Simple Mail Transfer Protocol Service Extensions (ESMTP). It is possible, that between originator and destination a relay host is located, where the message is stored until it can be forwarded at a convenient time. When the message reaches the destination host it is moved to the mailbox of the receiver. There it can be picked up by the user agent of the recipient.

A relay host is necessary for protocol translations when originator and destination host use different protocols. Also if a company decides to send and receive all email-traffic over one server (for security reasons) a relay host is needed.

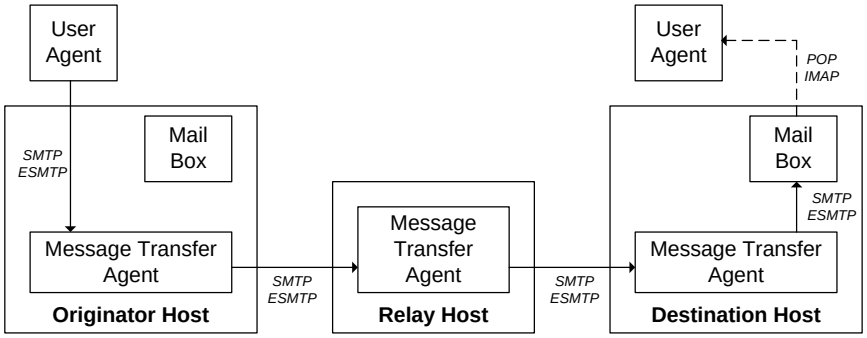


Figure 1.1: Email message transfer

Simple Mail Transfer Protocol

SMTP (specified in [Klensin, 2001]) is a protocol from the IETF, that defines a way to transfer email from one host to another, whereas the sender acts as a client and the receiver as a server. First, the sender opens a Transmission Control Protocol (TCP) connection to the receiver, which is listening at port 25. Note that SMTP also supports other protocols than TCP. Then the server identifies itself. Next, the client sends 'HELO', followed by the senders host name. Then the client identifies the message originator, one ore more recipients, and finally sends the mail data followed by a period, *Carriage Return (CR)* and *Line Feed (LF)*.

[Klensin et al., 1993] describes ESMTP as a way to extend SMTP. When a client wants to use ESMTP, it sends at the beginning of a section 'EHLO' instead of 'HELO'. If the server does not except this command, then the client must communicate via SMTP.

When an email has arrived at the mailbox of the destination host, the user agent can pick it up by using either the Post Office Protocol (POP) or the Interactive Mail Access Protocol (IMAP).

Post Office Protocol

The current version of POP is POP3 [Rose, 1991]. It is a very simple protocol, that handles the moving of the emails from the server to the client. Normally, the server listens at port 110 for incoming client requests.

When the TCP-connection has been established, the server sends a greeting. Then the client can send commands to the POP3 server. A command consists of a keyword possible followed by an argument. The server responses by sending a success operator and maybe additional information. Every command and response is terminated by CR and LF. If the response contains more than one line, a final line consisting of a '.' and a CR/LF, indicates the end.

A POP3 session begins with the authentication of the user. The client sends the *USER* command followed by the username. Then the *PASS* command with

the secret password is sent. Note that this unencrypted password is a big security problem. When the user is authenticated he can for example retrieve or delete emails. To quit a session, the client simply sends a *QUIT* command. After the response from the server, the TCP connection can be closed. Figure 1.2 shows an example of a POP3 session.

```
S: <wait for connection on TCP port 110>
...
C: <open connection>
S: +OK dewey POP3 server ready (Comments to: PostMaster@UDEL.EDU)
C: USER mrose
S: +OK mrose is a real hoopy frood
C: PASS secret
S: +OK mrose's maildrop has 2 messages (320 octets)
C: STAT
S: +OK 2 320
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
C: RETR 1
S: +OK 120 octets
S: <the POP3 server sends message 1>
S: .
C: DELE 1
S: +OK message 1 deleted
C: RETR 2
S: +OK 200 octets
S: <the POP3 server sends message 2>
S: .
C: DELE 2
S: +OK message 2 deleted
C: QUIT
```

Figure 1.2: Example of a POP3 session (from [Rose, 1991])

Interactive Mail Access Protocol

A more sophisticated protocol to maintain emails is IMAP [Crispin, 1996]. An IMAP server does not copy the emails to the client like POP3. This has the advantage that users with more than one computer can read their emails also from other locations. Another benefit is that IMAP servers are able to manage multiple mailboxes.

The IMAP protocol needs a reliable data stream like TCP provides it. When the client has established a connection to the server, which usually listens at port 143, the server responds with an initial greeting. Then the client/server interaction can proceed. A client command is always prefixed with an identifier (e.g. A0001, A0002, etc.). Normally a command is limited to one line, but if this is not the case, the server sends a command continuation request response when it is ready for the remainder. This response is prefixed with the token "*". Each command from the client is parsed by the IMAP server. Then the server transmits the data and a command completion result response, which indicates the success or the failure of the operation. It uses the same tag like the client command that began the operation.

The IMAP server stores the emails in a mailbox. The user can create new mailboxes, rename or delete them. Every message has a unique identifier. A sequence number states the relative position in the mailbox. There are some flags to indicate the status of the message. For example *Seen* marks the message as read or *Answered* states, that the message already has been answered. Other flags are *Deleted*, *Recent* (message is new), *Draft* or *Flagged* (for an urgent message).

Figure 1.3 shows an example of a client/server interaction. The *SELECT* command selects a mailbox so that messages in the mailbox can be accessed.

```
C: A142 SELECT INBOX
S: * 172 EXISTS
S: * 1 RECENT
S: * OK [UNSEEN 12] Message 12 is first unseen
S: * OK [UIDVALIDITY 3857529045] UIDs valid
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS (\Deleted \Seen \*)] Limited
S: A142 OK [READ-WRITE] SELECT completed 6.3.2. EXAMINE Command
```

Figure 1.3: Example for an IMAP4 client/server interaction (from [Crispin, 1996])

More information on IMAP can be found in the specification or at the website of *The IMAP Connection*[1].

1.2 News

News is another communication service on the Internet that is used by many people. Newsgroups make it possible, that users who do not know each other can discuss about topics they are interested in. There are more that 100000 newsgroups. The system of newsgroups is also known as USENET. Since news is not just a kind of communication service but also a huge resource for information, searching in a specialized newsgroup often provides better results than searching in the World Wide Web (WWW).

[1] www.imap.org

1.2.1 Newsgroup Organization

There are thousands of public and private newsgroups. They are organized hierarchical. Table 1.3 shows the official top level hierarchies, but there are also other ones which mostly are grouped by countries or languages. An example of a newsgroup is *rec.sport.soccer*.

Name	Description
alt	Any conceivable topic
biz	Business products, services and more
comp	Information about computer hardware or software
humanities	Literature, art, philosophy ...
misc	Things that do not fit into another group
news	Discussions about USENET
rec	Sport, hobbies, games ...
sci	Science and Technology
soc	Social issues, culture ...
talk	Current issues and topics

Table 1.3: Top level hierarchies for newsgroups

If somebody wants to create a new public newsgroup he posts a message in *news.groups*. This message should contain the name and the purpose of the newsgroup. After a discussion an email voting is made. If the supporter of the new group has more than twice of the votes than the opponents and there are at least one hundred yes-votes more than no-votes, the group is accepted. Since in the *alt* hierarchy often very controversial issues are discussed, the creation of a new group is not as formal as in the other hierarchies.

1.2.2 Article Format

A news article has the same format as an email message (see section 1.1) except for some additional headers. These headers are listed in Table 1.4. Like email messages, also news articles can use MIME to include multimedia contents.

1.2.3 Distribution of Articles

If newsgroups do not have much traffic they can be realized as mailing lists, where all articles are sent to every subscriber. Of course this is not possible for large groups with thousands of subscribers. A news server creates a directory structure for all subscribed newsgroups and stores incoming articles in the appropriate directory on the local hard disk (e.g. an article in *rec.sport.soccer* could be stored in *news/rec/sport/soccer*). To receive articles the server must have a *newsfeed* from a server in the USENET. When this server has new articles available it sends them automatically to all servers he feeds. It is also possible that a server asks for news

Header	Description
Path	A list of all hosts, the article has passed from the source to the destination. Hosts are separated by an exclamation mark. Every host that transfers the article adds its host name to the list
Newsgroups	Specifies the groups this article is related to.
Followup-To	Indicates the groups where replies should be posted.
Distribution	States the distribution of the article. This can be a specific country, a network or the whole world.
NNTP-Posting-host	Specifies the news server that has sent the article.
Reference	This identifier specifies a previous article this one refers to. If the article starts a new thread this field is empty.
Organization	Describes the organization where the author of the message works.
Lines	The length of the content in lines without header fields.

Table 1.4: Additional headers for news articles

but this is quite unusual. A news reader or news agent gets the articles from the suitable directory on the server.

Network News Transfer Protocol (NNTP)

NNTP is used by news agents to communicate with news servers as well as by servers to communicate with other servers. The protocol is specified in [Kantor and Lapsley, 1986]. An NNTP server usually listens at TCP port 119. When a client wants to retrieve articles, it establishes a connection to this port. The protocol runs over a telnet network virtual terminal session like SMTP does. The client sends a command and the server responses with a status number and data.

Figure 1.4 shows a session of a user that retrieves an article from a newsgroup. To display the newsgroups available on this server the client sends either the *list* or the *newgroups* command. The difference is that *newgroups* just requests new groups that have been created since a given date and time, and *list* gets all available groups. As result the server sends a list of the newsgroups followed by the last known article currently in that newsgroup, the number of the first article currently in the newsgroup, and either 'y' or 'n', indicating whether posting to this newsgroup is allowed ('y') or prohibited ('n'). Then the client can select a group by using the *group* command. The server responds with the estimated number of articles in the group, the first and the last article and the name of the group. Finally the client gets the desired article by using the *article* command with the article number as argument.

```
C: newgroups 990101 000000
S: 231 New newsgroups follow.
S: tu-graz.fm.announce 39975 38972 m
...
S: tu-graz.lv.duep 372 1 y
...
S: tu-graz.lv.gen 0 1 y
S: .
C: group tu-graz.lv.duep
S: 211 353 1 372 tu-graz.lv.duepa
C: article 370
S: 220 370 <ib3d3zg7sq.fsf@fiicmpc54.tu-graz.ac.at> article
S: Path: news.tu-graz.ac.at!not-for-mail
S: From: Harald Krottmaier <hkrott@iicm.edu>
S: Newsgroups: tu-graz.lv.duep
S: Subject: fyi: modifizierte LaTeX-Vorlage am ftp-server...
S: Date: 31 Oct 2001 16:15:17 +0100
S: Organization: Technische Universitaet Graz, Austria
S: Lines: 14
S: Message-ID: <ib3d3zg7sq.fsf@fiicmpc54.tu-graz.ac.at>
S: NNTP-Posting-Host: fiicmgk01.tu-graz.ac.at
S: Mime-Version: 1.0
S: Content-Type: text/plain; charset=us-ascii
...
S: .
C; quit
S: 205 .
S: Connection closed by foreign host.
```

Figure 1.4: Retrieving an article using NNTP

1.3 Internet Relay Chat

In the year 1988 the Finnish student Jarkko Oikarinen from the University of Oulu implemented the first IRC client and server. IRC became very popular and was distributed all over the world. Several IRC networks like IRCnet[2] or EFnet[3] arose.

IRC was a success because it enables people to communicate in an easy way. To allow a reasonable communication between users, channels were introduced. They subdivide the chat into different topics. Users in a chat do not use their real name but nicknames. It is not allowed that two users have the same nickname at the same time.

[2] http://www.ircnet.org/
[3] http://www.efnet.org/

A very popular alternative to IRC are web-based chat systems. Because they are not standardized but in principle very similar to IRC, web chats are not explained.

1.3.1 Architecture

An IRC network is based on a distributed client/server architecture [Kalt, 2000a]. It consists of at least one server but in the normal case there are several ones that are connected. These servers form the backbone of IRC. Clients can establish a connection to one of the servers. A big problem of such an IRC network is that every server needs a copy of the global state information. This limits the maximum size a network can reach.

The only allowed network configuration in IRC is that of a spanning tree. That means that there is exactly one possible way between two servers. IRC allows several kinds of communication between clients. The simplest one is a normal one-to-one communication. Figure 1.5 shows an example of an IRC network. When Client 1 sends a message to Client 3 it is transferred via Server A and Server B to the receiver. No other server sees that message. Another possible communication is that of one client to all clients in a channel. In that case the message is sent only to the servers that support a client on the given channel. Similar to this kind of communication is the sending of messages to a host/server mask and to a list of receivers. Also messages to all users in the IRC network are feasible.

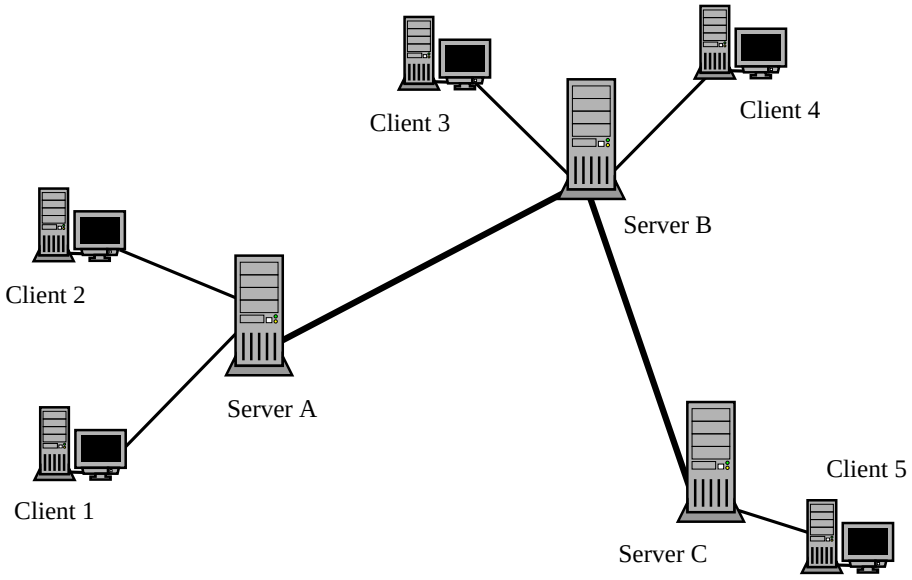


Figure 1.5: Internet Relay Chat

IRC has a very sophisticated channel management (specified in [Kalt, 2000b]). A channel can be local to the server where it was created. A local channel is

prefixed with '&'. A channel can also be known by servers specified in a channel mask or by all servers. The properties of a channel are defined by so-called channel modes. Privileged members like the channel creator or the channel operators can be defined, too.

1.3.2 Protocols

IRC has two kinds of protocols: a server protocol for the communication between the IRC-servers and a client protocol for the client/server communication.

Client Protocol

The client protocol uses an asynchronous communication between client and server. This means that the server may or may not sent a reply to a request of the client and vice versa. An IRC message may consist of a prefix (optional), a command, and the command parameters (limited to 15 parameters). Prefix, command, and all parameters are separated by one blank (ASCII-code 0x20). If the first character of a message is a colon, the message includes a prefix. This prefix indicates the true origin of the message. A client should not use a prefix. If it uses one, the only valid prefix is its registered nickname. To register a connection, the client first sends a *password* message (optional; usage: *PASS <password>*) followed by a *nickname* message (*NICK <nickname>*). The server replies with a *service* message. Finally, the client sends a *user* command (*USER <user> <mode> <unused> <realname>*).

Other commands for joining or leaving a channel, inviting other users to a channel or sending messages are also available. Details of the client protocol can be found in [Kalt, 2000c].

Server Protocol

Like the client protocol, the server protocol uses an asynchronous communication. A server provides three basic services:

- client locating
- message relaying
- channel hosting and management

Each server has to maintain a global state database, which is, in theory, identical on all servers. A server must know the identifier of every user in the network and to which server it is registered. The server protocol is defined in [Kalt, 2000d].

1.4 Summary

There are two possibilities in classifying communication services. They can be either synchronous and asynchronous, and one can also differ, if a message is sent to one person or to a group of people.

To date, email is the most popular communication service. It consists of two parts: a user agent and a message transfer agent. If an email contains non-US-ASCII-text, MIME must be used. The IETF has specified several protocols for the transfer of emails: SMTP and its extension ESMTP were designed for sending the messages to the mailbox at the destination host, while POP and IMAP are used to download them from the mailbox.

News is another popular communication service. Newsgroups are organized with an official top level hierarchy. There are more that 100000 newsgroups. For the distribution of articles, NNTP was specified.

The last kind of communication introduced in this chapter is IRC. An IRC network is based on a distributed client/server architecture. Two protocols for IRC exist: a client protocol for the communication between client and server, and a server protocol for the server-to-server communication.

Chapter 2

IP Telephony

The communication services presented in the introduction chapter are text-based and need no guaranteed Quality of Service (QoS). That is the reason why they are perfect for use in the packet-switched Internet. In contrast to these services the telephony system was invented more than 100 years ago and is based on a circuit-switched network. This Public Switched Telephone Network (PSTN) is very reliable and telecommunication companies have invested much money to build it up and maintain it. Therefore they need good reasons to migrate the speech-data to a packet-switched network.

This chapter first describes the basics of the Public Switched Telephone Network and then explains how IP telephony works. Also the advantages and problems of IP telephony are discussed. After a brief introduction to audio codecs also fax over IP, video over IP and data conferencing are described.

2.1 The Public Switched Telephone Network

Although this chapter deals with IP telephony, it is especially for software engineers very important to understand the basics of PSTN. This section briefly explains the architecture and the most important components.

2.1.1 Architecture

Figure 2.1 shows the structure of PSTN. A call from a telephone is transferred over an access line (local loop) to a Central Office. There, a Class 5 voice switch digitizes the call to a 64 kbps Pulse Code Modulation (PCM) voice stream (according to ITU G.711) [ITU, 1999a].

Using Time Division Multiplexing (TDM), this stream is multiplexed onto trunk lines, which connect to Class 4 voice switches or to Private Branch Exchanges (PBXs). A PBX is used for example by companies or universities to bundle their telephone lines. Like Class 5 switches, they handle the routing of the calls to the telephones, but often they have additional features.

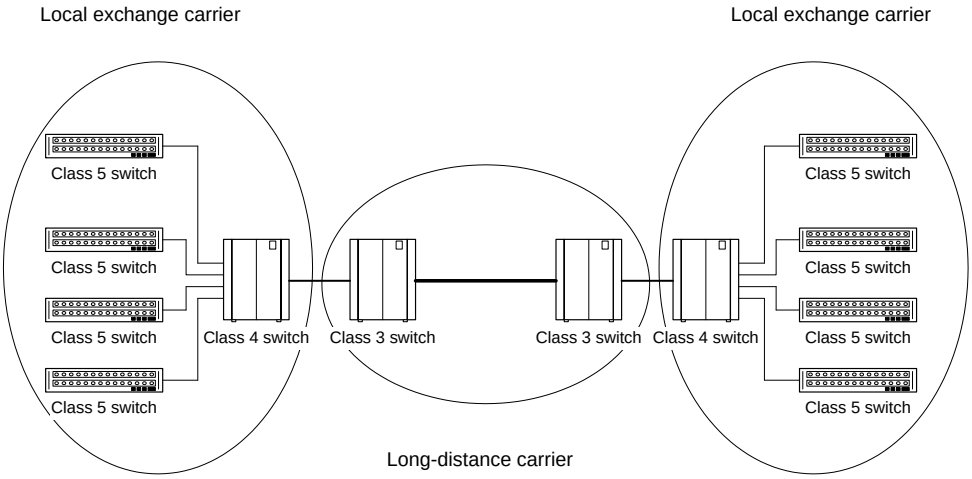


Figure 2.1: Public Switched Telephone Network

2.1.2 Signaling

Before a person can talk to another, a connection must be established. This functionality is realized by signaling protocols. A distinction is drawn between user-to-network signaling and network-to-network signaling.

User-to-Network Signaling

The most frequented method for analog user-to-network signaling is Dual Tone Multi Frequency (DTMF). Every numeral is assigned to a unique frequency. This tone is sent over the voice channel to the switch (In-Band signaling).

Integrated Services Digital Network (ISDN) uses Out-of-Band signaling. In this case, a separate *Data channel* (D-channel) is used for the signaling information. This channel has a transfer rate of 16 kbps. The voice is transmitted over a so-called *Bearer channel* (B-channel) with 64 kbps. An ISDN Basic Rate Interface (BRI) has two B-channels and one D-channel. The more powerful Primary Rate Interface (PRI) uses 30 B-channels and one D-channel (in Europe).

Network-to-Network Signaling

Typical In-Band signaling methods for network-to-network signaling on trunk lines are Single Frequency (SF), Multi Frequency (MF) or Robbed-Bit signaling. When SF is used, no tone is sent while the circuit is up. When one party hangs up, a disconnection is signaled by sending a 2600 Hz tone over the circuit. MF uses different frequencies to signalize either events like seizure, release, answer or acknowledgement or to send information like the phone number. While SF and MF

is used for analog telephone systems, Robbed-Bit signaling was developed for digital ones. Here, the least-significant bit from the frames of the voice bit stream is dedicated to signaling.

Now let us take a look at Signaling System 7 (SS7), the most widespread type of Out-of-Band signaling. It was defined by the International Telecommunications Union (ITU) in 1980 and includes three classes of devices: Service Switching Points (SSPs) are switches that originate or terminate calls, Service Control Points (SCPs) offer access to databases with additional routing information and Signal Transfer Points (STPs) route SS7 messages. These devices are called SS7 nodes.

The SS7 protocol stack consists of four layers. Message Transfer Part (MTP) 1, MTP 2 and MTP 3 are equivalent to the three layers in the Open Systems Interconnection (OSI) reference model (physical, data link and network layer). Following protocol sets form layer 4: Telephone User Part (TUP), ISDN User Part (ISUP), Transaction Capabilities Application Part (TCAP) and Signaling Connection Control Part (SCCP). TUP was created to perform basic phone calls. Because it does not support ISDN and intelligent network functions like call forwarding or selective call blocking, ISUP was developed. It originates, manages and terminates ISDN and non-ISDN connections. TCAP enables connections to external databases. The obtained information is transported in form of a TCAP message. Finally, SCCP provides end-to-end routing and is required to route TCAP messages to their proper database.

