A SIP of IP-telephony

Master's Thesis

Fredrik Fingal & Patrik Gustavsson

10 February, 1999

Examiner:
Lars Reneby
Department of Communication Systems
Lund Institute of Technology, Lund University

Supervisor:
Peter Hedman
Sigma Exallon Systems AB
Malmö

CODEN:LUTEDX(TETS-5348)/1-44/(1999)&LOCAL 5

ABSTRACT

There are two different protocols competing in the Internet telephony world today, the Session Initiation Protocol (SIP) emerged from IETF and the H.323 emerged from ITU. The IP-telephony market is growing and will most likely affect the traditional circuit-switched telephony business in the future.

We have studied the Session Initiation Protocol in the purpose of evaluating it and to make a small implementation of a client and a server. The tools Rational Rose and Java was used for the implementation. We have also compared it against the H.323 and tried to give a hint of what the future holds.

Table of Contents

1 Introduction

The telephony business is coming to a new era. The IP telephony concept is growing bigger and the number of companies interested is continuously increasing. Eventually this will become a threat to the traditional telephony business. The fact that it is possible to route traditional calls over the IP networks by using gateways is also a big threat to the long distance business for the traditional telephony service providers. Both the ITU and the IETF are involved in the standardization, and their protocols are competing to become the de facto standard.

1.1 Purpose of this master's thesis

This master's thesis consist of two parts. The main purpose is to evaluate the SIP protocol and implement a simple client/server to run on the internal network at Sigma Exallon Systems AB, where the project is executed. The second part is to compare the SIP protocol against the H.323 protocol and give a hint of what the future holds for these protocols and the general Internet telephony concept.

1.2 Our background

Before starting our Master's thesis we had a background of the Electrical Engineering program at the Lund Institute of Technology, Lund University, and a concentration of Software Engineering courses. Only a few courses in programming and no courses about object oriented software engineering or programming are included. Courses in theoretical Computer Communication and Communication Systems are also included in our education. The only practical experience of Internet is using applications such as web browsers and ftp programs.

1.3 How to read this report

Chapter 2 gives a general introduction to the Internet telephony and the organizations developing it.

Chapter 3 presents an introduction to the TCP/IP model that is necessary for the understanding of the protocols used in this report.

Chapter 4 gives an overview of the protocol architecture and explains the surrounding protocols.

Chapter 5 presents an overview of the Session Initiation Protocol. Its components and usage are also explained.

Chapter 6 presents the Session Description Protocol and its components, that are used together with SIP.

Chapter 7 compares the two competing protocols, SIP from IETF and H.323 from ITU, from different aspects. It also includes a discussion based on some articles written.

Chapter 8 presents the practical part of this Master's thesis, including a chronology and the end results.

Chapter 9 is the conclusion of this report. It concludes our results from this project and ends this report.

Chapter 10 contains the list of the references that we have used in this Master’s thesis report.

Appendix A contains a short explanation of relevant words and acronyms used in this report.

Appendix B is a table over the payload types for standard audio and video encodings, used by the RTP.

Appendix C is a copyright note for the use of our audio application called the Robust-Audio Tool.

2 Internet telephony

Today most of the telephony is still made on the old Public Switched Telephone Network (PSTN). This means that a call reserves the connection between the two users and no one else can use this connection. When the call is disconnected the line is free for other users again.

The difference with Internet telephony, also known as Voice-over-IP (VoIP) or IP telephony (IPtel), is that the transport is made on an IP-network. Here it is possible to send packets between two or more parties without reserving the connection. This is done by digitizing the audio signals, recorded by using the microphone of the computer, encapsulating them into packets and then sending them across networks using the Internet protocols. The other side then decapsulates the packets and plays the original message through the speakers of the computer. Other media, such as video and shared applications, are also possible to include without any major changes.

The negative side with this new method is that it is difficult to guarantee any Quality of Service (QoS), as there is no way to guarantee when a packet arrives. This problem could be minimized by not using the public Internet and use private managed networks instead, where a certain QoS can be guaranteed.

The need for signaling functionality distinguishes Internet telephony from other Internet multimedia services such as broadcast and media-on-demand services. This signaling has to be able to create and manage calls. One of the problems in Internet telephony is locating participants for a phone call. Personal mobility, call delegation, availability, and desire to communicate make the process of signaling more complex. For this, it is possible to use the Session Initiation Protocol (SIP) that is part of the protocol stack that has emerged from Internet Engineering Task Force (IETF) [26] – the *Internet Multimedia Conferencing Architecture*. The SIP translates application-layer addresses, establishes and manages calls. Another protocol in use, that emerged from the International Telecommunication Union (ITU) [27], is the H.323, which is similar to the SIP.