More information about SS7 and other signaling protocols can be found in [Davidson and Peters, 2000] or in [Russel, 2000].

2.2 An Overview of IP Telephony

Figure 2.2 shows several possibilities to use IP telephony. If people use their computers to communicate they must be online to receive calls. It is also possible to use gateways that connect a packet-switched network with PSTN. Then, connections from and to a conventional telephone are possible, too. A third alternative would be that both dialog partners use telephones but the call is routed through a packet-switched network.

Note that the terms IP telephony and Voice over IP (VoIP) in this thesis are interchangeable. Although some documents define IP telephony as a subset of VoIP, there is no standard that specifies any differences.

2.2.1 Advantages

IP telephony was developed for the transmission of speech-data over the Internet Protocol. One of its most important advantages is, that a voice transmission over a packet-switched network is much cheaper because of the more efficient use of bandwidth. While a phone call over the PSTN needs a full-duplex 64 kbps channel for the duration of the call a VoIP-transmission requires about 14 kbps with

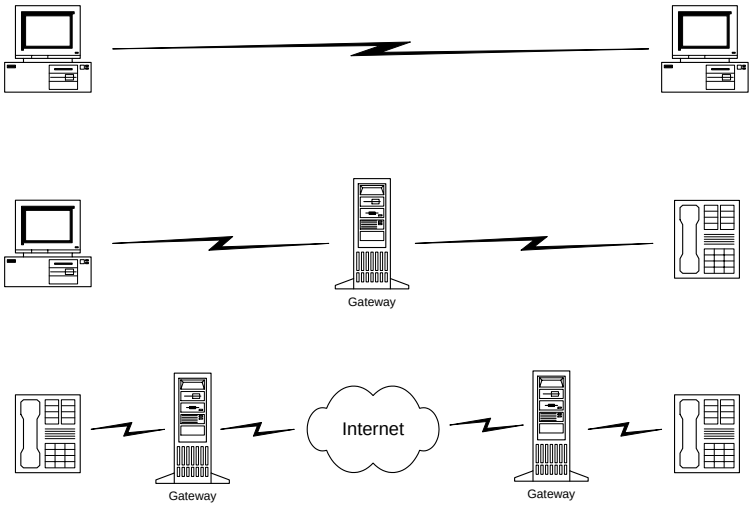


Figure 2.2: Possible ways to use IP telephony

compression. This bandwidth is used only when something has to be transmitted. Another advantage is, that just one instead of two different networks has to be maintained. Either computers with multimedia capabilities or IP phones that can be plugged into a conventional RJ-45 socket can be used for phone calls.

People often do not use just one kind of communication service. They send and receive emails or faxes, make phone calls or participate at video conferences. It is very hard to integrate all these services when different kinds of networks are used. If all services run over a packet-switched network, it is possible to use all these communication services within one application. A Unified Messaging System (UMS), which is described in chapter 7 realizes such an integration.

Another advantage of VoIP is the much easier development of additional services. This leads to a higher functionality of VoIP systems.

2.2.2 Problems

The quality of IP telephony depends on the network that transports the speech-packets. While a Local Area Network (LAN) can send these packets in an appropriate time and with a negligible packet-loss, a Wide Area Network (WAN) often can not. This is a big problem for VoIP solutions. Normal data-packets are almost not very time-critical, but VoIP-data needs a guaranteed QoS. The goal of QoS is to provide the bandwidth and latency an application needs. This can be realized by prioritizing the data, whereas time-critical data are transferred faster than non-critical.

Latency

Latency or delay is the duration of the voice from the speaker to the listener. The ITU-T G.114 recommendation defines 150 ms as limit for a good quality. This limit is often unreachable, but up to 300 ms delay is in the majority of the cases also accepted. Three kinds of latency are defined [Davidson and Peters, 2000]:

Propagation Delay: Because the light travels in a vacuum with a speed of 300000 kilometers per second and the electrons in copper cables or the light in light-wave cables travel with 200000 kilometers per second it takes some time to pass the route from the sender to the receiver. For example a signal over a light-wave cable around the half globe takes about 100 ms. Propagation delay is not avoidable, but often negligible.

Handling Delay: This kind of latency depends on the active components. The more routers and switches a packet has to pass, the more delay arises. The coding and decoding of the audio signal takes about 20 ms each. When proxies or firewalls must be passed, a delay up to 500 ms can occur. Of course, such latency would be unacceptable.

Queuing Delay: Queuing delay occurs when a component like a switch or a router must handle more packets than it can process. It is dependent on the degree of utilization of the network. A queue at the endpoint is needed to reduce the influence of jitter on the overall delay.

Jitter

When sending two voice-packets with a defined offset, but these two packets are received with a larger offset, this effect is called jitter. Jitter also affects the latency of the voice-transmission, because the more jitter appears the larger the queue at the receiver must be. Under normal conditions, this queue should store a packet for at least 30 ms.

Echo

In analog telephone systems, not tuned electric impedances at transitions between 4-wire trunks and 2-wire local-loops can cause signal reflections. Modern telephones can suppress these echos. In packet-based networks so-called echo cancellers are used. When User A talks to User B the echo canceller stores the voice-signal. Then it is adding the negation of the stored signal at the appropriate place to the signal from User B.

If a computer as endpoint is used, also feedback must be kept in mind. When User A talks to User B, its voice is emitted by the loudspeaker and is then, through the microphone of User B, retransmitted to the originator. A good way to reduce or even eliminate feedback is to use headsets.

Packet Loss

In an overloaded network it frequently can happen that routers discard data-packets. If TCP is used the lost packet is retransmitted. This is an excellent strategy when sending conventional data, but for streamed-data it is not a good idea. That is why VoIP uses UDP as underlying protocol.

When a packet-loss occurs, the receiver has several possibilities. It can either send nothing (silence), send a noise or replay the last received packet (concealment-strategy). If just one packet was lost, silence is the worst, while replaying a packet is the best solution. Another possibility is to add to every transmitted packet the last transmitted one. This alternative wastes more bandwidth than the concealment-strategy does.

2.2.3 Voice Encoding

This section deals with the conversion of the analog voice-signal into a (compressed) digital signal and vice versa. The requirements for the digital signal are a high quality and a low bit rate. Coding methods which produce a very low bit rate often need a fast hardware to compute the bit stream in an appropriate time. In contrast, coding methods that enable a high voice-quality with the drawback of a high bit rate need less computational power but require a faster network. Therefore it is very important to adjust the coding method to the given equipment.

The main speech coding techniques can be divided into waveform codecs, source codecs and hybrid codecs [Woodard, 2002]. Figure 2.3 shows the varying of bit rate and speech quality for the three classes.

Waveform Codecs

Waveform codecs quantize the original signal without any knowledge of the type of the signal. Therefore they are signal independent and work also for non-speech signals. Low computational costs and a good speech-quality are the advantages of waveform codecs.

The most popular waveform codec is Pulse Code Modulation (PCM). At the beginning the amplitude and the bandwidth of the analog signal is limited. Then the amplitude of the signal is measured with a fixed rate (sampling rate). The duration between two samples is defined by the Shannon-theorem, which states that a signal can be sampled without a loss of information when the sampling-frequency is at least twice than the highest frequency that occurs in the analog signal. A higher sampling rate does not lead to a higher quality of the signal and is therefore not necessary. So the sampling rate for a signal that is limited to 4 kHz should be at least 8 kHz. The precision of the measured value is defined by the number of bits used for one value. The error that occurs by converting the analog value to a digital one is called quantization error. Since this error is more significant for lower signal levels, often nonlinear converters are used. This means that digital values at lower levels lie closer together than values at higher levels. Two nonlinear quantization

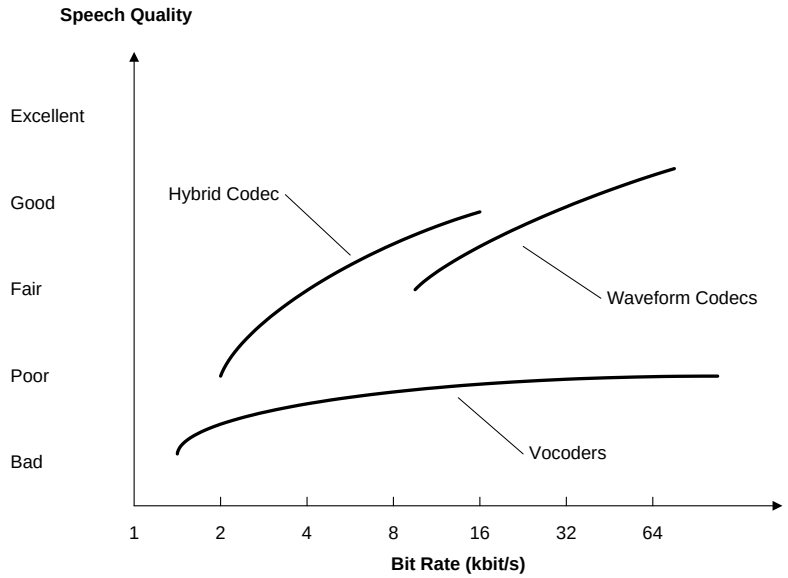


Figure 2.3: Speech Quality versus Bit Rate for Common Classes of Codecs (from [Woodard, 2002])

schemes are standardized: μ-law is used in North America, Japan and South Korea while A-law is used in the rest of the world.

To reobtain the analog signal, the quantized values are converted to analog ones with the same sampling rate as in the encoding phase. To smooth the resulting signal, it is sent through a lowpass filter. The ITU-T defines G.711 as a PCM-coder for narrowband signals (3,4 kHz) [ITU, 1999a]. G.711 is part of H.323 and produces a bit rate of 64 kbps with a delay of about 5 ms. The speech-quality of G.711 is often used as a reference for other coding methods.

A variation of PCM is the Differential Pulse Code Modulation (DPCM) where the PCM values are encoded as differences between the previous and the predicted current value. Because this difference has a lower variance than the original signal, the number of bits required for an audio signal can be reduced by about 25 percent.

An improvement of DPCM is the Adaptive Differential Pulse Code Modulation (ADPCM). This method tries to adjust the parameters for prediction and quantization to the input-signal. This leads to lower bit rates with the drawback of higher computational costs. An example for an ADPCM-standard is G.721 [ITU, 1988a] with 32 kbps, that was later integrated into the ITU-T G.726 standard [ITU, 1990].

All the codecs described above analyse the signal in the time domain. But there exist waveform codecs that also use the frequency domain. An example of such a codec is Sub-Band Coding (SBC). In this method the signal is split into several frequency bands. Each of these bands is then coded using PCM, DPCM

or ADPCM. The splitting into frequency bands is done because some frequencies are perceptually more important than others. These frequency bands can be coded with a higher bit rate and therefore lower noise which leads to an improved quality of the speech-signal. The ITU-T standard G.722 specifies an SBC codec for 7 kHz signals [ITU, 1988b]. It produces 48, 56 or 64 kbps with a delay of about 3 ms.

Source Codecs

Source codecs for speech are called vocoders. In contrast to waveform coders, they do not try to reproduce the original signal but to detect relevant parameters in the signal. In doing so, the content is preserved while the sound of the speaker's voice is lost. Because just the detected parameters are transferred, the bit rate of vocoders is significant lower than for waveform coders. Despite of this advantage, vocoders are not used in IP telephony because the speakers can not be identified by the sound of their voice. Source codecs are mainly used in military applications where low bit rates are more important than natural sounding speech.

An example for source coders are so-called Linear Predictive Coding (LPC) vocoders. They process an LPC analysis, where the signal is split into small segments. It is assumed that the characteristic of the signal does not change within a segment. The predicted values in a segment are generated by using a linear combination of the previous values. The coefficients of the linear combination (LPC coefficients) are specified by the minimization of the difference between sampled and predicted value. The quality of an LPC-vocoder depends on the number of LPC coefficients and the duration of a segment.

Hybrid Codecs

Hybrid Coders use methods of waveform and source codecs, whereas they try to minimize their disadvantages. Hybrid codecs usually have bit rates between 2 and 16 kbps.

The Residual Excited Linear Prediction Coding (RELP) uses a method where the signal is represented by the coefficients of an LPC analysis and a waveform-coded residual signal. The LPC coefficients are recalculated every 10 to 20 milliseconds. RELP coders usually have bit rates between 4.8 and 16 kbps. A variant of RELP is used by the Global System for Mobile Communications (GSM). This codec has a bit rate of 13 kbps.

To reduce the bit rate, Code Excited Linear Predictive Coding (CELP) does not quantize each sampled value for its own (scalar quantization) but uses vector quantization. This method combines several sampled values to a vector. The coder compares this vector with reference vectors listed in a codebook and transfers the code of most similar reference vector.

There are several standardized hybrid codecs. Low Delay CELP (LD-CELP) is a CELP-coder that is defined as G.728 [ITU, 1992]. Its advantage is a very low processing delay of about 2 ms. Its bit rate is 16 kbps. Another standard is

G.732.1 with bit rates of 6.3 kbps and 5.3 kbps, respectively [ITU, 1995a]. This standard either uses Multipulse Maximum Likelihood Quantization (MP-MLQ) or Algebraic CELP (ACELP) as coding schemes. Finally, G.729 [ITU, 1996a] uses Conjugate Structure - Algebraic Code Excited Linear Prediction (CS-ACELP) as coding method. Its bit rate is 8 kbps. All of the standards mentioned above are part of H.323.

Mean Opinion Score

The quality of a reconstructed signal is an important characteristic of a voice codec. Therefore it is necessary to measure the quality. While humans perform subjective tests, a computer measures the quality of a signal in relation to the original signal in an objective way. A subjective measurand for the evaluation of a codec is the Mean Opinion Score (MOS). A MOS-test of a codec is performed as follows. A group of people listens to different samples of voice recordings and rates each recording with one (bad quality) up to five points (excellent quality). The mean of the ratings of all people states the MOS score of a codec. Table 2.1 shows the MOS scores for some ITU-T codecs.

Compression Method	Bit Rate	Framing Size	MOS Score
G.711 PCM	64 kbps	0,125 ms	4,1
G.711 PCM	64 kbps	0,125 ms	4,1
G.726 ADPCM	32 kbps	0,125 ms	3,85
G.728 LD-CELP	15 kbps	0,625 ms	3,61
G.729 CS-ACELP	8 kbps	10 ms	3,92
G.729a CS-ACELP	8 kbps	10 ms	3,7
G.723.1 MP-MLQ	6,3 kbps	30 ms	3,9
G.723.1 ACELP	5,3 kbps	30 ms	3,65

Table 2.1: ITU-T codec MOS values (from [Davidson and Peters, 2000])

2.3 Related Communication Services

In addition to IP telephony there are other techniques from the PSTN domain which have alternatives based on IP. This section explains Fax over IP (FoIP), video over IP and data conferencing. While the first one becomes less important and is more and more replaced by emails, the popularity of video transmissions over the Internet rises due to the increasing bandwidth of backbones and last-mile technologies. Also data conferencing is becoming more and more important.

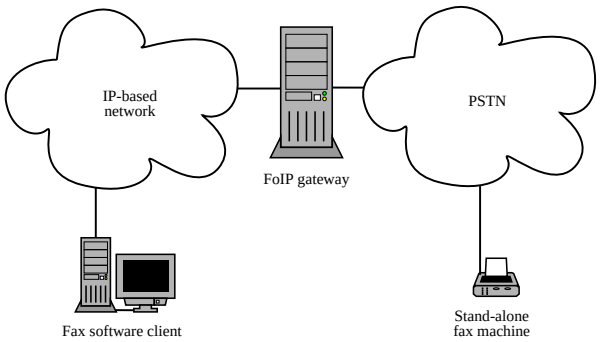


Figure 2.4: Fax over IP Architecture

2.3.1 Fax over IP

FoIP provides the possibility to send faxes over an IP-based network. An endpoint of a connection can be a software application on the IP-network side or a conventional fax device on the PSTN side. The translation between Internet and PSTN is realized by a FoIP-gateway (see Figure 2.4) [Telogy Networks, 1998]. When both endpoints are conventional fax devices and the message is transferred over a packet-based network, it passes two gateways.

A gateway can be implemented as a store-and-forward or as a real-time gateway. When using a store-and-forward gateway the message is stored at the gateway until the endpoint confirms the reception of the fax-message. The gateway then forwards the confirmation of the correct transmission to the originator. The protocol for store-and-forward FoIP was specified by the ITU-T in T.37 [ITU, 1998k]. A better solution is real-time FoIP, because it complies with conventional fax-sending. While store-and-forward FoIP simulates an existing connection to the receiver, this FoIP method directly sends the message to the recipient. T.38 [ITU, 1998l] is the standard for real-time FoIP. The protocol used on the PSTN-side is called T.30 [ITU, 2001].

In contrast to voice over IP, FoIP does not allow lost data. Therefore gateways can use T.30 to inform the fax machine to delay or to interrupt the transmission or to resend parts of the message.

Because real-time FoIP requires guaranteed delay, which is possible in local area networks but not on the Internet, *near real-time FoIP* is a compromise between store-and-forward and real-time FoIP.

A main aspect why companies migrate to FoIP are the reduced costs. Overseas connections are very expensive and therefore it is a good idea to send the data via a FoIP-gateway. If a company does not have an own gateway, Internet-fax service provider offer the possibility to send messages via their gateway. If both communication partners have an IP-connection and a fax software, a gateway is not necessary. Another advantage of FoIP is the much easier integration into a unified messaging system (see chapter 7) where faxes can be converted into emails

and vice versa. An important point which is often ignored is the security aspect. Unfortunately, a fax machine is usually used by the complete department or even the entire company. Insofar it can happen, that employees can read the faxes of colleagues. If FoIP in combination with a unified messaging system is used, the faxes are directly sent to the inbox of the recipient of the message.

2.3.2 Video over IP

Although H.323 is known as a standard for VoIP, it was also specified for video transmissions over IP and data conferencing (see section 2.3.3). This section introduces the basics of video telephony and video conferencing, other kinds of video over IP, like video on demand or real-time video are not explained.

While audio codecs are mandatory elements on H.323 terminals, video codecs are optional elements. However, if they exist, video telephony with two users and video conferencing with two or more users are possible. If connections to the PSTN are required, the gateways must be extended in a way that they can communicate with conventional ISDN or PSTN video phones and video conferencing systems. Video phones normally use H.320 (for ISDN) or H.324 (for PSTN) as communication protocol (see section 3).

When a multipoint video conference is performed, the Multipoint Control Unit (MCU), which manages the conference (see chapter 3), must decide which of the incoming data streams are sent to the other participants. The audio channels of these streams are simply mixed together and sent to all endpoints. For the video channels there are several possibilities. The incoming video streams can be merged to one video image (i.e. the resulting video image includes the downsized video images of each participant). Alternatively the resulting video stream can be one of the incoming streams. For example if a teacher performs a video conference with his pupils (distance learning) it is a good idea to lock the focus on the teacher. Finally, the MCU can transmit the video stream of the most prominent (loudest) participant to the others.

Video Codecs

There are several standards defined to encode and decode a video stream. If an H.323 terminal supports video conferencing, it should at least have H.261 QCIF implemented, which is explained later.

To encode and decode a video stream, following steps are necessary [Köhler, 2002]:

- Recording and digitization of data
- Compression and encryption (on sender-side)
- Decryption and decompression (on receiver-side)

At this point, just the compression/decompression part is discussed. An often used technique for compressing a single image or individual pictures in a video is called intraframe compression. As compression method JPEG (Joint Photographic Experts Group) or JPEG-2000, which is based on the wavelet-technology are used. Because intraframe compression leads to high bit rates the interframe compression technique is mostly preferred. Because there are relatively little changes from one video frame to the next one, this technique exploits the similarities between two frames. This reduces the volume of data enormous.

The Moving Pictures Experts Group (MPEG) defined some standards that are based on interframe compression. All these standards have three different types of frames:

I-Frame (intra-coded frame): The I-frame is the only type of frame that uses intraframe compression. It is stored in the JPEG-format and can be decoded without any information of other frames in the video stream.

P-Frame (predictive-coded frame): In a P-frame just the differences to the last I- or P-frame are stored. Therefore these frames are also needed for the decompression of this frame.

B-Frame (bidirectionally predictive-coded frame): This type is similar to the P-frame, except that not only preceding but also succeeding frames are used for encoding this frame.

The more P- and B-frames are used in a video stream the smaller is the needed bandwidth for the transmission of an MPEG video.

The ITU-T also defined standards for video compression. H.261 [ITU, 1993a] describes a coding method for compressing a standard television color-signal into a video stream with a bandwidth from 64 kbps to 2 Mbps. It includes two formats: The Common Intermediate Format (CIF) has a resolution of 352x288 pixels with 30 pictures per second. The Quarter Common Intermediate Format (QCIF) defines a resolution of 176x144 pixels which is a quarter of the pixels of CIF. QCIF provides between 7.5 and 15 pictures per second. Sub-QCIF (SQCIF) again has a smaller resolution than QCIF (128x96 pixels).

H.263 [ITU, 2000c] is an ITU-T standard that was defined to offer video compression with higher resolution and lower bit rates (between 15 kbps and 20 kbps) than H.261.

Table 2.2 lists the resolutions and the bit rates of H.261 and H.263 as well as some MPEG-codecs. Note that the listed resolutions of the MPEG-codecs are just the most popular ones.

2.3.3 Data Conferencing

T.120 [ITU, 1996b] is a data conferencing standard that is part of H.323. It simplifies collaboration by allowing users to share data and applications, or use whiteboards from different computers. In fact, T.120 is not a single protocol but defines a

Compression Method	Resolution	Bit rate
MPEG-1	352 x 240	1.2 Mbps
MPEG-2	352 x 240 720 x 480 1920 x 1080 1440 x 1152	typical: 4 to 9 Mbps but up to 80 Mbps
MPEG-3	176 x 144 1920 x 1080 1550 x 1152	20 to 40 Mbps
MPEG-4	176 x 144	5 kbps to 4 Mbps
H.261	352 x 288 (CIF) 176 x 144 (QCIF)	64 kbps to 2 Mbps
H.263	128 x 96 (SQCIF) 176 x 144 (QCIF) 352 x 288 (CIF) 704 x 576 (4CIF) 1408 x 1152 (16CIF)	15 to 20 kbps

Table 2.2: Overview of some video standards (from [Köhler, 2002])

suite of protocols on the networking and applications level to enable real time multimedia transmissions, multipoint data connections and conferencing. It is beyond the scope of this thesis to explain all these protocols in detail. Just an overview of some features is given.

Protocols on the networking level are T.122, T.123, T.124 and T.125. T.122 [ITU, 1998h] is a standard for multipoint services. It allows the participants of a conference to send data to the other ones. T.123 [ITU, 1999c] is responsible for transporting and sequencing data, and for controlling the flow of data across networks. It includes also an error-correction for a reliable data transport. T.124 [ITU, 1998i] provides the generic conference control for the initiation and administration of multipoint data conferences and T.125 [ITU, 1998j] specifies defines private and broadcast channels to transport the data. Together, T.122 and T.125 make up the T.120 multipoint communication services.

The T.126 and T.127 standards define the applications level of T.120. T.126 [ITU, 1997] enables whiteboard functionality. It specifies how applications must send and receive data. Data transfer can either be compressed or uncompressed. T.127 [ITU, 1995b] is responsible for the correct file exchange among conference participants.

2.4 Summary

To establish a connection in the PSTN, two kinds of signaling protocols are used: user-to-network and network-to-network signaling. An example for analog user-

to-network signaling is DTMF. ISDN has a separate data channel for the signaling information. The most widespread type of network-to-network signaling is SS7, which uses Out-of-Band signaling.

IP telephony has some advantages compared to common telephony systems. It uses the bandwidth more efficient. This is due to speech-data is being transmitted over a packet-based network. Also the integration of different communication services is much easier if the same type of network is used. Latency, jitter, echo and packet loss must be considered when implementing an IP telephony system.

The main speech coding techniques can be divided into waveform, source, and hybrid codecs. While source codecs create low bit rates, waveform codecs generate a good speech quality. Hybrid codecs use the methods of source and waveform codecs, trying to minimize the disadvantages of both.

FoIP, video over IP and data conferencing are communication services related to IP telephony. FoIP offers the possibility to transmit faxes over the Internet. video over IP is based on the same protocols as IP telephony, but needs more bandwidth. Also other kinds of codecs, like MPEG-4 are used. Data conferencing allows users to share data and applications or use white boards.

Chapter 3

The H.323 Protocol Suite

H.323 is a standard from the ITU-T for multimedia collaboration on packet-based networks like IP or Asynchronous Transfer Mode (ATM). It consists of several ITU-T recommendations which include protocols for the interaction of H.323 components and the communication with Switched Circuit Networks (SCNs). Figure 3.1 shows some of the H.323 protocols on the TCP/IP protocol stack.

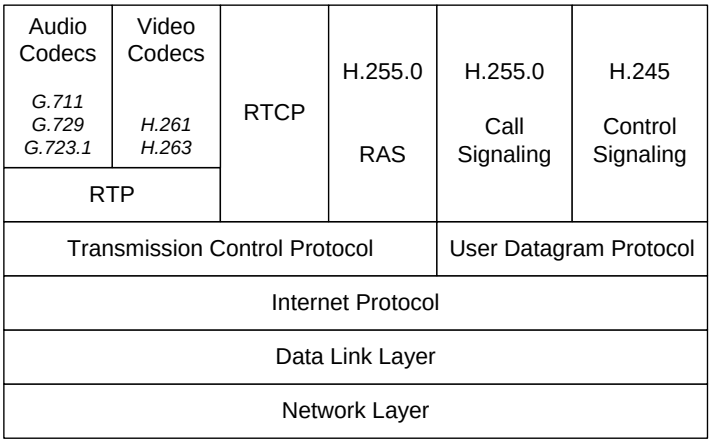


Figure 3.1: H.323 protocols on TCP/IP stack

This chapter describes the architecture of H.323 and defines the components that are collaborating. Finally, the most important protocols of the H.323 protocol suite are explained. A detailed description of the H.323 standard can be found in the ITU-T recommendation H.323 [ITU, 1999b] or in [Kumar et al., 2001].

3.1 Architectural Overview

Figure 3.2 shows an example of two H.323 IP telephony networks which are connected to the Internet and to PSTN. Both networks include elements like terminals, gateways and gatekeepers. H.323 zone B also includes a Multipoint Control Unit (MCU). At the PSTN-side a conventional telephone and an H.324 video phone are connected.

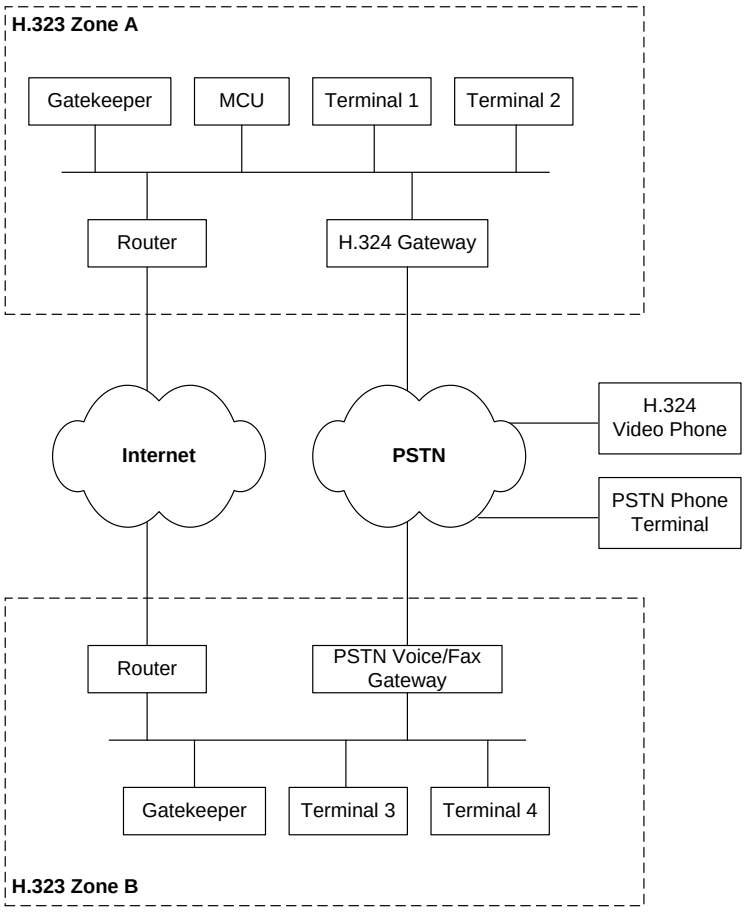


Figure 3.2: H.323 IP telephony networks

The term *H.323 zone* was defined by ITU-T as follows:

> A zone is the collection of all terminals, gateways and MCUs managed by a single gatekeeper. A zone includes at least one terminal, and may or may not include gateways or MCUs. A zone has one, and only one gatekeeper. A zone may be independent of network topology and may be comprised of multiple network segments which are connected using routes or other devices [ITU, 1999b, p. 8].

3.1.1 Terminal

An H.323 terminal can be a personal computer (PC) with multimedia capabilities or a stand-alone device. It must include an H.323 protocol stack. The basic service of a terminal is audio communication. Optional services are video or data communication. Audio and video streams run over the Real-Time Transport Protocol (RTP). The Real-Time Transport Control Protocol (RTCP) provides feedback on the quality of the transmission. Another function a terminal must implement is Registration, Admission and Status (RAS). It is used to register endpoints (terminals and gateways) at gatekeepers. RAS also supports admission control and bandwidth changes. Call signaling provides functions to establish a connection between endpoints and control signaling is used to send control messages.

3.1.2 Gateway

A gateway is an element that links different networks (e.g. a packet-based network with the PSTN). For this purpose they convert media formats and translate the protocols for call setup and release. Gateways act on the H.323-side like an H.323 terminal and on the SCN-side like an SCN terminal. Therefore, gateways have to implement services like RAS, call signaling, control signaling, RTP and RTCP. If no connection to another type of network or protocol is needed, gateways are not necessary.

Modern gateways are split into two functional units: a Signaling and a Media Gateway. A Signaling Gateway connects to SS7 and provides signaling control information to the Media Gateway Controller (MGC) or Call Agent. The MGC takes this information and provides call routing control and billing information. Finally, the Media Gateway transfers and converts the different media streams from one network to another. Figure 3.3 shows a distributed switch architecture.

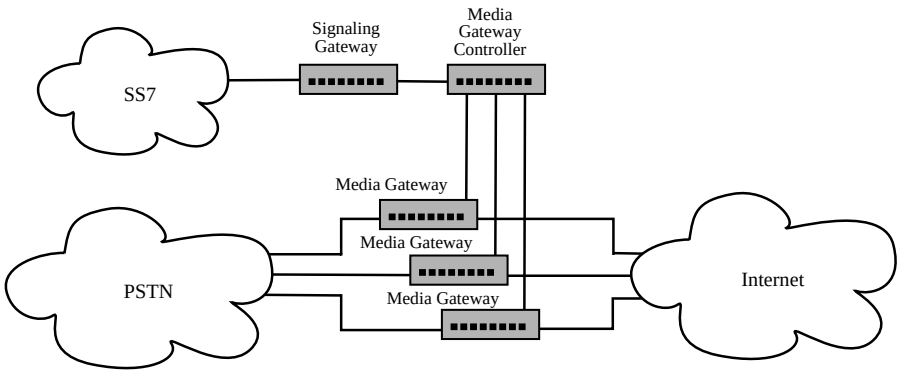


Figure 3.3: Distributed Switch Architecture

The protocol the Media Gateway uses to communicate with the Media Gateways is called Media Gateway Controller Protocol (MGCP). A newer standard

jointly specified by ITU and IETF is called H.248 (by ITU) or Megaco (by IETF) [Cuervo et al., 2000].

3.1.3 Multipoint Control Unit

An MCU is an endpoint which provides support for multipoint conferences. It shall consist of a Multipoint Controller (MC) and zero or more Multipoint Processors (MPs) [ITU, 1999b, p. 30].

The MC checks the capabilities of three or more endpoints and then sends the possible operating modes to all endpoints attending at the conference. The MP reads one or more input streams, switches or mixes them and then sends the resulting stream to all endpoints in the conference.

3.1.4 Gatekeeper

A gatekeeper is an optional element in an H.323 environment. However, if it exists in a network, it must be used by terminals and gateways. Gatekeepers provide several call control services for H.323 endpoints. Some of these services are mandatory while others are optional.

Mandatory services a gatekeeper must provide, include address translation (IP address to standard E.164 telephone number) or admissions control. Also bandwidth control and zone management must be provided.

Optional services include call-control signaling (based on the Recommendation H.225 of the ITU-T), call authorization (a gatekeeper can accept or reject a call depending on access-based or time-based restrictions to and from terminals or gateways) or call management (e.g. rerouting a call to achieve load balancing).

3.2 H.225.0 Call Signaling Protocols

The H.225.0 recommendation [ITU, 2000a] from the ITU includes H.225.0 RAS for the registration of an endpoint at the gatekeeper and H.225.0-Q.931 for setting up and terminating a call between two endpoints. All kinds of H.225.0 messages (as well as the H.245 messages described in section 3.3) are ASN.1 (Abstract Syntax Notation One) [ITU, 1994a] encoded. This is a binary format based on so called Packet Encoding Rules (PER) [ITU, 1994b].

3.2.1 H.225.0 RAS - Registration, Admission and Status

H.225.0 RAS defines a protocol between an endpoint (terminal or gateway) and the gatekeeper. RAS messages are transferred over an unreliable protocol like the User Datagram Protocol (UDP). Therefore, timeout checks and retries must be considered when implementing the protocol.

H.225.0 RAS provides a separate channel - the RAS channel - to transfer RAS messages. They are used to find the gatekeeper, to register endpoints at the gatekeeper or to check the admission of an endpoint to initiate or to receive a call.

Gatekeeper Discovery

Before an endpoint can register and authenticate at the gatekeeper it has to identify it in the network. If the gatekeeper discovery is done statically, then the endpoint is preconfigured with the transport address of the gatekeeper. If the discovery is done dynamically the endpoint multicasts a *Gateway Request (GRQ)* message. This *GRQ* message includes the transport address of the endpoint, the identifier of the gatekeeper that normally provides the service to the endpoint (or a null string) and a series of alias addresses that identify the endpoint. When a gatekeeper receives a *GRQ* message it determines whether or not to allow the endpoint to use the services it provides. If the endpoint is accepted, the gatekeeper will reply with a *Gateway Confirm (GCF)* message which includes the address and the identifier of the gatekeeper. If the endpoint is not accepted, the gatekeeper sends a *Gateway Reject (GRJ)* message.

Endpoint Registration

The next step after discovering the gatekeeper is to register the user and the associated endpoint with the gatekeeper. Therefore, it sends a *Registration Request (RRQ)* message. This message includes the transport address of the endpoint at which the gatekeeper should send H.225.0-Q.931 messages (see subsection 3.2.2). This address is necessary when the signaling is routed via the gatekeeper (gatekeeper routed call). When the gatekeeper receives an *RRQ* message it performs checks to determine whether or not to allow the endpoint to register. If the endpoint is allowed to register, the gatekeeper sends a *Registration Confirm (RCF)*, otherwise a *Registration Reject (RRJ)* message. The *RCF* message includes the transport address of the gatekeeper where the endpoint should send H.225.0-Q.931 messages to. By registering at a gatekeeper, the endpoint joins an H.323 zone.

Admission Control

Admission control enables the gatekeeper to authorize each outgoing or incoming call to and from an endpoint. By sending an *Admission Request (ARQ)* message the endpoint requests an alias address resolution and a call authorization. The gatekeeper determines the address and checks the authorization for the call. For the authorization it may contact a user policy server. A user policy server handles a profile for each user. Such a profile defines the geographic locations where calls can be made, or restrictions on the duration of a call. If the gatekeeper allows the user to make the call it sends an *Admission Confirm (ACF)*, otherwise an *Admission Reject (ARJ)* message.

Accounting

For billing it is important to know the participants as well as the duration of a call. Because H.225.0-Q.931 signaling is between endpoints, the gatekeeper must be notified of the start and the release of a call. This is done by sending an *Information Request (IRR)* message to the gatekeeper. When a call was released either the gatekeeper itself or an accounting server generates a Call-Detail Record (CDR). This CDR includes information like starting time, duration, identification of the two parties and so on.

Call Termination

Not just due to billing but also because of bandwidth reasons the gatekeeper must be notified when a call was released. If a gatekeeper has allocated bandwidth for a call it then can reallocate them. Therefore, the endpoint sends a *Disengage Request (DRQ)* message which includes the call identification. But also the gatekeeper can use the *DRQ* message to force the endpoint to terminate a call. This is necessary for example if a user of a prepaid card has run out of credit.

User Deregistration

To cancel a registration of a user, either the gatekeeper or the endpoint sends an *Unregistration Request (URQ)* message to the other side, which replies with an *Unregistration Confirm (UCF)* message. If an endpoint sends a *URQ* message for a registration that does not exist, the gatekeeper replies with an *Unregistration Reject (URJ)* message.

If an endpoint fails unexpectedly it cannot unregister anymore. This can cause problems when a gatekeeper is at full capacity and has to reject new registrations. To avoid such situations either the gatekeeper can ping all registered endpoints periodically and delete the not responding ones or the endpoint can provided a heartbeat in defined intervals. The latter method is called lightweight registration.

3.2.2 H.225.0-Q.931 - Call Signaling

H.225.0-Q.931 defines the communication between two endpoints (point-to-point calls) for setting up a call and terminating a connection. It uses a subset from the Q.931 ISDN signaling messages, which enables an easy interconnection with ISDN networks. Each H.225.0-Q.931 message has the same information elements as the corresponding Q.931 recommendation message. A detailed description of the Q.931 protocol can be found in [ITU, 1993b].

H.225.0-Q.931 is running over a reliable protocol like TCP. Before call signaling can start, the transport address of the other endpoint must be known. The transport address consists of an IP address and a TCP port number. The default port number for H.225.0-Q.931 is 1720. To obtain a transport address from a URL the H.225.0 Annex G protocol is used.

The following scenario describes the call signaling messages between Terminal A and Terminal B, where Terminal A initiates the call.

Call Initiation

At the beginning of a call Terminal A connects to the H.225.0-Q.931 transport address of Terminal B. Then a setup message is sent to Terminal B. The setup message includes two transport addresses of terminal A. One is the H.225.0-Q.931 and the other is the H.245 transport address (see section 3.3). Terminal A will wait at least four seconds for a response from Terminal B before disconnecting.

Call Proceeding

If between Terminal A and Terminal B a gateway exists, this gateway can use a call proceeding message to inform Terminal A that the setup message was received but it can not be relayed to Terminal B within four seconds. The call proceeding message is optional.

Call Alerting

When Terminal B has received the setup message it informs the user of the incoming call. This can be done by popping a dialog box on the user-interface display or in form of a ringing tone. Then Terminal B sends an alerting message to Terminal A, which waits at least 180 seconds for another H.255.0-Q.931 message before disconnecting.

Call Connection

If the user at Terminal B takes the call a connect message is sent to Terminal A. This message includes the transport address of Terminal B for the H.245 call control (see section 3.3). Terminal A will then open a TCP channel for call control to the correct address. Now the call has been established and the transmission of multimedia data can begin.

Call Termination

When the call is finished or the user at Terminal B has not taken it, either endpoint sends a release complete message. Finally, both TCP connections are closed.

3.3 H.245 - Call Control

After the call between endpoints is established and before the multimedia transmission can begin the capabilities of the endpoints must be determined. This is necessary to use audio or video codecs that are available on both sides. The H.245 protocol provides messages to gather this information. A *Terminal Capability Set*

message includes the receiver's capabilities for audio/video transmissions, data applications and user input. The sender specifies the kind of data with the *Open Logical Channel* message, which includes the type of the data stream (audio, video or application data).

All messages defined in H.245 are ASN.1-encoded. A detailed description of the Call Control protocol can be found in [ITU, 2000b].

3.4 Supplementary Services

In addition to the basic telephony service, the ITU-T has defined a set of supplementary services in the H.450 protocols [ITU, 1998a]. Most of these services already exist in PSTN (a modern PBX provides up to 300 supplemental services). Examples for standardized supplementary services are call transfer [ITU, 1998b], call forwarding [ITU, 1998c], call hold [ITU, 1998d], call park and pickup [ITU, 1998e], call waiting [ITU, 1998f], and message waiting [ITU, 1998g]. A good introduction to this issue is [Kumar, 1999].

3.5 Summary

H.323 is a protocol suite for multimedia collaboration on packet-based networks. There are four components defined in the H.323 standard.

A terminal is an endpoint that initiates and terminates calls. It can either be a PC with multimedia capabilities or a stand-alone device.

Gateways link different networks. They convert media formats and translate the protocols for call setup and release.

An MCU provides support for multipoint conferences. It reads one or more input streams, switches or mixes them and sends the resulting stream to all connected endpoints.

Gatekeepers provide call control services like address translations, admissions and bandwidth control as well as zone management.

The H.323 standard consists of a number of protocols. Examples are the H.225.0 Call Signaling protocols and H.245 for Call Control. The H.225.0 recommendation includes H.225.0 RAS for the registration of an endpoint at the gatekeeper and H.225-Q.931 for setting up and terminating a call between two endpoints. H.245 provides functionality to determine the capabilities of the endpoints.

Chapter 4

Session Initiation Protocol

The Session Initiation Protocol (SIP) is a signaling protocol standardized by the IETF [Handley et al., 1999]. A good introduction to this protocol can be found in [Rosenberg and Shockey, 2000] and [Collins, 2001]. Although SIP can be used to establish IP telephony calls, it is not limited to that kind of sessions. Also sessions for instant messaging (see section 6.2), network games or video conferencing can be managed. SIP operates over any packet-based network, reliable or unreliable, but usually SIP uses UDP as underlying protocol.

In contrast to H.323 which is ASN.1-coded, SIP is a text-based protocol using the ISO 10646 character set in UTF-8 encoding. It is more similar to the Hypertext Transfer Protocol (HTTP) [Fielding et al., 1997] rather than to legacy signaling. A comparison between SIP and H.323 can be found in [Schulzrinne and Rosenberg, 1998] and [Nortel Networks, 2000].

This chapter first explains how SIP addresses are defined. Then the components of SIP are specified. Also the structure of a message and the different types of messages are described. Next, some scenarios like the registration of a user agent or the set-up of a call are shown. Finally, SIP-specific event notifications and the Session Description Protocol (SDP) are explained.

4.1 SIP Addressing

A SIP address is necessary to identify a user. A user who has a SIP address is globally reachable. Such an address has the form of a Uniform Resource Locator (URL) [Berners-Lee et al., 1998] and is similar to a mailto URL, i.e. *sip:userinfo@host*. The *userinfo* can either be a username or a telephone number. The *host* is a domain name or an IP address. A SIP URL can designate an individual, a group or a defined person in a group.

When the host indicates an Internet telephony gateway the *userinfo* part of the SIP address can specify a telephone number. This can either be a local or a global number (a global number begins with a '+'). Because this number could also be a valid username, a parameter denotes that it is a telephone number (e.g.

+1234567890@gateway.xy user=phone). SIP addresses can be inserted in a web-page. By clicking the address-link a connection to the specified user is initiated.

Because a SIP URL can often be guessed from the email-address of the user, SIP offers authentication and access control mechanisms.

4.2 SIP Components

The SIP specification [Handley et al., 1999] defines several components that are necessary to implement a SIP environment.

User Agent: A user agent is an end-device which can be implemented in hard- or software. It consists of a user agent client and a user agent server. The user agent client is a client application that initiates the SIP request while the user agent server listens for incoming calls and notifies the user when a call is requested.

Proxy Server: A proxy server is an intermediary program that acts as both, a server and a client for the purpose of making requests on behalf of either clients. Requests are serviced internally or by passing them on, possibly after translation, to other servers. A proxy server interprets, and, if necessary, rewrites a request message before forwarding it [Handley et al., 1999, p. 10].

Redirect Server: A redirect server is a server that accepts a SIP request, maps the address into zero or more new addresses and returns these addresses to the client. Unlike a proxy server, it does not initiate its own SIP request. Unlike a user agent server, it does not accept calls [Handley et al., 1999, p. 10].

Registrar: A registrar is a server that accepts *Register* requests. A registrar is typically co-located with a proxy or redirect server and may offer location services [Handley et al., 1999, p. 10].

4.3 SIP Messages

Messages in SIP are text-based and use the ISO 10646 character set in UTF-8 encoding. Lines must be terminated with a CR/LF. A SIP message is either a request from a client to a server or a response from a server to a client. A message sent by a SIP component consists of a start-line, a header, a blank line and an optional body. Figure 4.1 shows an example of an *Invite* message and its response from the server. This example is used to explain the most important elements of a SIP message.

The request message has following structure. The start-line states the kind of message, the callee's SIP URL and the SIP version. The *Via* header field describes the path of the request so far. The *From* field states the sender and the *To* field the

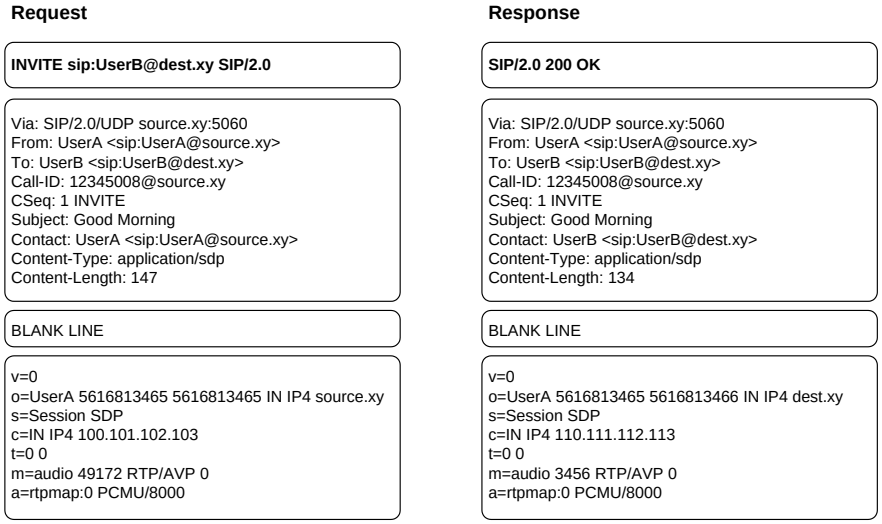


Figure 4.1: SIP Messages

recipient of the message. The *Call-ID* field uniquely identifies a particular invitation or all registrations of a particular client. This identifier should be generated by a good random number generator [Crocker and Schiller, 1994] to avoid session hijacking. Consecutive request with the same Call-ID must have an increasing *CSeq* header field. The *Subject* field can contain a string that describes the session while the *Contact* field provides information how the user can be reached for further communication. Finally, *Content-Type* and *Content-Length* describe the kind and the length of the content in the message body, respectively. In the example the content in the message body is in the SDP data format to specify the media type (audio), the codec (G.711) as well as the host name (100.101.102.103) and the port number (49172). SDP is explained in section 4.6.

In the response message the start-line includes the SIP version, the response code and a keyword that specifies the correct execution of the request. The *Contact* header field states the current location of the callee. This information can be used to directly contact the user and bypass any proxy servers. All other fields in the response message are similar to the request message.

The following list describes the standardized SIP messages. SIP extensions (e.g. SIP Extensions for Instant Messaging, see section 6.2) may define additional message types.

Register: The *Register* message binds a permanent SIP address to a current location. It includes a *To* header field which states the SIP URL and a *Contact* header field with the current contact address. If the *Contact* header field is empty the registrar returns all current contacts of the user. The *Expires* header field states how long the current location is valid (in minutes).

Invite: With this kind of message a client asks the callee to join a particular conference or to establish a two-party conversation. The message body contains a description of the session. Re-invites can be used to change the session state.

Options: This message is used to inquire the capabilities of a SIP component. Proxy and redirect servers simply forward an options message.

Ack: *Ack* is used to tell the callee that his confirmation of the *Invite* message was received.

Bye: The *Bye* message sent by the user agent client indicates that the user wishes to quit the session.

Cancel: If *Invite* requests are pending, the *Cancel* message is used to abort them.

The response codes in a reply message have the form "xyz explanatory text". Receivers of such a response code just have to understand "x".

1yz Informational

100 - Trying
180 - Ringing
181 - Call is being forwarded

2yz Success

200 - OK

3yz Redirection

300 - Multiple choices
301 - Moved permanently
302 - Moved temporarily
305 - Use proxy

4yz Client Error

400 - Bad request
401 - Unauthorized
402 - Payment required
486 - Busy here

5yz Server Error

500 - Server internal error
501 - Not implemented
503 - Service unavailable

6yz Global Failure

600 - Busy everywhere
604 - Does not exist anywhere

4.4 SIP Transactions

This section describes transactions to register an endpoint at a current location (section 4.4.1), to initiate and to terminate a session (section 4.4.2). A user agent client can initiate a session either directly or via an intermediate server. The intermediate server can act as a proxy and forward the request (section 4.4.3) or as a redirect server (section 4.4.4).

4.4.1 Registration

A user agent registers at a SIP registrar to bind the user's permanent SIP URL to a current location. This is realized by sending a *Register* message to the registrar. The *Register* message does not have to be sent to a dedicated registrar, also a multicast address ("sip.mcast.net") where a registrar listens to, can be used.

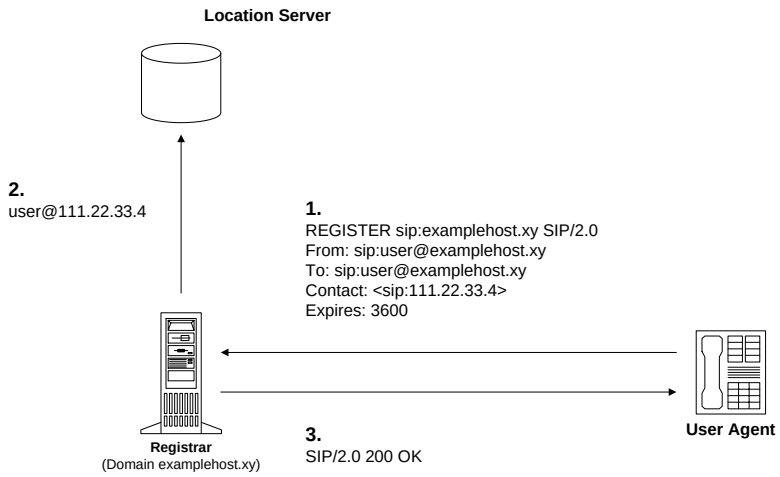


Figure 4.2: SIP Registration

Figure 4.2 shows an example of a registration. The user agent client sends a *Register* message to the registrar. Then the registrar forwards the SIP URL together with the current location (from the *Contact* header field) to a location server. A location server is used by a SIP redirect or proxy server to obtain information about a callee's possible location [Handley et al., 1999, p. 10]. Finally, the registrar sends a reply to the user agent.

If the user changes his location again he first sends a register message with the current location and sets the *Expires* field to zero. This removes the current location from the database. Then the user can register with the new location.

4.4.2 Session Initiation and Termination

When a person wants to establish a call, the user agent client first sends an *Invite* message to ask the callee to join the conversation. Then the callee can accept the call by sending a response (response code *200 OK*). Before the callee accepts a call the callee's user agent client can send responses with informational response codes (e.g. *100 TRYING* or *180 RINGING*). If the callee has accepted the call the caller confirms that it has received the response by sending an *Ack* request. If the caller no longer wants to communicate, it can send a *Bye* message. To release an existing call, one of the participants sends a *Bye* request. A party receiving a Bye request must stop transmitting media streams.

The following two sections describe the invitation to a session when between the two parties one or more a proxy or a redirect servers are located.

4.4.3 SIP Invitation in Proxy Mode

When an *Invite* message passes a proxy server, the server requests the current location from a location service. Then it adds itself on top of the *Via* header field list, so that the packet can be routed back the same path and forwards the *Invite* message to the current location. If the location server returns more than one location, the proxy server either can try the addresses one after another or parallel.

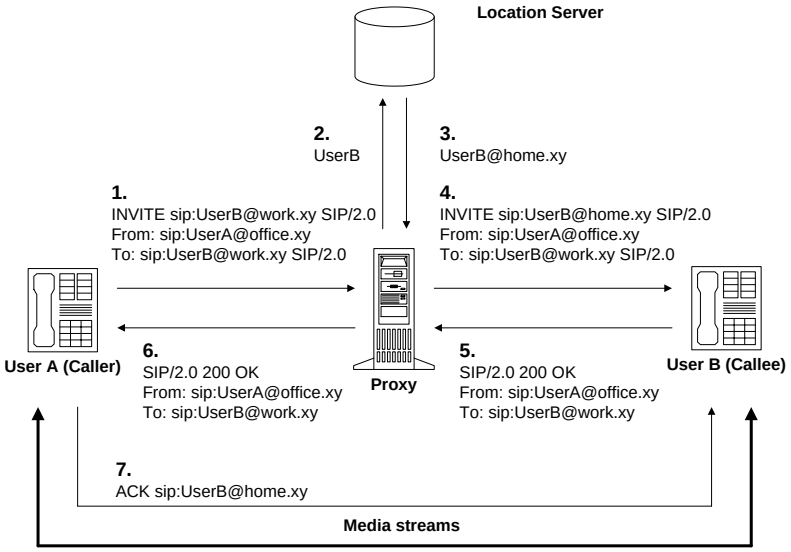


Figure 4.3: SIP Invitation in Proxy Mode

When the callee's user agent responses, each host on the path must remove its address from the *Via* field of the response. This is necessary to hide the internal routing information from the caller.

Figure 4.3 shows an invitation passing a proxy server. Note that *Ack* may be sent either directly to the callee using the callee's address from the *Contact* header field of the response message or via the proxy server again.

4.4.4 SIP Invitation in Redirect Mode

Using a redirect server is useful when a person has moved or changed the provider. Then the caller does not need to try the same server next time. The redirect server responses to an *Invite* message with a *301* (moved permanently) or a *302* (moved temporarily) response code. Figure 4.4 shows an example for the redirection of an *Invite* message.

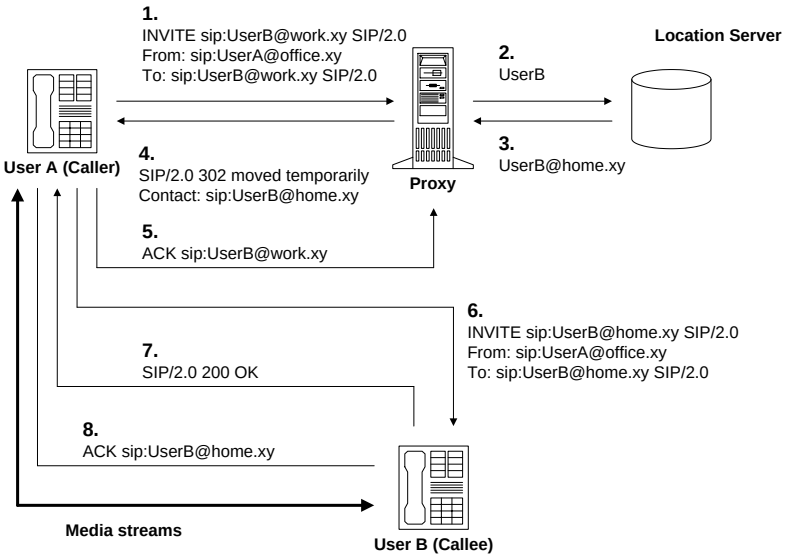


Figure 4.4: SIP Invitation in Redirect Mode

4.5 SIP-specific Event Notification

[Roach, 2002] specifies an abstract framework as an extension to SIP, which enables SIP nodes to request event notifications from remote nodes. Because this document does not describe concrete event notifications, for an implementation additional documents must specify further syntax and semantics. Such an additional document is herein referred to as *event package*. Section 6.2 describes an event package that defines presence events.

To receive the current state of a remote node (and to receive updates of the state) an entity (the subscriber) has to send a *Subscribe* message to the remote node. This node (the notifier) decides whether or not the subscriber is allowed to receive its notifications. If the subscription request was accepted, the notifier sends

immediately after the acknowledgement message a *Notify* message with the state of the node. When the state changes, the notifier sends again a *Notify* message to all accepted subscribers. Figure 4.5 shows a typical flow of messages.

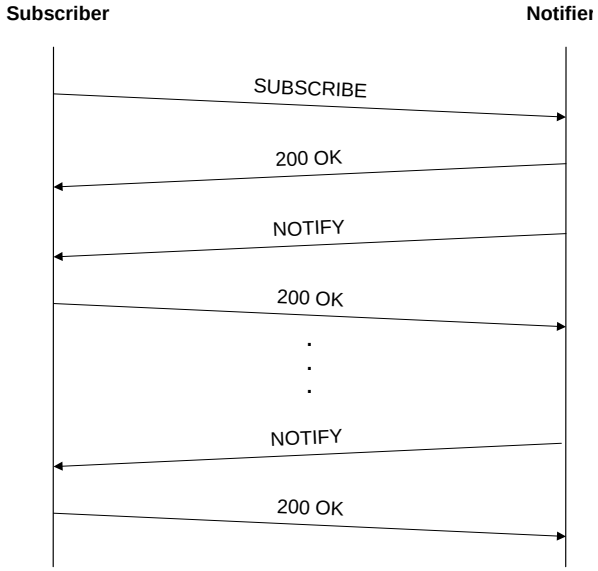


Figure 4.5: Typical flow of *Subscribe* and *Notify* messages (from [Roach, 2002])

Subscribers must include exactly one *Event* header in *Subscribe* requests. This header field indicates which event or class of events the subscription concerns. The *Expires* header field indicates the duration of a subscription. To unsubscribe, a *Subscribe* message with the *Expires* header set to '0' must be sent.

4.6 Session Description Protocol

Data in the body of an *Invite* message like shown in Figure 4.1 is often encoded using SDP [Handley and Jacobsen, 1998]. The purpose of this data is to provide information of the multimedia capabilities of the caller. The body of the response message includes the capabilities of the callee. SDP includes following information [Handley and Jacobsen, 1998, p. 4]:

- Session name and purpose
- Time(s) the session is active
- The media comprising the session
- Information to receive those media (addresses, ports, formats and so on)

- Information about the bandwidth to be used
- Contact information

To convey enough information to be able to join a multimedia session, following additional data about the kind of media used must be given [Handley and Jacobsen, 1998, p. 5]:

- The type of media (e.g. video or audio)
- The transport protocol (e.g. RTP/UDP/IP)
- The format of the video (e.g. MPEG video)

For an IP multicast session also the multicast address of the media and the transport port for the contact address must be provided.

4.7 Summary

SIP is a signaling protocol that can be used to establish many kinds of sessions. It is a text-based protocol similar to HTTP. SIP addresses have the form of a URL (e.g. *sip:user@host*).

The following SIP components are defined in the IETF specification: A user agent is an end-device which initiates calls and waits for incoming calls. A proxy server is an intermediary program that forwards SIP messages. A redirect server maps the address of a SIP request into zero or more new addresses and returns them. A registrar accepts *Register* messages and is typically co-located with a proxy or redirect server.

A registration is a SIP transaction where a user agent registers at a SIP registrar to bind the user's permanent SIP URL to a current location. To establish a call, the caller sends an *Invite* message to the callee. An invitation can be executed either in proxy or redirect mode.

Data in the body of an *Invite* message is often encoded using SDP. This data provides information of the multimedia capabilities of the caller.

To enable SIP nodes to request event notifications from remote nodes, the IETF has specified an abstract framework. This framework defines the *Subscribe* and *Notify* messages.

Chapter 5

Multimedia Transport over IP Networks

Chapter 3 (H.323) and chapter 4 (SIP) focused on the initiation, termination and the control (i.e. the signaling) of multimedia sessions. Both, H.323 and SIP use the Real-Time Transport Protocol (RTP) and the Real-Time Transport Control Protocol (RTCP) to transmit and control the media stream, respectively.

Real-time data has requirements different to non-time-critical transmissions. For example they do not need a highly reliable packet transport. [Schulzrinne, 2000] defines following requirements of multimedia transport over IP networks:

Sequencing: If packets are received out of order they must be reordered in real-time. Lost packets have to be detected and are not requested again.

Intra-media synchronization: It is important that the time between two packets at the receiver is the same as at the sender. Therefore, the receiver has to be informed about the amount of time that should elapse between two frames. Usually, the packets are played out at a fixed interval. An exception are silence periods, were no packets are sent.

Inter-media synchronization: Often different media types like audio and video are used. It is necessary to synchronize them so that the audio that is played out matches the video (lip-sync).

Payload identification: It is often necessary to change the encoding for the media on the fly. For example this is the case when the available bandwidth has changed). Then a mechanism is needed to identify the encoding of each packet.

Frame indication: Multimedia data are sent in logical units called frames. It must be guaranteed that it can be determined where the frame begins and ends.

RTP and RTCP fulfill these requirements. Together they also provide functionality beyond sequencing and loss detection [Schulzrinne, 2000]:

Multicast-friendly: RTP and RTCP have been designed to operate in small multicast groups like a three-person phone call as well as in huge ones like multimedia broadcast events.

Media Independence: RTP provides services needed for real-time media like voice or video. For every codec a separate specification with the additional header fields and the semantics has to be defined.

Mixers and Translators: A mixer is a system that receives RTP packets from one or more sources, combines these packets to one RTP stream and sends it out. A translator takes a single media stream and converts it to another format. Then the new stream is sent away.

Quality of Service (QoS) Feedback: RTCP allows the receiver of a media stream to provide feedback on the quality of the reception. RTP sources can use this information to adjust the data rate.

Loose Session Control: The Real-Time Transport Control Protocol enables users to distribute identification information periodically. This enables other users to see who is participating in a session. Identification information can be for example the name, a phone number or the email address.

Encryption: RTP media streams can be encrypted using keys that are exchanged by (non-RTP) methods.

5.1 Real-Time Transport Protocol

RTP provides end-to-end network transport functions suitable for applications that transmit real-time data, such as audio, video or simulation data, over multicast or unicast network services [Schulzrinne et al., 1996]. Although RTP can run over any suitable transport or network protocol, it usually runs on top of UTP and uses its multiplexing and checksum services.

5.1.1 RTP Header Format

Figure 5.1 shows the RTP header format with an optional payload. The RTP header is 12 bytes long. The length of the payload is just limited by the underlying protocol, not by RTP itself. The following description of the RTP header fields was taken from [Schulzrinne et al., 1996].

Version (V): Indicates the version of the protocol. The current version defined in [Schulzrinne et al., 1996] is two.

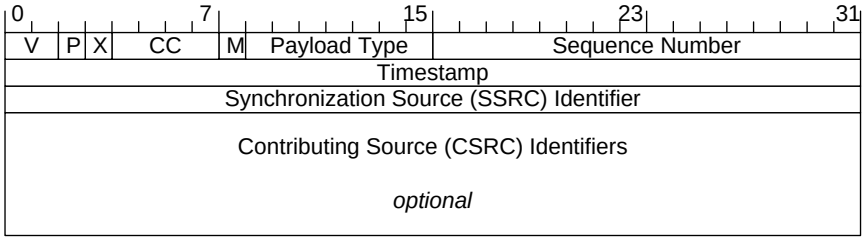


Figure 5.1: RTP header format (from [Schulzrinne et al., 1996])

Padding (P): If this bit is set, the packet contains one or more additional padding octets at the end which are not part of the payload. The last octet of the padding contains a count of how many padding octets should be ignored. Padding may be needed by some encryption algorithms with fixed block sizes or for carrying several RTP packets in a lower-layer protocol data unit.

Extension (X): If the extension bit is set, the fixed header is followed by exactly one header extension. The format of this extension is defined in section 5.1.2.

CSRC Count (CC): The Contributing Source (CSRC) count contains the number of CSRC identifiers that follow the fixed header.

Marker (M): The interpretation of the marker bit is defined by a profile. It is intended to allow significant events such as frame boundaries to be marked in the packet stream. A profile may define additional marker bits or specify that there is no marker bit by changing the number of bits in the payload type field.

Payload Type: This field identifies the format of the RTP payload and determines its interpretation by the application. A profile specifies a default static mapping of payload type codes to payload formats. Additional payload type codes may be defined dynamically through non-RTP means. An RTP sender emits a single RTP payload type at any given time; this field is not intended for multiplexing separate media streams.

Sequence Number: The sequence number increments by one for each RTP data packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence. The initial value of the sequence number is random (unpredictable) to make known-plaintext attacks on encryption more difficult, even if the source itself does not encrypt, because the packets may flow through a translator that does.

Timestamp: The timestamp reflects the sampling instant of the first octet in the RTP data packet. The sampling instant must be derived from a clock that increments monotonically and linearly in time to allow synchronization and

jitter calculations. The resolution of the clock must be sufficient for the desired synchronization accuracy and for measuring packet arrival jitter (one tick per video frame is typically not sufficient). The clock frequency is dependent on the format of data carried as payload and is specified statically in the profile or payload format specification that defines the format, or may be specified dynamically for payload formats defined through non-RTP means. If RTP packets are generated periodically, the nominal sampling instant as determined from the sampling clock is to be used, not a reading of the system clock. As an example, for fixed-rate audio the timestamp clock would likely increment by one for each sampling period. If an audio application reads blocks covering 160 sampling periods from the input device, the timestamp would be increased by 160 for each such block, regardless of whether the block is transmitted in a packet or dropped as silent.

The initial value of the timestamp is random, as for the sequence number. Several consecutive RTP packets may have equal timestamps if they are (logically) generated at once, e.g., belong to the same video frame. Consecutive RTP packets may contain timestamps that are not monotonic if the data is not transmitted in the order it was sampled, as in the case of MPEG interpolated video frames. (The sequence numbers of the packets as transmitted will still be monotonic.)

Synchronization Source (SSRC) Identifier: The SSRC field identifies the synchronization source. This identifier is chosen randomly, with the intent that no two synchronization sources within the same RTP session will have the same SSRC identifier. Although the probability of multiple sources choosing the same identifier is low, all RTP implementations must be prepared to detect and resolve collisions. If a source changes its source transport address, it must also choose a new SSRC identifier to avoid being interpreted as a looped source.

Contributing Source (CSRC) Identifier List: The CSRC list identifies the contributing sources for the payload contained in this packet. The number of identifiers is given by the CC field. If there are more than 15 contributing sources, only 15 may be identified. CSRC identifiers are inserted by mixers, using the SSRC identifiers of contributing sources. For example, for audio packets the SSRC identifiers of all sources that were mixed together to create a packet, are listed, allowing correct talker indication at the receiver.

5.1.2 RTP Header Extension

If the extension bit in the RTP header is set, a variable-length header extension follows the basic RTP header. The header extension enables individual implementations to create payload-format-independent functions that require additional information. Figure 5.2 shows the RTP header extension format.

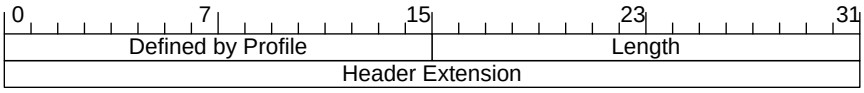


Figure 5.2: RTP header extension format (from [Schulzrinne et al., 1996])

The first two octets of the header are left open for distinguishing identifiers or parameters. The format of this field has to be defined by the profile specification under which the implementations are operating. The next two octets contain the number of 32-bit words in the extension (excluding the four-octet extension header).

5.2 Real-Time Transport Control Protocol

RTCP is based on the periodic transmission of control packets to all participants of a session. These packets are using the same distribution mechanism as the data packets. Following four functions are performed by RTCP [Schulzrinne et al., 1996]:

Provide Quality Feedback: The primary function is to provide feedback on the quality of the data distribution. The feedback may be directly useful for control of adaptive encodings, but experiments with IP multicasting have shown that it is also critical to get feedback from the receivers to diagnose faults in the distribution. Sending reception feedback reports to all participants allows one who is observing problems to evaluate whether those problems are local or global. With a distribution mechanism like IP multicast, it is also possible for an entity such as a network service provider who is not otherwise involved in the session to receive the feedback information and act as a third-party monitor to diagnose network problems. This feedback function is performed by the RTCP sender and receiver reports, described later in this chapter.

Carrying Transport-level Identifier: RTCP carries a persistent transport-level identifier for an RTP source called the *canonical name* or *CNAME*. Since the SSRC identifier (see section 5.1.1) may change if a conflict is discovered or a program is restarted, receivers require the CNAME to keep track of each participant. Receivers also require the CNAME to associate multiple data streams from a given participant in a set of related RTP sessions, for example to synchronize audio and video.

Control Rate of Packets: The first two functions require that all participants send RTCP packets, therefore the rate must be controlled in order for RTP to scale up to a large number of participants. By having each participant send its control packets to all the others, each can independently observe the number of

participants. This number is used to calculate the rate at which the packets are sent.

Control Session: A fourth, optional function is to convey minimal session control information, for example participant identification to be displayed in the user interface. This is most likely to be useful in loosely controlled sessions where participants enter and leave without membership control or parameter negotiation. RTCP serves as a convenient channel to reach all the participants, but it is not necessarily expected to support all the control communication requirements of an application.

5.2.1 RTCP Packets

Each RTCP packet begins with a fixed part (similar to that of RTP), followed by structured elements that may be of variable length. The length depends on the packet type, but it always ends on a 32-bit boundary. RTCP packets are stackable. This means, that multiple packets can be concatenated without any separators. This compound packet can be sent within one packet of the underlying protocol. The specification of RTCP [Schulzrinne et al., 1996] defines five RTCP packet types:

Sender Reports: Users who are sending media generate sender reports. A sender report consists of a header, the sender information (timestamps, data sent so far) and report blocks. A report block contains statistics on the reception of RTP packets from a single synchronization source.

Receiver Reports: The receiver report is created by users who receive media. Each report contains one block for each RTP source.

Source Description Items (SDES): This packet type is used for session control. It contains the CNAME (see section 5.2), which is used to associate different media streams generated by the same source and for resolving conflicts in the *SSRC* value. *SDES* packets also identify the participant through its name, email and phone number.

BYE: A *BYE* packet is included when a user leaves a session.

APP: This type of packet can be used to add application-specific information to RTCP packets.

5.3 Summary

Protocols that transmit data over packet-based networks in real-time, have requirements different to non-real-time protocols.

In the majority of cases, a highly reliable packet transport is not needed. Lost packets are not requested again. Out-of-Order packets must be re-ordered in real-time. It is also important that at the receiver and at the sender the time between

two packets is the same. If different types of media are transferred, they must be synchronized correctly. Other requirements for real-time protocols are frame indication and the possibility to change the encoding of the media on the fly.

RTP and RTCP fulfill these requirements. RTP is used to transport multimedia data over an IP network, while RTCP provides feedback of the quality of the transmission and conveys minimal session control information.

Chapter 6

Presence and Instant Messaging Systems

A presence and instant messaging system enables users to send each other short textual messages und receive the online status of a subscribed user. Also file sharing and chatting with friends is supported by most such systems.

Instant messaging and presence are two independent ideas. They do not have to be implemented in one system. For example the UNIX tool *talk* is an instant messaging system without a presence service. When talking about instant messaging, presence capabilities are often presumed. Therefore, and because this thesis deals exclusively with instant messaging systems that are coupled with presence, the terms *Instant Messaging System* and *Presence and Instant Messaging System* are herein interchangeable.

The first presence and instant messaging system was ICQ ("I Seek You")[1]. It was developed in 1996 by a small Israeli company called Mirabilis. The popularity of this software rose enormous and in the year 2000, ICQ was requested more than 100 million times from the server of download.com[2]. Encouraged by the success of ICQ also other companies developed instant messaging systems. The most acquainted ones beside ICQ are the Microsoft Messenger[3], the AOL Instant Messenger (AIM)[4] and the Yahoo! Messenger[5]. In 1998 Mirabilis was bought by AOL (now AOL Time Warner)[6].

Because all the different instant messaging clients use a different protocol for the communication with the server, interconnection between different systems is nearly impossible. Therefore many users have to install and run several instant messaging clients on their computer. This situation caused Jeremie Miller to orig-

[1] http://www.icq.com
[2] see http://download.cnet.com/downloads/0-10000-7-2026450.html
[3] http://messenger.msn.com
[4] http://www.aim.com
[5] http://messenger.yahoo.com
[6] http://www.aoltimewarner.com

inate an open source project called Jabber[7]. Jabber should be able to interconnect to other messaging systems via so called transports. Since the introduction of Jabber several transports have been developed. Jabber will be explained in detail in section 8.1.

Also the IETF recognized the problem of the competing protocols. A working group developed the SIMPLE protocol which should be a standard for next generation instant messaging services. Among other companies, also the market leaders AOL Time Warner and Microsoft[8] announced to support this new protocol. The SIMPLE protocol will be explained in section 6.2.

6.1 A Model for Presence and Instant Messaging Systems

[Day et al., 2000] describes an abstract model for presence and instant messaging systems. Its purpose is to specify terms, entities and services. This section explains the ideas behind the abstract model. Section 8.1 describes Jabber as a concrete instant messaging system.

The abstract model in [Day et al., 2000] defines two services: a presence and an instant message service.

6.1.1 Presence Service

The presence service has two distinct sets of clients it has to collaborate with. While *presentities* provide presence information to the server, *watchers* receive this kind of information in form of notifications (see Figure 6.1). However, in an implementation of an instant messaging system a client usually realizes both, presentities and watchers.

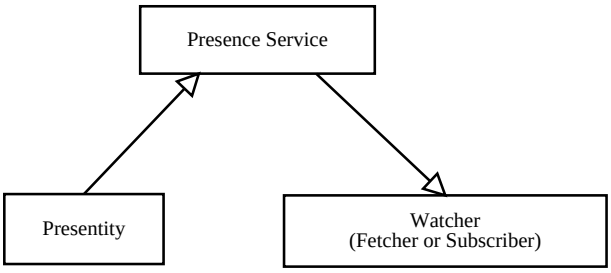


Figure 6.1: Overview of a Presence Service

When the presence service receives a presence status from a presentity it first stores the information. Then it determines the watchers where the status has to be distributed. There are two kinds of watchers: A *fetcher* simply wants to know the current presence status of a presentity while a *subscriber* has a subscription to

[7] http://www.jabber.org
[8] http://www.microsoft.com

the status information. If a fetcher regularly requests the status of a presentity it is called *poller*. A protocol that defines the interaction between presence service, presentity and watcher is called *presence protocol*.

RFC 2778 [Day et al., 2000] suggests following format for presence notifications: A notification consists of an arbitrary number of elements, called presence tuples. Each such tuple has a status marker (states the presence status of the user like online, offline, away or do not disturb), an optional communication address and other presence markup (also optional). The communication address may consist of a communication means and a contact address.

6.1.2 Instant Message Service

Like the presence service also the instant message service has two distinct sets of clients: A *sender* which transmits the created message to the service and an *instant inbox* which is the specified destination of the message (see Figure 6.2). The service gets the message from the sender and delivers it to the appropriate instant inbox. A protocol that defines the interaction between instant message service, sender and instant inbox is called instant message protocol.

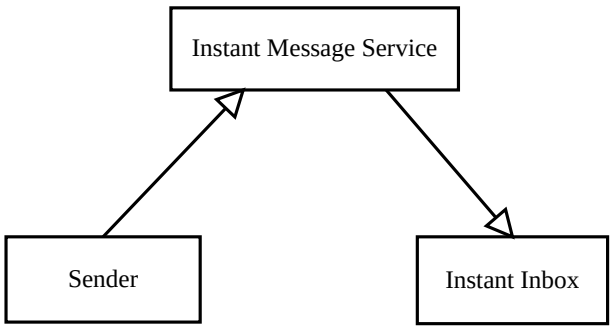


Figure 6.2: Overview of an Instant Message Service

6.2 The SIMPLE Protocol

SIMPLE is an abbreviation of *SIP for Instant Messaging and Presence Leveraging Extensions*. The SIMPLE protocol is based on SIP (see chapter 4) and extends it by additionally defining one object and one event package. The specification consists of two documents: one for presence [Rosenberg et al., 2002] and one for instant messaging [Campbell and Rosenberg, 2002]. Note that both documents are drafts and can be changed regularly.

6.2.1 SIP Extensions for Presence

SIP is very convenient for being extended for presence, because SIP registrars and location services already know, if and where a user is registered. SIP Extensions

for Presence [Rosenberg et al., 2002] is an IETF document that describes an event package within the general SIP event notification framework (see section 4.5). It makes use of the *Subscribe* and *Notify* methods defined there. The event package is compliant with the Common Presence and Instant Messaging (CPIM) framework defined in [Crocker et al., 2002], [Atkins and Klyne, 2002] and [Sugano et al., 2002] and therefore it is able to interwork with other presence systems compliant to CPIM.

Definitions

Additionally to the terms defined in [Day et al., 2000] (see section 6.1) following terms are defined [Rosenberg et al., 2002, p .5]:

Presence User Agent (PUA): A presence user agent manipulates presence information for a presentity. This manipulation can be the side effect of some other action (such as sending a SIP *Register* request to add a new contact) or can be done explicitly through the publication of presence documents.

Presence Agent (PA): A presence agent is a SIP user agent which is capable of receiving *Subscribe* requests, responding to them, and generating notifications of changes in presence state. A PA must have knowledge of the presence state of a presentity.

Presence Server: A presence server is a physical entity that can act as either a presence agent or as a proxy server for *Subscribe* requests. When acting as a PA, it is aware of the presence information of the presentity through some protocol means. When acting as a proxy, the *Subscribe* requests are proxied to another entity that may act as a PA.

Edge Presence Server: An edge presence server is a PA that is co-located with a PUA. It is aware of the presence information of the presentity because it is co- located with the entity that manipulates this presence information.

Note that the specification explicitly allows multiple PUAs per presentity. This enables a user to have many devices like cell phones or Personal Digital Assistants (PDAs). Each of these devices contributes to the overall presence information for a presentity. PUAs do not receive *Subscribe* or send *Notify* messages.

Overview of Operation

To subscribe the presence information from another user, the subscriber creates a *Subscribe* request. This request contains a Uniform Resource Identifier (URI) to identify the presentity, the URI of the subscriber, the *Call-ID* and other fields known from SIP messages. Figure 6.3 shows an example of a *Subscribe* request.

This request is handled as any other SIP request. When the message arrives at a presence server, this server can either act as a presence agent for the presentity and

```
SUBSCRIBE sip:resource@example.com SIP/2.0
Via: SIP/2.0/UDP watcherhost.example.com;branch=z9hG4bKnashds7
To: <sip:resource@example.com>
From: <sip:user@example.com>;tag=xfg9
Call-ID: 2010@watcherhost.example.com
CSeq: 17766 SUBSCRIBE
Max-Forwards: 70
Event: presence
Accept: application/cpim-pidf+xml
Contact: <sip:user@watcherhost.example.com>
Expires: 600
Content-Length: 0
```

Figure 6.3: Example of a *Subscribe* request (from [Rosenberg et al., 2002])

terminate the subscription, or proxy it on to an Edge Presence Server. The decision whether to terminate or to proxy the subscription is a local matter.

After the authentication and authorization of the subscription at the PA (whether in the presence server or edge presence server), a response is returned. This response can either be a *200 OK* response if the subscription was authorized or a *202* response if the authorization is pending. In both cases, the PA immediately sends a *Notify* message with the state of the presentity to the subscriber. If the authorization is pending the state of the presentity indicates *offline*. This is for security reasons to protect the privacy of the presentity. Figure 6.4 shows an example of a response when the subscription was authorized.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP watcherhost.example.com;branch=z9hG4bKnashds7;
received=192.0.2.1
To: <sip:resource@example.com>;tag=ffd2
From: <sip:user@example.com>;tag=xfg9
Call-ID: 2010@watcherhost.example.com
CSeq: 17766 SUBSCRIBE
Event: presence
Expires: 600
Contact: <sip:server.example.com>
Content-Length: 0
```

Figure 6.4: Example of a response when the subscription was authorized (from [Rosenberg et al., 2002])

Always when the state of the presentity changes, the PA sends *Notify* messages with the current state to all authorized subscribers (see Figure 6.5). The body of a *Notify* message describes the state of the presentity. The format of the body is

declared in the *Content-Type* field of the message. In Figure 6.5, the body of the message has the Presence Information Data Format (PIDF), which is specified in [Atkins and Klyne, 2002].

```
NOTIFY sip:resource@example.com SIP/2.0
Via: SIP/2.0/UDP watcherhost.example.com;branch=z9hG4bKnashds7
From: <sip:resource@example.com>;tag=ffd2
To: <sip:user@example.com>;tag=xfg9
Call-ID: 2010@watcherhost.example.com
Event: presence
Subscription-State: active;expires=599 Max-Forwards: 70
CSeq: 8775 NOTIFY
Content-Type: application/cpim-pidf+xml
Content-Length: ...

PIDF Document
```

Figure 6.5: Example of a *Notify* message (from [Rosenberg et al., 2002])

Examples when the presence of a user can change are [Rosenberg et al., 2002]:

- Turning on and off a cell phone
- Modifying the registration from a softphone
- Changing the status on an instant messaging tool

To terminate a subscription, the subscriber sends a *Subscribe* message with an *Expires* header of zero. If the presentity wants to terminate a subscription, the presence agent sends a *Notify* message with a *Subscription-State* header indicating that the subscription has been terminated.

6.2.2 SIP Extensions for Instant Messaging

The SIP Extensions for Instant Messaging are specified in [Campbell and Rosenberg, 2002]. Sending an instant message using the SIMPLE protocol is very easy. The sender creates a *Message* request containing the URL of the destination and the message as body. Figure 6.6 shows an example of a *Message* request.

Note that apart from *im:* URLs, also normal SIP URLs can be used and the body of the message can be of any MIME type. When the message has reached the receiver and the receiver has accepted the message, a *200 OK* response is generated.

6.3 Integrated Communication Platforms

Most instant messaging systems available today already offer the possibility to make phone or video calls and also chatting with friends is often included.

```
MESSAGE im:user2@domain.com SIP/2.0
Via: SIP/2.0/UDP user1pc.domain.com
From: im:user1@domain.com
To: im:user2@domain.com
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Content-Type: text/plain
Content-Length: 18
Watson, come here.
```

Figure 6.6: Example of a *Message* request (from [Campbell and Rosenberg, 2002])

Because instant messaging includes a presence service it can easily be used as an integrated communication platform. The presence service can help a user to decide which kind of communication service is the best to reach another person. If, for example, the presence of a user states that he or she is not at the office desk, the preferred service would be an IP telephony call to the user's mobile phone.

An intelligent instant messaging service could also offer the possibility to define how incoming messages must be handled. The server then routes the messages according to the user's preferences. This kind of communication platform is like a unified messaging system where many kinds of messages (text, voice, video) are stored on the server. The next chapter explains unified messaging systems in detail.

6.4 Summary

Presence and instant messaging systems have become very popular over the past years and the number of people who use ICQ, AIM or the Microsoft Messenger has risen enormously.

The task of a presence service is to notify subscribed users when the online status of the presentity has changed. An instant messaging service enables a user to send short textual messages to other users.

The SIMPLE protocol specification describes a presence and instant messaging protocol based on SIP. It uses an event package within the general SIP event notification framework to support presence capabilities and defines a new SIP message to enable instant messaging.

A presence and instant messaging system can be used as an integrated communication platform. The presence service can help a user to decide which kind of communication service is the best to reach another user. An intelligent instant messaging server can also route messages according to the user's preferences.

Chapter 7

Unified Messaging

When talking about IP telephony, we mean the convergence of telephony and data networking over a common infrastructure. This chapter deals with another kind of convergence: the ability to access many types of messages (e.g. email, voice mail, fax and perhaps instant messages) with different types of devices (e.g. mobile phones, PC, fax machine). Examples would be the possibility to access voice mail messages from the PC and email messages from the mobile phone. This form of convergence is called unified messaging.
[Lotus Software, 2002] defines unified messaging as:

> *An advanced message management solution for all media types, providing access to any message, anytime, anywhere, from any device.*

Chapter 6.3 introduced the term *integrated messaging*. In contrast to unified messaging, with integrated messaging the system administrators still have to maintain two separate messaging infrastructures. However, the capabilities of integrated messaging to the end user are very similar to that of unified messaging.

This chapter first discusses the benefits of unified messaging. Then, the architecture of a Unified Messaging System (UMS) and how it is integrated into a network environment, is explained. Also the integration of a UMS in a SIP-based network is described. Finally, methods to send voice mails and faxes over the Internet is specified.

7.1 Benefits

This section states the benefits of unified messaging. Also examples of the usage of a UMS are presented.

The main advantage for the user of a UMS is flexibility. Unified messaging breaks down the terminal and media barriers. People using different technologies, different media and different terminals can still communicate to anyone, anywhere, at any time [IEC, 2002].

In other words, it allows the user to choose any mechanism to access their messages. This can be done with network clients, web clients or telephones (see Figure 7.1).

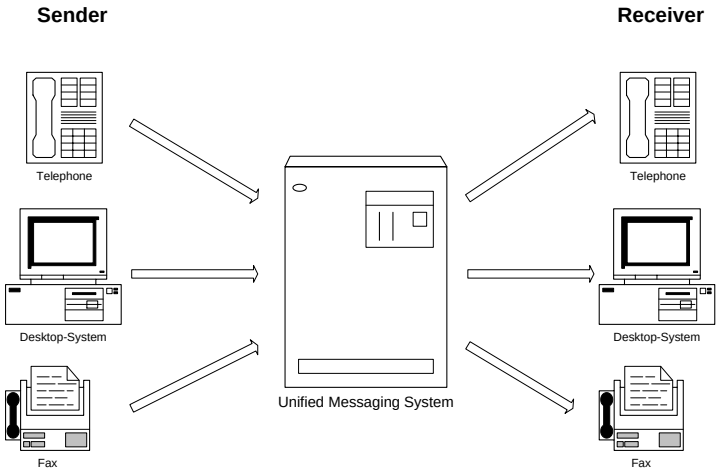


Figure 7.1: Unified Messaging System

Imagine following situation: A user stays at a hotel without Internet connection. When checking his voice mail, he is told that he has four new emails and a new fax. The user is also informed about the originators of the messages. Now he can decide whether he wants to have them read over the phone, to ignore a message or to forward a message to the fax machine of the hotel. When the user wants to reply to a message, he simply sends a spoken response back to the originators UMS.

Another reason for the acceptance of unified messaging is that users have a single point of access to their messages. They have one interface to check their unified mailbox for incoming voice, fax or email messages. This saves time and increases the productivity of the user.

A UMS allows people to control when and how they can be reached. Many of these systems offer rules for processing, delegating, archiving and notificating. With the rules the user can define the times he wants to be notified about incoming messages (for example via Short Message Service (SMS)) and the times he does not want to be disturbed. They make it also possible, to handle messages from defined originators in a different way.

Unified messaging saves the time (and therefore the costs) of network administrators, too. Instead of maintaining multiple, independent messaging servers, a UMS offers a unified administration and management interface.

7.2 Technical Overview

This section explains how a UMS can be integrated into a network environment. Note that there is no standardized way to realize this. A concrete integration depends on the requirements for the unified messaging solution.

First, the difference between a network with multiple messaging servers and a network with a UMS is shown. Then, the integration of unified messaging into a SIP environment is described.

7.2.1 A Sample Architecture

In most organizations there are two separate networks: The fax and voice mail servers are connected to the PSTN via a PBX while the email server is connected to the Internet (see Figure 7.2)

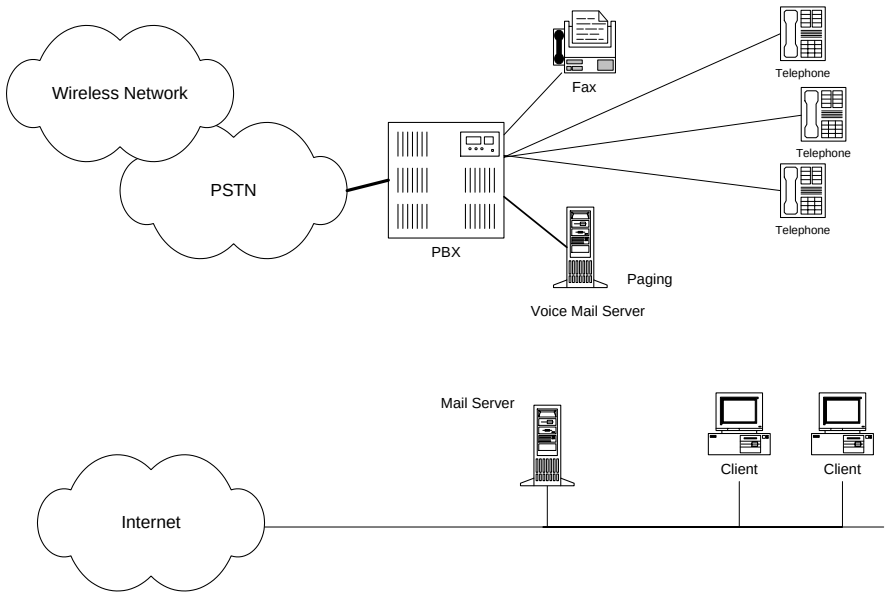


Figure 7.2: Conventional Network Environment (from [Lotus Software, 2002])

To exchange messages across the borders of the separate networks, a connection between these two worlds must exist (see Figure 7.2). Four devices are important for this unified messaging solution:

- The UMS with an integrated email server manages all types of messages and stores them in a single message store.
- The Telephony User Interface (TUI) implements the voice mail *auto attendant* function. It takes voice messages and transfers them to the UMS. It also enables a user to access all stored messages with the telephone.

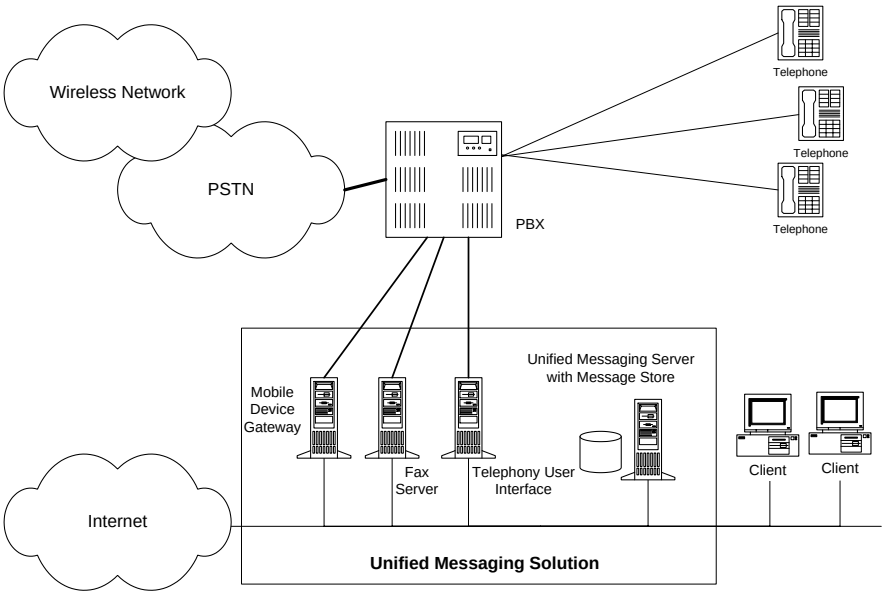


Figure 7.3: Unified Messaging Solution (from [Lotus Software, 2002])

- The Fax Server receives faxes and sends them to the UMS.
- The Mobile Device Gateway provides wireless notification, access and synchronization functionality, using either the Internet or the PSTN to connect to the wireless network.

Note that the TUI, the Fax Server and the Mobile Device Gateway are just logical units. They may also run on only one physical device.

7.2.2 Unified Messaging using SIP

This section describes the design of a voice mail system for IP telephony [Singh and Schulzrinne, 2000]. It is based on SIP (see chapter 4), because SIP has an open architecture and the protocols are extensible. The stored messages should be accessible from any Internet connected device (e.g. with a standard media player).

As protocol for delivering the streaming media the Real-Time Streaming Protocol (RTSP) [Schulzrinne et al., 1998] is chosen. To access the messages from a PSTN phone, a SIP-PSTN gateway must be used.

The presented multimedia mail system describes a voice mail server, an RTSP media server, and possible a SIP proxy server. The voice mail server is a SIP interface to the media server. It allows connections from a SIP user agent. The RTSP media server handles the recording and the play back of the messages.

Figure 7.4 shows the forwarding of a call to the voice mail server and storing it at the RTSP media server's database. The SIP proxy server handles all users of

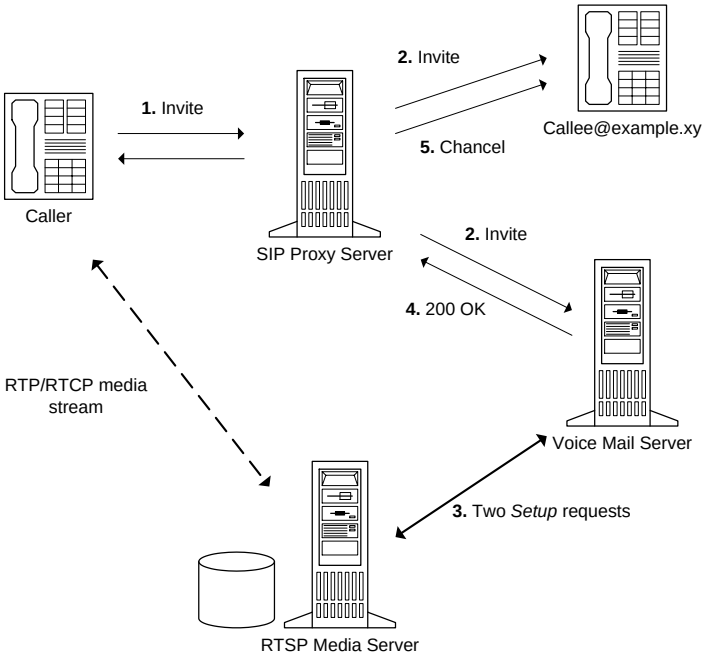


Figure 7.4: Unified Messaging using SIP (from [Singh and Schulzrinne, 2000])

the *example.xy* domain. Users register at this server to give information about their current location. The voice mail server registers its location, too. From the SIP proxy server's point of view, there are two current locations for every user. One is the registered location of the user and one is the voice server.

When User A calls User B, the call is proxied to both locations. When User B accepts the call, the call to the voice server is cancelled. When User B does not pick up the phone, the voice mail server accepts the call after a defined time. Before accepting the call, the voice mail server sets up the media path with the RTSP media server. An RTSP *Setup* message tells the RTSP server to play back the voice prompt for leaving a message. Then, again a *Setup* message is sent to start the recording. Finally, when the caller hangs up (and generates a SIP *Bye* message), the voice mail server informs the RTSP media server to stop the recording. To retrieve a voice mail, a user has several choices:

- With an RTSP media player, the user can directly play the voice message. This is done by stating the URI of the audio file at the RTSP server.
- The user can use his SIP phone to contact the SIP voice server. The voice server then tells the RTSP server to send the message to the users SIP phone.
- The voice mail message can also automatically be sent to an email address or accessed via a web page.

7.3 Message Transport over the Internet

When voice mails or faxes are received, a mechanism is needed to transport them over a packet-based network. The IETF defines methods which use ESMTP (see section 1.1.2) as transport protocol and the Internet Mail Protocol Suite (see section 1.1.1) as message format specification.

7.3.1 Voice Mails

The Network Working Group of the IETF has created an Internet Draft called *Voice Profile for Internet Mail - version 2 (VPIM)* [Vaudreuil and Parsons, 2002], which specifies the message format and the transport mechanism of voice mails over the Internet.

As message format, MIME (see section 1.1.1) is used. MIME enables the encoding of binary data so that it can be transported over the 7-bit text-oriented ESMTP protocol. The encoding type must be stated in the *Content-Transfer-Encoding* MIME field. Two types are possible: *Quoted-Printable* was designed for text-based data, while *Base64* was designed for binary data. For audio data, the latter one is much more efficient.

The addresses in VPIM messages have a format as it is defined in [Resnick, 2001]. They consist of a local part (used for username or mailbox identification) and a host part (used for global machine identification).

[Vaudreuil and Parsons, 2002] defines a number of header fields which are permitted in VPIM messages. They are similar to the header fields of email messages. Additionally, following MIME audio content descriptions are specified:

Content-Description: Can state a textual description of the voice content.

Content-Disposition: This field is necessary to identify the body parts within a VPIM voice message. This is especially useful for multipart messages.

Content-Duration: Indicates the length of the audio stream in seconds.

Content-Language: This field specifies the spoken language of the audio body part. The encoding is defined in [Alvestrand, 2001].

A VPIM message may contain different content types, such as normal text, PCM-encoded audio or images (e.g. for faxes). For a detailed description of the possible content types see [Vaudreuil and Parsons, 2002].

7.3.2 Fax

For transferring faxes over the Internet the IETF has defined a standard [Toyoda et al., 1998] and an extension to this standard [Masinter and Wing, 1999] (describes additional features like the confirmation of delivery and processing). As transport

mechanism ESMTP was chosen. The header fields of a fax message must be compliant with [Resnick, 2001] and [Braden, 1989], which define the format of mail headers. Fax data is included in a MIME object. The data format of the fax image is based on the minimum set of Tag Image File Format (TIFF) for Facsimile [McIntyre et al., 1998].

7.4 Summary

Unified messaging breaks down the terminal and media barrier by enabling users to access their emails, voice mails and faxes with different kinds of devices. Therefore, a user is more flexible and saves time. This is because a UMS offers a single point of access to the user's messages. Also system administrators benefit from unified messaging. A UMS provides a unified administration and management interface for all supported types of media.

To integrate a voice recorder (as part of a UMS) in a SIP environment, a voice mail server, an RTSP media server and a SIP proxy server is needed. The voice mail server works as a SIP interface for the media server, which manages the stored messages.

To transport voice mails and faxes over the Internet, ESMTP is used as transport protocol and the Internet Mail Protocol Suite, which includes MIME, as message format specification.

Chapter 8

Practical Work

The task of the practical part of this thesis was to implement a client for an instant messaging system for the Infonova GmbH[1]. As instant messaging server Jabber[2] should be used. This open source server defines a protocol for the communication between client and server which is XML-based (see section 8.1.3). The main advantage of this server is that it can be extended, so that it can communicate with instant messaging servers of other companies as well. This is done via so called transports, which can be added to the core server. Before the client was implemented, the server had to be installed and configured on a Linux computer in the company.

Several requirements for the instant messaging client had to be fulfilled:

- The target platform for the client should be Microsoft Windows. C++ should be used as programming language together with the Microsoft Foundation Classes (MFC) as supporting libraries.
- The client must be able to maintain other users (buddies). It should be possible to group these users. Also the presence status of a buddy must be shown.
- The user must be able to define his own presence-status.
- The client must support several message-types. This includes conventional instant messages as well as chat-sessions.
- The client must support telephony connections to other users. This should be realized by connecting to the Open-H.323-based Infonova IP telephony client, which then carries out the VoIP-call.

The following sections describe the installation and configuration, the architecture and the protocol of the Jabber server. Also the usage and the structure of the instant messaging client is explained.

[1] http://www.infonova.at
[2] http://www.jabber.org

8.1 The Jabber Server

Jabber is an XML-based, open source instant messaging server and can be freely downloaded from the Internet [3]. The development of this project started in the year 1998 by Jeremie Miller.

Compared to other instant messaging servers, Jabber has some advantages. The biggest advantage is the ability of Jabber to communication with other instant messaging services. While users of normal instant messaging services can just communicate with users of the same service, Jabber provides so-called transports. These transports enable the communication to other services. They act as a proxy for the user and translate the protocols. Transports exist to ICQ, AOL Instant Messenger (AIM), MSN Instant Messenger and Yahoo! Messenger as well as IRC.

Another advantage of Jabber is that everybody can run his own server. This is especially interesting for companies which want to make their instant messaging inhouse communication independent from external services. Also security aspects should be concerned.

For running an own Jabber server, the license of the server is important. Jabber is covered under the Jabber Open Source License (JOSL). This license enables developers to design and write additional modules without making those modules open source. Modifications of the original source code must be contributed back to the Jabber community.

8.1.1 Installation and Configuration

The Jabber server was developed mainly on Linux but also was built and tested on FreeBSD, Solaris and other UNIX operating systems. After downloading, unpacking and building, the server has to be customized.

The customization is done by changing the original configuration files. They are XML-based and therefore human readable. The main configuration file is called *jabber.xml*. The basic configuration includes the declaration of the hostname. This is done by replacing *localhost* in the line

<host><jabberd:cmdline flag="h">localhost</jabberd:cmdline></host>

with the desired hostname.

If transports are used by the server they must be configured as well. After downloading and installing the transport (details can be found at the download-page of the appropriate transport) it must be decided if the service should run within the same process as the server or as a separate process. The latter method should be preferred because it is more scalable and the services can be started and stopped independently (without stopping the core server).

To configure the service, the file *jabber.xml* must be changed again. Additionally, when the service is running as separate process, an XML-file with the configuration of the service must be created (and declared in *jabber.xml*).

[3] http://jabberd.jabberstudio.org

Further information about the installation and configuration of the Jabber server can be found in [Saint-Andre, 2002].

8.1.2 Architecture

From a technical point of view, the differences between Jabber and other instant messaging systems are following [Saint-Andre, 2001]:

- XML foundation
- distributed network
- open protocol and codebase
- modular, extensible architecture

In contrast to some other instant messaging systems, Jabber has not a client-to-client architecture where messages may be directly sent from one client to another. Jabber uses a client-server architecture. All messages and other data go through the server. Of course, for application specific data transport the client is free to negotiate a direct connection to another client.

The network architecture of Jabber is similar to the email system (see section 1.1). Each client is connected to a local server. Every communication from or to this client passes this server. The Jabber identification of a user specifies the username as well as the local server of a user. For example

johann.madlberger@jabber.infonova.at

states, that the user *johann.madlberger* is known and registered at a Jabber server called *jabber.infonova.at*.

If a user wants to send a message to a user registered at another server, the local server gets the host name of the target from the identifier and relays the message to that server. Server-to-server connections run over port 5269, while client-to-client connections use port 5222. As underlying protocol TCP/IP is used.

The protocols for both kinds of connections are XML-based. This enables structured, intelligent messaging. Because XML is also a human readable format, the debugging of a client when an error has occurred is much easier as if using a binary format. A disadvantage of an XML-based protocol is the overhead of the data transfer.

One main demand when the Jabber server was designed, was to move the complexity from the client to the server. Therefore, the development of clients for Jabber is relatively easy. A Jabber client must:

- communicate to the Jabber server via TCP sockets
- parse and interpret well-formed XML packets
- understand message data types

There exist several libraries that support a developer with the handling of the protocol. So the developer can focus on the design of the user interface and the handling of user and operating system events.

Server Components

The Jabber server was designed to be very extensible and flexible. Therefore, the developers of the server use modules to handle a specific functionality like user authentication or data storage. This kind of modularity enables users to simply add new functionality like the support of another instant messaging protocol to the server.

The core Jabber server includes components that handle the following common tasks [Saint-Andre, 2001]:

- session management
- client-to-server communication
- server-to-server communication
- DNS resolution
- user authentication
- user registration
- database lookups
- storing messages for offline users
- storing and retrieving vCards
- filtering messages based on user preferences
- group chat (many-to-many communication)
- system logging

A Jabber server has four base components: accept, connect, exec and load. These components deserialize XML input for delivery to other base components. They also serialize XML for use by components that need data in this format. All these components interact with the deliver component. It directs deserialized XML from one base component to another one.

When writing an own component, it must be an accept, a connect, an exec or a load component. Often an accept component is the preferred type. This type of component runs as an own process. It connects to a Jabber daemon (jabberd) and authenticates itself as a certain id. Then the server will send any packets bound to this id to the authenticated component. A connect component is used, when the

author does not want to distribute it, but rather host it himself. Then the server connects and authenticates to such a component. The executed type of component is started by the jabberd process locally. The communication between component and server is done via the standard input and output. Finally, a loaded module is not globally usable but tightly bound to a server. It is a bit faster than the other types of components but it not as flexible as the other ones. More detailed information about writing an own component can be found in [Muldowney, 2001].

Transports

Transports are components that are often designed to communicate with services that use other protocols than Jabber does. Figure 8.1 shows the principle of a transport. A transport implements the proxy pattern, so their task is to convert messages from one protocol format to another one.

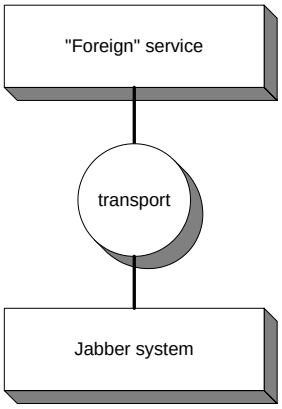


Figure 8.1: Jabber Transport (from [Saint-Andre, 2001])

A transport is registered at a server with a unique id. For example *icq.jabber.org* would be a valid id. Imagine a user sends a message to a friend with the id *bob@icq.jabber.org*, the server forwards the message to the appropriate transport. The transport converts the message to the foreign format and sends it to the ICQ instant messaging server.

When writing a transport to another instant messaging service, the most complicated part of the work is to write the conversion of the protocols. Many of the foreign protocols are not documented. Often they are changed regularly to prevent others to communicate with the server. Transports for Jabber can be downloaded from the Jabber homepage.

Basic Message Flow

To gain a deeper understanding of the Jabber server, message flow diagrams can help to see which components are involved when handling a message. This one

and the following two sections show diagrams of a basic message flow, the authentication of a user and the session manager.

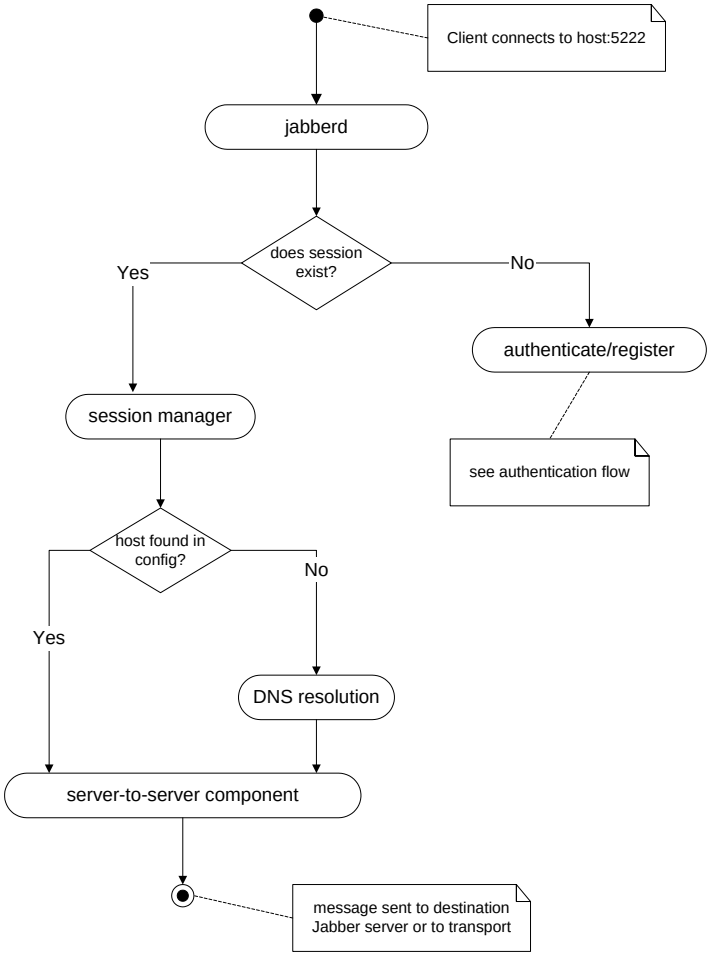


Figure 8.2: Basic Message Flow (from [Saint-Andre, 2001])

As can be seen in Figure 8.2, the Jabber server listens at TCP port 5222 for incoming connection requests. For secure socket layer (SSL) connections the server listens at port 5223. When such a request arrives, jabberd (the Jabber daemon) checks if a session for this user already exists. If not, the authentication or registration procedure starts (consult section 8.1.2 for details). If a session for this user exists, the message package is sent to the Jabber session manager.

The session manager compares the host name of the destination address with a list of addresses in the configuration file. If the host name is defined in the configuration file (e.g. the host name is linked to a registered transport) the message is relayed to the server-to-server component. If the host name is not defined in the

configuration file, the *dnsrv* component determines the IP address and the port of the host name. Then the message is sent to the server-to-server component.

Authentication

Before a registered user can send and receive messages it must be authenticated at a server. Therefore, the client connects to the server and initiates an XML stream. The server then checks the packet type and the namespace of the packet. If the packet type is *iq* (info/query), the subtype is *query* and the namespace is *jabber:iq:auth* the server knows that this packet contains authentication information.

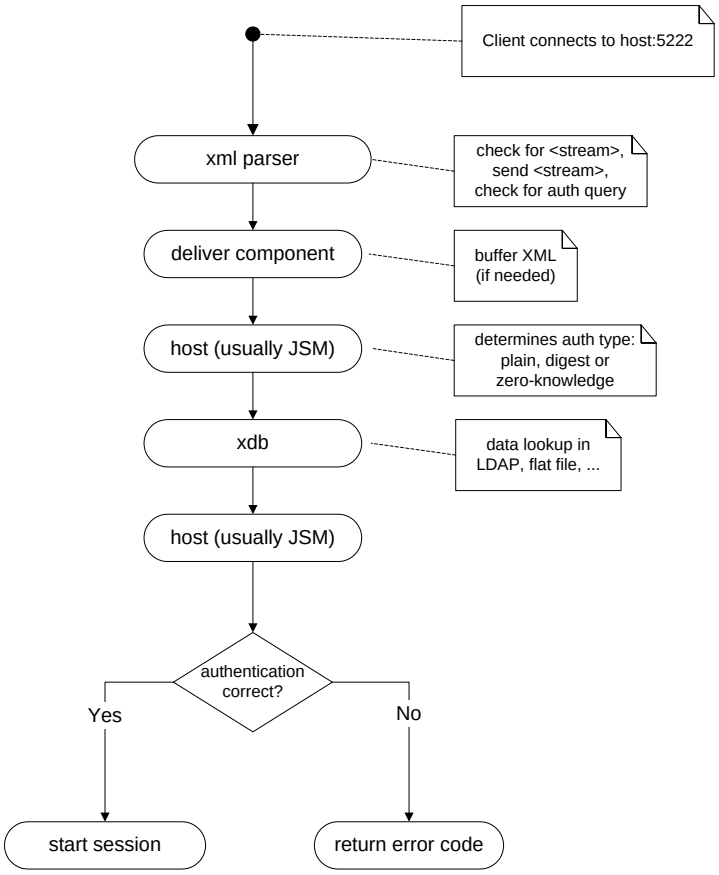


Figure 8.3: Authentication Flow (from [Saint-Andre, 2001])

The authentication information consists of a username, a resource and either a plain password, an encrypted password (SHA1) or appropriate data for the zero-knowledge authentication.

After the XML parser has analysed the package, the deliver component takes control of the XML stream and buffers the incoming data (if the client does not wait until the authentication has finished). Then the host (usually the Jabber session

manager) passes the authentication packet to the *xdb component* (XML Data Base component). This component relays the packet to the appropriate sub-component, which has registered for the type of authentication (e.g. plaintext-authentication component). User information with the correct authentication can be stored in a flat file or using a Lightweight Directory Access Protocol (LDAP) service.

Finally, the xdb component will return the result of the authentication procedure to the host (again, usually the Jabber session manager). Depending on the result either a new session for this user is started or an error message is returned. Figure 8.3 visualizes the flow of an authentication packet.

The Jabber Session Manager

The Jabber session manager is a central component in the architecture of the Jabber server. It handles messages, the presence status of a user as well as *info/query* packets. If an offline user receives a message, the Jabber session manager stores that message until the user is online again.

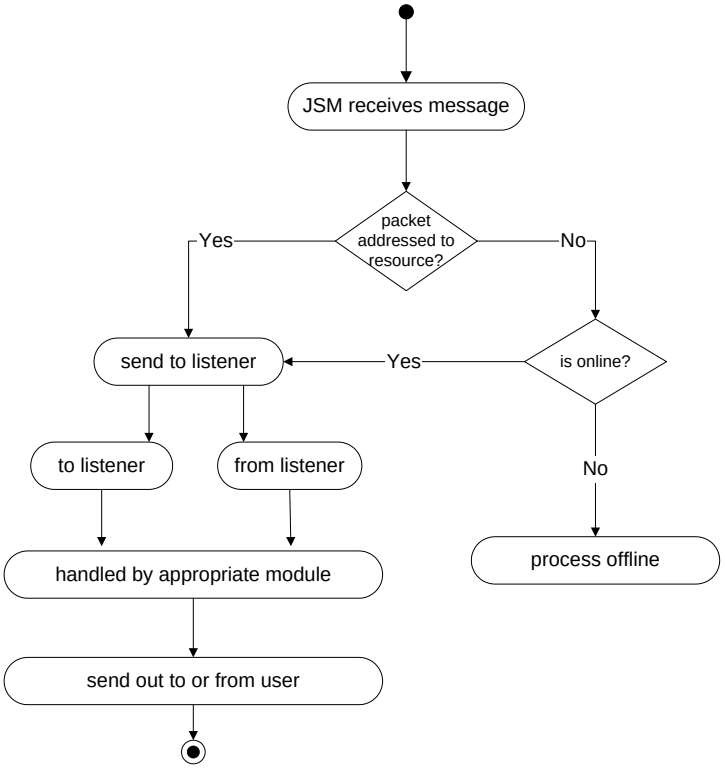


Figure 8.4: Jabber Session Manager (from [Saint-Andre, 2001])

Figure 8.4 shows the data flow of the session manager. When a packet is received, the resource of the target address is analysed. A resource states the location

from where the user is online. An example for a resource would be 'work', 'home' or 'laptop'. If no resource is stated, the session manager checks if an error has occurred and the user is online (online users should have always a resource defined). If the user is offline the message is stored, otherwise the session manager checks if the packet is intended to be delivered to the user or it is being sent from the user. Two listeners, one for 'to' messages and one for 'from' messages wait for packets and route them to the appropriate component. This component can filter and redirect messages, so that for example a user can define that messages from a specific friend should be forwarded to a given email account. After processing the packet it can either be routed again back to the listeners or sent out to or from the user.

To improve the performance of the Jabber session manager a thread pool is used. On server start-up, a defined number (as stated in the configuration file) of threads is created. When a new message arrives, the session manager takes an unused thread from the thread pool and binds it to the appropriate message port. A message port is a data structure that supports a client connection [Saint-Andre, 2001, p .17]. If there is no unused thread, the Jabber session manager can, but does not have to create a new thread.

8.1.3 The Jabber Instant Messaging Protocol

One of the main advantages of the Jabber instant messaging server is its open protocol. This protocol is XML-based and the specification can be freely downloaded from the Internet[4]. Excellent documents that describe the protocol and help implementing a Jabber client are [Muldowney and Landrum, 2000] and [Alfke, 2001]. In the year 2002 the protocol specification was submitted as Internet-draft to the IETF and was discussed at the IETF meeting in Yokohama, Japan in July 2002.

When creating an own Jabber client it is often not necessary to implement a Jabber protocol stack from scratch. Several libraries for different platforms exist that support the developer. Examples for such libraries are JabberCOM[5] for the Win32 platform and jabber.py[6] for Python developers. Other libraries can be found at the Jabber download page[7]. Note that many of these libraries are still in an early development stage and therefore do not support the complete Jabber protocol.

When a client connects to a Jabber server, it initiates an XML stream to the server. After the server received this stream it responses by initiating an XML stream, too. To quit a session the client closes the XML stream and waits for the end-tag of the server-to-client stream. Then the TCP/IP connection can be closed. Figure 8.5 shows the communication between a client and a server.

After establishing a connection and initiating the XML streams, the user must authenticate itself. Figure 8.6 shows the successful authentication of a user. Instead

[4] http://www.jabber.org/ietf/
[5] http://jabbercom.sourceforge.net
[6] http://jabberpy.sourceforge.net
[7] http://www.jabber.org/downloads.html

```
C: <?xml version='1.0' encoding='UTF-8' ?>
C: <stream:stream
C: to='jabber.infonova.at'
C: xmlns='jabber:client'
C: xmlns:stream='http://etherx.jabber.org/streams'>

S: <stream:stream
S: xmlns:stream='http://etherx.jabber.org/streams'>
S: id='34A453AD3'
S: xmlns='jabber:client'
S: from='jabber.infonova.at'
...
(authentication and rest of session)
...
C: </stream:stream>
S: </stream:stream>
```

Figure 8.5: Opening and closing an XML stream

of a plain password, the client could also send a digest (the hash result of the password using SHA1) to the server.

```
C: <iq type='set' id='ID'>
C: <query xmlns='jabber:iq:auth'>
C: <username>foo</username>
C: <password>bar</password>
C: <resource>home</resource>
C: </query>
C: </iq>

S: <iq type='result' id='ID'/>
```

Figure 8.6: Authentication of a user

Now the user is authenticated. The next step is to tell the Jabber server the presence or online status. This is done by sending a <*presence*> element. A user can either be available or unavailable. An unavailable user is seen by his buddies as offline. If the user is available the optional <*show*> element specifies the current status. This can either be away, extended away, do not disturb or free for chat. No <*show*> element indicates that the user is online.

When a user changes its presence the server relays this information to all others that have a subscription to the presence status of the user.

After authentication and sending the presence, the server delivers the buddy list (also called roster) to the client. This list contains Jabber entities with whom the

user has a presence relationship. When implementing a Jabber client, a main part is to display and manage the buddies of the user.

The Jabber protocol enables the user to do many things. He can subscribe and unsubscribe buddies, he can send messages to friends and chat with them. He can send files to them or simply read their electronic business card. For further information how to do that look at the detailed specification of the Jabber protocol.

8.2 The Instant Messaging Client

After describing the architecture of the Jabber instant messaging server this section explains the usage and structure of the client, whose implementation was an important part of this thesis.

First, the usage of the client is described in detail. Screenshots show the appearance of the program in different situations. Finally, an overview of the architecture is given.

8.2.1 User Manual

When starting the Infonova instant messaging client, an authentication dialog appears (see Figure 8.7). To begin a session, the user has to enter a valid username and a password. Also the domain name of the Jabber server and its port number should be stated.

Figure 8.7: Authentication dialog

The main window of the client consists of the buddy list, the system and transport message list and two menus underneath (see Figure 8.8). The users in the roster can be grouped. They are sorted alphabetically. The icon of each buddy signalizes its presence status. In the screenshot, Silvia is online, the friends Günther, Martin and Werner are offline, Jürgen is extended away and Christian does

not want to be disturbed. The maintenance of the groups and buddies can be either using the 'Manage' menu or by right-clicking the item in the roster.

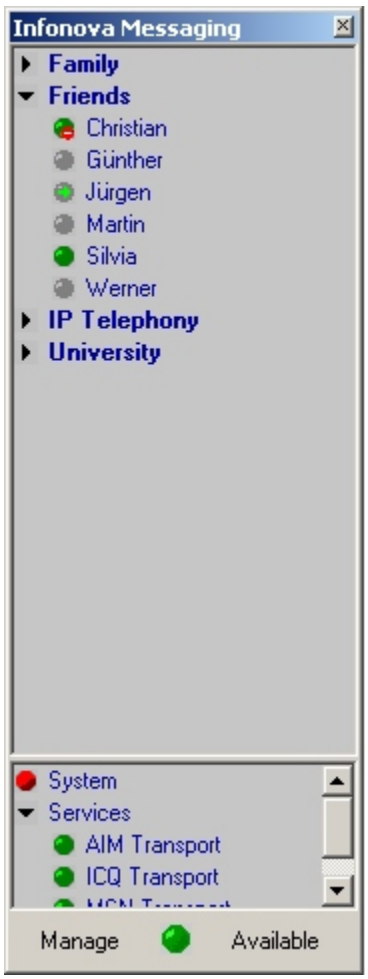


Figure 8.8: Infonova Instant Messaging Client

Following actions are possible for buddy items:

- Phone call
- Message
- Chat
- Info
- Remove

Note, that for a phone call the Infonova IP telephony client IPSylon[8] must be installed on the computer. The 'Info' menu item displays the electronic business card of the selected user. Within this dialog, the nickname of the buddy and the groups he belongs to can be changed. All other fields are read-only.

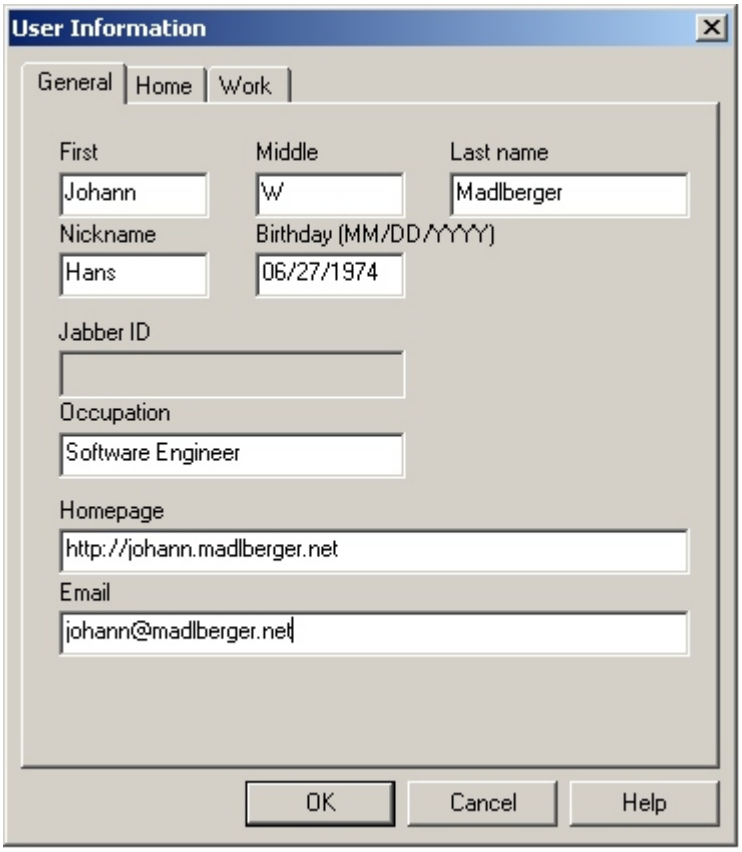


Figure 8.9: Business card

The user can define its own business card by selecting the item 'Edit vCard' in the 'Manage' menu. Figure 8.9 shows such a dialog. Other buddies can use for example the telephone numbers entered in the appropriate fields (in the 'Home' and 'Work' section) to give the user a call via the above mentioned IP telephony client. The list below shows the actions that are possible for group items:

- Add user
- Rename group
- Remove group
- Message to group

[8] http://ipsylon.infonova.at

When a user chooses the 'Add User' menu item, the dialog shown in Figure 8.10 appears. First the Message Server Type should be selected. This can either be Jabber Messaging Server (which is the default setting) or any transport that has been added to the server. Note that if a different transport than the default one is selected, some preconditions must be fulfilled. These preconditions are explained later. After entering the User ID (e.g. *johann.madlberger@jabber.infonova.at*) and the nickname, the groups the new buddy should belong to, can be selected. Finally, the reason why this buddy should allow the user to receive his presence, can be stated.

Figure 8.10: Add user dialog

If a buddy from a different instant messaging service should be added to the roster, some things have to be done first. It is important to note, that the user, who wants to add a buddy from another transport, has to be registered at this service. Jabber does not support the registration of users directly. This must be done by going to the homepage of the service provider. The account information of this service has to be delivered to the user's server. Therefore, the 'Manage' button has to be clicked and the menu item 'Services' has to be selected. Then a dialog shows the available transports and the user can choose one of them. Finally, the account information (e.g. username, password) has to be entered. When this is

done properly, the new transport is added as sub element of the 'Services' item in the system message list.

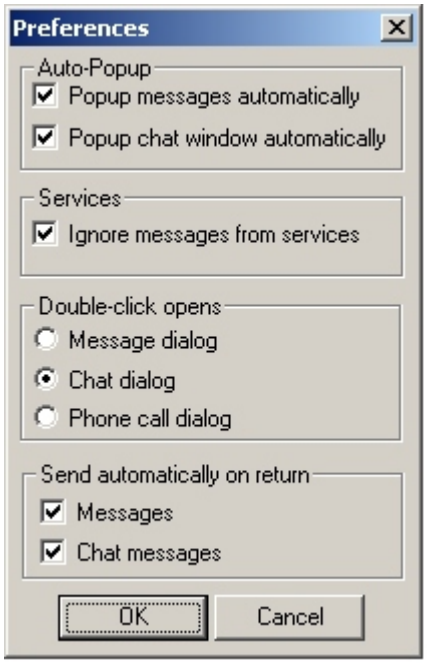


Figure 8.11: Preferences dialog

To customize the instant messaging client, the user has to open the 'Manage' menu and select 'Preferences'. Figure 8.11 shows the preferences dialog. Here it can be decided if incoming messages or chat request should pop up immediately or not. If the latter one is chosen, a blinking icon signalizes an incoming request. The user can also decide if incoming system messages from transports should be ignored. Another preference is whether a double-click on a buddy should open a message or chat dialog, or a phone call. Finally, it can be decided if the enter-key in the message and chat dialog should perform a line break, or if the message is sent.

Beside the 'Manage' menu there is another menu where the user can choose its own presence status. At the screenshot of Figure 8.8 the status is 'Available'. When clicking on the status, a menu pops up, which enables the user to choose another presence status.

8.2.2 Architecture

This section describes the architecture of the Infonova instant messaging client in detail. The program was developed with Microsoft Visual Studio 6.0. As programming language C++ was chosen. The Microsoft Foundation Classes (MFC) provide the framework for the application.

The application consists of three layers (see Figure 8.12). They are separated via well-defined interfaces. The *Presentation Layer* includes the user interface. It also handles events like expanding or collapsing a group in the buddy list.

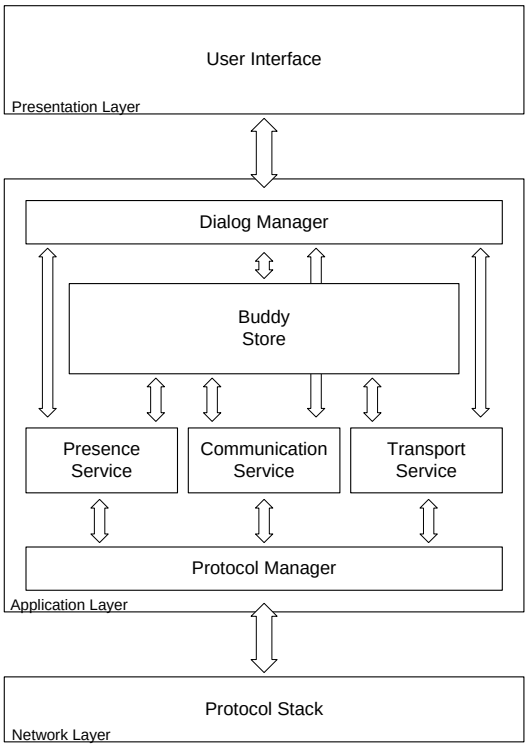


Figure 8.12: Architecture of the Infonova Instant Messaging Client

The *Application Layer* does the main part of the work. The current roster items are stored in the *Buddy Store*. The *Application Layer* includes three service objects. The *Presence Service* object handles presence changes coming from other users. Also subscription requests are managed there. Instant messages and chat sessions are handled within the *Communication Service* object. The *Transport Service* object manages the available Jabber transports. All three service objects send and receive Jabber messages via the *Protocol Manager*, which acts as an interface to the *Protocol Layer*.

The *Protocol Layer* contains the protocol stack, which again consists of three layers. As Figure 8.13 shows, these layers are the *Jabber Protocol Layer*, the *XML Layer* as well as the *Network Layer*.

The reception of data packets is handled in the *Network Layer*. There it is stored in a string variable and forwarded to the *XML Layer*. The *XML Layer* includes an XML parser, which reads the data and creates XML objects. Finally, the *Jabber Protocol Layer* takes these objects and calls the appropriate methods of the clients *Application Layer*.

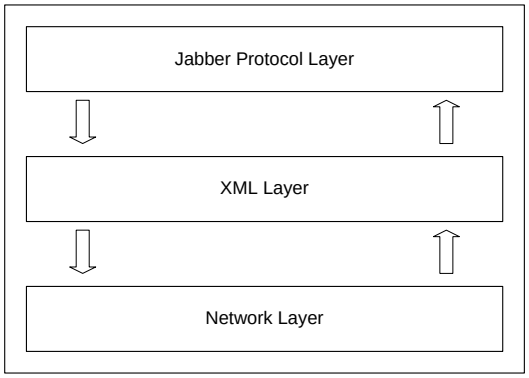


Figure 8.13: Jabber Protocol Stack

As the architectural overview of the Infonova instant messaging client shows, the application was designed to be as flexible as possible. All layers easily can be replaced. For example, if a new user interface is needed, just the *Presentation Layer* has to be replaced.

8.3 Summary

Jabber is an XML-based, open source instant messaging server. The main advantage of this server is, that it offers the possibility to exchange messages and presence information with other instant messaging systems.

Installing and configuring a Jabber server is simple. All information can be found on the Internet and the configuration file is XML-based, and therefore human readable. Jabber uses a client/server architecture, where all data between the clients pass the server.

The procedure for a Jabber session is as follows. The server listens at port 5222 for incoming connection requests. After the authentication procedure, the client receives his roster and the presence information of all elements in the roster. Then, the exchange of messages can begin.

The Infonova instant messaging client is a Microsoft Windows application, which was implemented in C++. It communicates with the Jabber server using the Jabber protocol (see section 8.1.3). The client has the functionality of a common instant messaging client, and additionally uses the Infonova IP telephony client IPSylon for phone calls. It is based on a layered architecture, with well-defined interfaces between the layers. This increases the flexibility of the software design.

Conclusion

We have seen in this thesis that there are many different kinds of communication services. Email, news and IRC are very popular text-based services. Except of a reliable data transport, they do not have great demands on the underlying network. In contrast, IP telephony requires a constant bit rate, and, depending on the speech quality, a high bandwidth. While the current version of IP does not support a mechanism to guarantee QoS, IP version 6 does.

There are two IP telephony signaling protocols for initiating and terminating calls competing. While the H.323 standard is very extensive and complex, SIP is an open, text-based protocol similar to HTTP. For the transmission of real-time data, RTP and RTCP are used in most of the cases.

Instant messaging is another popular communication form. Coupled with a presence service, it enables users to be informed about the online status of their friends.

People who need all these communication services have to use different applications or devices to look for incoming messages. Unified messaging was developed to create a single point of access to the messages of a user. A UMS stores all kinds of messages in a central repository. To access these messages, any supported device (e.g. a mobile phone or a PC) can be used.

All the communication services introduced in this thesis already exist. We do not know what the future holds.

With the utmost probability, circuit-switched networks will vanish in favor of packet-based networks. Then, all available devices can easily communicate with each other, and mobile phones will use H.323 or SIP to make calls.

As bandwidth increases, the possibilities of human communication over networks will change. For example, mobile phones with integrated video camera will certainly become popular.

Also the communication participants will change. At the moment most of the communication is between humans. IP telephony or video over IP are examples for that kind of communication. In the future, more and more communication will take place between humans and a computer, or even between electronic devices.

The size and the capabilities of these devices will also change. PCs as we know them today will not exist anymore. Smart devices with integrated voice recognition will automatically arrange meetings, make financial transactions and can be used as remote control. These devices will also have a navigation system included, so

that the user automatically will be informed, when friends in the buddy list of the instant messenger are near by.

One thing is sure: To realize such ideas, open and flexible standards that can be easily extended must be defined and used. Protocols like SIP show the right way to do this.

Appendix A

Standardization Organizations

Standards are necessary to guarantee the interoperability of different computers, software programs or networks. They describe software protocols, data formats, computer languages, communication interfaces and much more. There are several organizations which develop standards. The most important ones for communication technologies are described in the following sections.

A.1 International Telecommunications Union

International Telecommunications Union (ITU)[1] is an organization of the United Nations. Its headquarters is in Geneva and its task is the standardization of the international telecommunication. The ITU is divided into three parts:

1. Radiocommunication Sector (ITU-R)
2. Telecommunication Standardization Sector (ITU-T)
3. Telecommunication Development Sector (ITU-D)

All this sectors are organized in so called Study Groups, which do the major part of the work. Some examples of standards from ITU-T are ISDN, ATM, Intelligent Networks, X.25 and ADSL.

A.2 International Organization for Standardization

The International Organization for Standardization (ISO)[2] is a worldwide federation of national standards bodies like DIN from Germany or ANSI from the United States of America. The work is done in technical committees, subcommittees and working groups. Alone or together with its partner organization, the International

[1] http://www.itu.ch
[2] http://www.iso.org

Engineering Consortium (IEC)[3], ISO has developed many standards like ISO 9000 or the OSI model. Both, ISO and IEC, often cooperate with the ITU.

A.3 Institute of Electrical and Electronics Engineers

The Institute of Electrical and Electronics Engineers (IEEE)[4] is an institute which has developed many standards for LAN and Metropolitan Area Network (MAN), like Ethernet, Token-Ring or DQDB.

A.4 Internet Society

In the early days of the Internet, the Internet Architecture Board (IAB)[5] was formed to coordinate the evolution of the protocols. In 1992, the IAB was absorbed into the Internet Society (ISOC)[6]. This society includes two important groups:

1. The Internet Engineering Task Force (IETF)[7] is responsible for the design and implementation of new protocols.
2. The Internet Engineering Steering Group (IESG)[8] supervises and reviews the protocols of the IETF.

After completing the design of a new protocol, its specification is published as RFC.

A.5 World Wide Web Consortium

In 1994, Tim Berners-Lee, who invented the web at CERN[9] (the European Organization for Nuclear Research), founded the W3C[10]. This consortium develops specifications for the World Wide Web. These include standards like the HyperText Markup Language (HTML), the Extensible Markup Language (XML) and Cascading Style Sheets (CSS).

[3] http://www.iec.org
[4] http://www.ieee.org
[5] http://www.iab.org
[6] http://www.isoc.org
[7] http://www.ietf.org
[8] http://www.ietf.org/iesg.html
[9] http://www.cern.ch
[10] http://www.w3c.org

Acronyms

ACELP Algebraic Code Excitation Linear Predictive Coding

ACF Admission Confirm

ADPCM Adaptive Differential Pulse Code Modulation

AIM AOL Instant Messenger

ARJ Admission Reject

ARPA Advanced Research Projects Agency

ARPANET Advanced Research Projects Agency Network

ARQ Admission Request

ASN.1 Abstract Syntax Notation Number One

ATM Asynchronous Transfer Mode

Bcc Blind carbon copy

BRI Basic Rate Interface

Cc Carbon copy

CDR Call-Detail Record

CELP Code Excitation Linear Predictive Coding

CIF Common Intermediate Format

CNAME Canonical name

CPIM Common Presence and Instant Messaging

CR Carriage Return

CS-ACELP Conjugate Structure ACELP

CSRC Contributing Source

CSS Cascading Style Sheets

CTI Computer Telephony Integration

DPCM Differential Pulse Code Modulation

DRQ Disengage Request

DTMF Dual Tone Multi Frequency

ESMTP Simple Mail Transfer Protocol Service Extension

FoIP Fax over IP

GCF Gateway Confirm

GRJ Gateway Reject

GRQ Gateway Request

GSM Global System for Mobile Communications

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IAB Internet Architecture Board

ICQ I Seek You

IEC International Engineering Consortium

IEEE Institute of Electrical and Electronics Engineers

IESG Internet Engineering Steering Group

IETF Internet Engineering Task Force

IMAP Interactive Mail Access Protocol

IP Internet Protocol

IRC Internet Relay Chat

IRR Information Request

ISDN Integrated Service Digital Network

ISOC Internet Society

ISUP ISDN User Part

ITU International Telecommunications Union

JOSL Jabber Open Source License

JPEG Joint Photographic Experts Group

LAN Local Area Network

LD-CELP Low Delay Code Excitation Linear Predictive Coding

LDAP Lightweight Directory Access Protocol

LF Line Feed

LPC Linear Predictive Coding

MAN Metropolitan Area Network

MC Multipoint Controller

MCU Multipoint Control Unit

MF Multi Frequency

MFC Microsoft Foundation Classes

MGC Media Gateway Controller

MGCP Media Gateway Controller Protocol

MIME Multipurpose Internet Mail Extensions

MP Multipoint Processor

MP-MLQ Multipulse Maximum Likelihood Quantisation

MPEG Moving Pictures Experts Group

MOS Mean Opinion Score

MTA Message Transfer Agent

MTP Message Transfer Part

NNTP Network News Transfer Protocol

OSI Open Systems Interconnection

PA Presence Agent

PBX Private Branch Exchange

PC Personal Computer

PCM Pulse Code Modulations

PDA Personal Digital Assistant

PER Packet Encoding Rules

PIDF Presence Information Data Format

POP Post Office Protocol

PRI Primary Rate Interface

PSTN Public Switched Telephone Network

PUA Presence User Agent

QCIF Quarter Common Intermediate Format

QoS Quality of Service

RAS Registration, Administration and Status

RCF Registration Confirm

RELP Residual Excited Linear Prediction Coding

RFC Request for Comments

RRJ Registration Reject

RRQ Registration Request

RTCP Real-Time Transport Control Protocol

RTP Real-Time Transport Protocol

SBC Sub-Band Coding

SCCP Signalling Connection Control Part

SCN Switched-Circuit Network

SCP Service Control Point

SDES Source Description Items

SDP Session Description Protocol

SF Single Frequency

SIMPLE SIP for Instant Messaging and Presence Leveraging Extensions

SIP Session Initiation Protocol

SMS Short Message Service

SMTP Simple Mail Transfer Protocol

SQCIF Sub Quarter Common Intermediate Format

SS7 Signalling System 7

SSL Secure Socket Layer

SSP Service Switching Point

SSRC Synchronization Source

STP Signalling Transfer Point

TCAP Transaction Capabilities Application Part

TCP Transmission Control Protocol

TDM Time Division Multiplexing

TIFF Tag Image File Format

TUI Telephony User Interface

TUP Telephone User Part

UCF Unregistration Confirm

UDP User Datagram Protocol

UMS Unified Messaging System

URI Uniform Resource Identifier

URJ Unregistration Reject

URL Uniform Resource Locator

URQ Unregistration Request

VoIP Voice over IP

VPIM Voice Profile for Internet Mail

WAN Wide Area Network

WWW World Wide Web

xdb XML Data Base

XML Extensible Markup Language

Bibliography

[Alfke, 2001] Alfke, J. (2001). *The Jabber Client Developer's Cheat Sheet*. `http://homepage.mac.com/jens/Jabber/JabberClientCheatSheet.html`.

[Alvestrand, 2001] Alvestrand, H. (2001). *RFC 3066: Tags for the Identification of Languages*. Internet Engineering Task Force.

[Atkins and Klyne, 2002] Atkins, D. and Klyne, G. (2002). *Common Presence and Instant Messaging: Message Format*. Internet Engineering Task Force. Internet-Draft.

[Berners-Lee et al., 1998] Berners-Lee, T., Fielding, R., and Masinter, L. (1998). *RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax*. Internet Engineering Task Force.

[Braden, 1989] Braden, R. (1989). *RFC 1123: Requirements for Internet Hosts – Application and Support*. Internet Engineering Task Force.

[Campbell and Rosenberg, 2002] Campbell, B. and Rosenberg, J. (2002). *SIP Extensions for Instant Messaging*. Internet Engineering Task Force. draft-ietf-sip-message-05 (work in progress).

[Collins, 2001] Collins, D. (2001). *Carrier Grade Voice over IP*. McGraw-Hill.

[Crispin, 1996] Crispin, M. (1996). *RFC 2060: Internet Message Access Protocol - Version 4rev1*. Internet Engineering Task Force.

[Crocker et al., 2002] Crocker, D., Diacakis, A., Mazzoldi, F., Huitema, C., Klyne, G., Rosenberg, J., Sparks, R., and Sugano, H. (2002). *Common Presence and Instant Messaging (CPIM)*. Internet Engineering Task Force. Internet-Draft.

[Crocker and Schiller, 1994] Crocker, S. and Schiller, J. (1994). *RFC 1750: Randomness Recommendations for Security*. Internet Engineering Task Force.

[Cuervo et al., 2000] Cuervo, F., Greene, N., Huitema, C., Rayhan, A., Rosen, B., and Segers, J. (2000). *RFC 2885: Megaco Protocol version 0.8*. Internet Engineering Task Force.

[Davidson and Peters, 2000] Davidson, J. and Peters, J. (2000). *Voice over IP Fundamentals*. Cisco Press, 1. edition. ISBN 1578701686.

[Day et al., 2000] Day, M., Rosenberg, J., and Sugano, H. (2000). *RFC 2778: A Model for Presence and Instant Messaging*. Internet Engineering Task Force.

[Feit, 1998] Feit, S. (1998). *TCP/IP: Architecture, Protocols, and Implementation*. McGraw-Hill. ISBN 0-07-022069-7.

[Fielding et al., 1997] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., and Berners-Lee, T. (1997). *RFC 2068: Hypertext Transfer Protocol – HTTP/1.1*. Internet Engineering Task Force.

[Freed and Borenstein, 1996a] Freed, N. and Borenstein, N. (1996a). *RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. Internet Engineering Task Force.

[Freed and Borenstein, 1996b] Freed, N. and Borenstein, N. (1996b). *RFC 2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. Internet Engineering Task Force.

[Freed and Borenstein, 1996c] Freed, N. and Borenstein, N. (1996c). *RFC 2049: Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples*. Internet Engineering Task Force.

[Freed et al., 1996] Freed, N., Klensin, J., and Postel, J. (1996). *RFC 2048: Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures*. Internet Engineering Task Force.

[Handley and Jacobsen, 1998] Handley, M. and Jacobsen, V. (1998). *RFC 2327: Session Description Protocol*. Internet Engineering Task Force.

[Handley et al., 1999] Handley, M., Schulzrinne, H., Schooler, E., and Rosenberg, J. (1999). *RFC 2543: SIP: Session Initiation Protocol*. Internet Engineering Task Force.

[IEC, 2002] IEC (2002). *Unified Messaging*. International Engineering Consortium (IEC). `http://www.iec.org/online/tutorials/unified_mess/`.

[ITU, 1988a] ITU (1988a). *Recommendation 721: 32 kbit/s adaptive differential pulse code modulation (ADPCM)*. International Telecommunication Union.

[ITU, 1988b] ITU (1988b). *Recommendation G.722: 7 kHz Audio-coding within 64 kbit/s*. International Telecommunication Union.

[ITU, 1990] ITU (1990). *Recommendation G.726: 40, 32, 24, 16 kbit/s adaptive differential pulse code modulation (ADPCM)*. International Telecommunication Union.

[ITU, 1992] ITU (1992). *Recommendation G.728: Coding of speech at 16 kbit/s using low-delay code excited linear prediction*. International Telecommunication Union.

[ITU, 1993a] ITU (1993a). *Recommendation H.261: Video codec for audiovisual services at p x 64 kbit/s*. International Telecommunication Union.

[ITU, 1993b] ITU (1993b). *Recommendation Q.931: Digital Subscriber Signaling System No. 1 (DSS1) - ISDN User-Network Interface Layer 3 Specification for Basic Call Control*. International Telecommunication Union.

[ITU, 1994a] ITU (1994a). *Recommendation X.680: Information Technology - Abstract Systax Notation One (ASN.1): Specification of Basic Notation*. International Telecommunication Union.

[ITU, 1994b] ITU (1994b). *Recommendation X.691: Information Technology - ASN.1 Encoding Rules: Specification of Packed Encoding Rules (PER)*. International Telecommunication Union.

[ITU, 1995a] ITU (1995a). *Recommendation G.732.1: Dual Rate Speech codec for multimedia telecommunications transmitting at 6.4 and 5.3 kbit/s*. International Telecommunication Union.

[ITU, 1995b] ITU (1995b). *Recommendation T.127: Multipoint binary file transfer protocol*. International Telecommunication Union.

[ITU, 1996a] ITU (1996a). *Recommendation G.729: Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction (CS-ACELP)*. International Telecommunication Union.

[ITU, 1996b] ITU (1996b). *Recommendation T.120: Data protocols for multimedia conferencing*. International Telecommunication Union.

[ITU, 1997] ITU (1997). *Recommendation T.126: Multipoint still image and annotation protocol*. International Telecommunication Union.

[ITU, 1998a] ITU (1998a). *Recommendation H.450.1: Generic Functional Protocol for the Support of Supplemetary Services in H.323*. International Telecommunication Union.

[ITU, 1998b] ITU (1998b). *Recommendation H.450.2: Call Transfer Supplementary Service in H.323*. International Telecommunication Union.

[ITU, 1998c] ITU (1998c). *Recommendation H.450.3: Call Diversion Supplementary Service in H.323*. International Telecommunication Union.

[ITU, 1998d] ITU (1998d). *Recommendation H.450.4: Call Hold Supplementary Service in H.323*. International Telecommunication Union.

[ITU, 1998e] ITU (1998e). *Recommendation H.450.5: Call Park and Call Pickup Supplementary Service in H.323*. International Telecommunication Union.

[ITU, 1998f] ITU (1998f). *Recommendation H.450.6: Call Waiting Supplementary Service in H.323*. International Telecommunication Union.

[ITU, 1998g] ITU (1998g). *Recommendation H.450.7: Message Waiting Indication Supplementary Service in H.323*. International Telecommunication Union.

[ITU, 1998h] ITU (1998h). *Recommendation T.122: Multipoint communication service - Service definition*. International Telecommunication Union.

[ITU, 1998i] ITU (1998i). *Recommendation T.124: Generic Conference Control*. International Telecommunication Union.

[ITU, 1998j] ITU (1998j). *Recommendation T.125: Multipoint communication service protocol specification*. International Telecommunication Union.

[ITU, 1998k] ITU (1998k). *Recommendation T.37: Procedures for the transfer of facsimile data via store-and-forward on the Internet*. International Telecommunication Union.

[ITU, 1998l] ITU (1998l). *Recommendation T.38: Procedures for real-time Group 3 facsimile communication over IP networks*. International Telecommunication Union.

[ITU, 1999a] ITU (1999a). *Recommendation G.711: Pulse Code Modulation (PCM) of Voice Frequencies*. International Telecommunication Union.

[ITU, 1999b] ITU (1999b). *Recommendation H.323: Packet-based Multimedia Communications Systems*. International Telecommunication Union.

[ITU, 1999c] ITU (1999c). *Recommendation T.123: Network-specific data protocol stacks for multimedia conferencing*. International Telecommunication Union.

[ITU, 2000a] ITU (2000a). *Recommendation H.225.0: Call signalling protocols and media stream packetization for packet-based multimedia communication systems*. International Telecommunication Union.

[ITU, 2000b] ITU (2000b). *Recommendation H.245: Control Protocol for Multimedia Communication*. International Telecommunication Union.

[ITU, 2000c] ITU (2000c). *Recommendation H.263: Enhanced reference picture selection mode*. International Telecommunication Union.

[ITU, 2001] ITU (2001). *Recommendation T.30: Procedures for document facsimile transmission in the general switched telephone network*. International Telecommunication Union.

[Kalt, 2000a] Kalt, C. (2000a). *RFC 2810: Internet Relay Chat: Architecture*. Internet Engineering Task Force.

[Kalt, 2000b] Kalt, C. (2000b). *RFC 2811: Internet Relay Chat: Channel Management*. Internet Engineering Task Force.

[Kalt, 2000c] Kalt, C. (2000c). *RFC 2812: Internet Relay Chat: Client Protocol*. Internet Engineering Task Force.

[Kalt, 2000d] Kalt, C. (2000d). *RFC 2813: Internet Relay Chat: Server Protocol*. Internet Engineering Task Force.

[Kantor and Lapsley, 1986] Kantor, B. and Lapsley, P. (1986). *RFC 977: Network News Transfer Protocol*. Internet Engineering Task Force.

[Köhler, 2002] Köhler, R.-D. (2002). *Voice over IP*. mitp, 1. edition. ISBN 3-8266-4067-5.

[Klensin, 2001] Klensin, J. (2001). *RFC 2821: Simple Mail Transfer Protocol*. Internet Engineering Task Force.

[Klensin et al., 1993] Klensin, J., Freed, N., Rose, M., Stefferud, E., and Crocker, D. (1993). *RFC 1425: SMTP Service Extensions*. Internet Engineering Task Force.

[Kumar, 1999] Kumar, V. (1999). Supplementary Services in the H.323 IP Multimedia Telephony Network. *Intel Technology Journal*, 3rd quarter. `http://developer.intel.com/technology/itj/q31999/pdf/telephony.pdf`.

[Kumar et al., 2001] Kumar, V., Korpi, M., and Sengodan, S. (2001). *IP Telephony with H.323*. John Wiley and Sons, Inc. ISBN 0-471-39343-6.

[Lotus Software, 2002] Lotus Software (2002). *Unified Messaging White Paper*. Lotus Software. `ftp://ftp.lotus.com/pub/lotusweb/product/domino/whitepapers/messaging.p%df`.

[Masinter and Wing, 1999] Masinter, L. and Wing, D. (1999). *RFC 2532: Extended Facsimile Using Internet Mail*. Internet Engineering Task Force.

[McIntyre et al., 1998] McIntyre, L., Zilles, S., Buckley, R., Venable, D., Parsons, G., and Rafferty, J. (1998). *RFC 2301: File Format for Internet Fax*. Internet Engineering Task Force.

[Moor, 1996] Moor, K. (1996). *RFC 2047: Multipurpose Internet Mail Extensions (MIME) Part Three: Message Header Extensions for Non-ASCII Text*. Internet Engineering Task Force.

[Muldowney, 2001] Muldowney, T. (2001). *An Introduction to Components*. http://docs.jabber.org/general/html/component-intro.html.

[Muldowney and Landrum, 2000] Muldowney, T. and Landrum, E. (2000). *The Jabber Programmers Guide*. http://docs.jabber.org/jpg/pdf/main.pdf.

[Nortel Networks, 2000] Nortel Networks (2000). *A Comparison of H.323v4 and SIP*. Nortel Networks.

[Resnick, 2001] Resnick, P. (2001). *RFC 2822: Internet Message Format*. Internet Engineering Task Force.

[Reynolds and Postel, 1994] Reynolds, J. and Postel, J. (1994). *RFC 3265: ASSIGNED NUMBERS*. Internet Engineering Task Force.

[Roach, 2002] Roach, A. B. (2002). *RFC 3265: Session Initiation Protocol (SIP)-Specific Event Notification*. Internet Engineering Task Force.

[Rose, 1991] Rose, M. (1991). *RFC 1225: Post Office Protocol - Version 3*. Internet Engineering Task Force.

[Rosenberg et al., 2002] Rosenberg, J., Willis, D., Schulzrinne, H., Huitema, C., Aboba, B., Gurle, D., and Oran, D. (2002). *SIP Extensions for Presence*. Internet Engineering Task Force. draft-ietf-simple-presence-07 (work in progress).

[Rosenberg and Shockey, 2000] Rosenberg, J. D. and Shockey, R. (2000). The Session Initiation Protocol (SIP): A Key Component for Internet Telephony. *Computer Telephony*, 8:124–139.

[Russel, 2000] Russel, T. (2000). *Signaling System #7*. McGraw-Hill, 3rd edition. ISBN 0071361197.

[Saint-Andre, 2001] Saint-Andre, P. (2001). *Jabber Technology Overview*. http://docs.jabber.org/general/html/overview.html.

[Saint-Andre, 2002] Saint-Andre, P. (2002). *Jabberd Server Howto*. http://jabberd.jabberstudio.org/howto.html.

[Schulzrinne, 2000] Schulzrinne, H. (2000). *IP Networks*. Columbia University. http://www.cs.columbia.edu/~hgs/teaching/ais/slides/videobook.pdf.

[Schulzrinne et al., 1996] Schulzrinne, H., Casner, S., Frederick, R., and Jacobsen, V. (1996). *RFC 1889: RTP: A Transport Protocol for Real-Time Applications*. Internet Engineering Task Force.

[Schulzrinne et al., 1998] Schulzrinne, H., Rao, A., and Lanphier, R. (1998). *RFC 2326: Real Time Streaming Protocol (RTSP)*. Internet Engineering Task Force. ftp://ftp.lotus.com/pub/lotusweb/product/domino/whitepapers/messaging.p%df.

[Schulzrinne and Rosenberg, 1998] Schulzrinne, H. and Rosenberg, J. (1998). *A Comparison of SIP and H.323 for Internet Telephony*. http://www.cs.columbia.edu/~hgs/papers/Schu9807_Comparison.pdf.

[Singh and Schulzrinne, 2000] Singh, K. and Schulzrinne, H. (2000). *Unified Messaging using SIP and RTSP*.

[Sugano et al., 2002] Sugano, H., Fujimoto, S., Klyne, G., and Bateman, A. (2002). *CPIM Presence Information Data Format*. Internet Engineering Task Force. Internet-Draft.

[Tanenbaum, 1996] Tanenbaum, A. S. (1996). *Computer Networks*. Prentice Hall, 3rd edition. ISBN 0-13-349945-6.

[Telogy Networks, 1998] Telogy Networks (1998). *Fax Over Packet White Paper*. Telogy Networks. http://www.telogy.com/our_products/golden_gateway/FOPwhite.html.

[Toyoda et al., 1998] Toyoda, K., Ohno, H., Murai, J., and Wing, D. (1998). *RFC 2305: A Simple Mode of Facsimile Using Internet Mail*. Internet Engineering Task Force.

[Vaudreuil and Parsons, 2002] Vaudreuil, G. and Parsons, G. (2002). *Voice Profile for Internet Mail - version 2*. Internet Engineering Task Force.

[Woodard, 2002] Woodard, J. (2002). *Speech Coding*. University of Southampton, Department of Electronics and Computer Science. http://www-mobile.ecs.soton.ac.uk/speech_codecs.