One of the largest advantages with Internet telephony is that it is not restricted to a single media or unicast delivery. Compared to the PSTN, the advantage is the transparency of the network to the media carried, so that adding a new media type requires no changes to the network infrastructure. Also, at least for signaling, the support of multiparty calls differs only marginally from two-party calls.

3 The TCP/IP model

When discussing computer communication the concepts of protocol and computer-communications architecture (or protocol architecture) are vital. A protocol may be defined as a set of rules governing the exchange of data between two entities. The key elements of a protocol are

- Syntax. Includes such things as data format and signal levels.
- Semantics. Includes control information for coordination and error handling.
- Timing. Includes speed matching and sequencing.

A protocol is just a module in a structured set of modules that implements the communications function. That structure is referred to as a protocol architecture.

Two protocol architectures have served as the basis for the development of interoperable communications standards: the TCP/IP, Transmission Control Protocol/Internet Protocol, protocol suite and the Open Systems Interconnection (OSI) reference model. TCP/IP is the most widely used interoperable architecture, and the OSI has become the standard model for classifying communications functions.

TCP/IP is a result of protocol research and development conducted on the experimental packet-switched network, ARPANET, funded by the Defence Advanced Research Projects Agency (DARPA). This protocol suite consists of a large collection of protocols that have been issued as standards by the Internet Activities Board (IAB).

There is no official TCP/IP protocol model as in the case of OSI. However, based on the protocol standards that have been developed, the communication task for TCP/IP can be organized into five relatively independent layers [15], see figure 1:

- Application layer. Contains the logic needed to support the user application.
- Transport layer. Provides a flow of data between two end systems involved in the communication.
- Internet layer. Provides communication between two end systems attached to different (but interconnected) networks. It deals with the routing of packets through the path to the destination system.
- Network access layer. Is concerned with the exchange of data between an end system and the network to which it is attached.
- Physical layer. Covers the physical interface between a data transmission device and a transmission medium or network.

Application
Transport
Internet
Network Access
Physical

Figure 1: TCP/IP protocol suite

In a layered architecture, like this, every layer uses the services of the lower layers and provides services to the higher layers. The user data is encapsulated by every layer, from the application layer to the physical layer, adding its control information for handling the packet.

4 Architectural Overview

SIP is designed as a part of the overall IETF multimedia data and control architecture. SIP is very modular. It handles basic call signaling, user location and basic registration. Call control signaling can also be added using an extension. Billing, QoS, session content description and other functionality are orthogonal and are handled by other protocols. This means that a protocol can be modified without affecting the other protocols using it.

The whole Internet telephony protocol stack is shown in figure 2 below. The architecture is currently incorporating many protocols. TCP, UDP, IP and the protocols beneath are used to serve the protocols above them with a possibility to send/receive data to/from other locations across a network. We will only mention them here and then they are further described later in this report:

- Internet Protocol (**IP**) is a network layer protocol that provides an unreliable, connectionless data delivery service on a "best effort basis”, section 4.1.1.
- Transmission Control Protocol (**TCP**)**,** is a connection-oriented transmission protocol on an IP-based network, section 4.1.2.
- User Datagram Protocol (**UDP**), is a connectionless transmission protocol on an IP-based network, section 4.1.3.
- **H.323** is the International Telecommunication Union’s (ITU) protocol for multimedia communication, section 4.1.7.
- The Real Time Streaming Protocol (**RTSP**)**,** is for controlling delivery of streaming media, section 4.1.6.
- The Session Initiation Protocol (**SIP**)**,** is an application-layer control (signaling) protocol for creating, modifying and terminating sessions with one or more participants, section 5.
- The RTP Control Protocol (**RTCP**)**,** provides support for real-time conferencing of groups of any size within an internet, section 4.1.4.
- The Real-time Transport Protocol (**RTP**)**,** is the Internet-standard protocol for the transport of real-time data, including audio and video, section 4.1.4.
- The Resource reSerVation Protocol (**RSVP**)**,** is used for reserving network resources and is a way keeping a certain QoS of the transmission, section 4.1.5.
- The Session Description Protocol (**SDP**), is used for describing multimedia sessions, section 6.

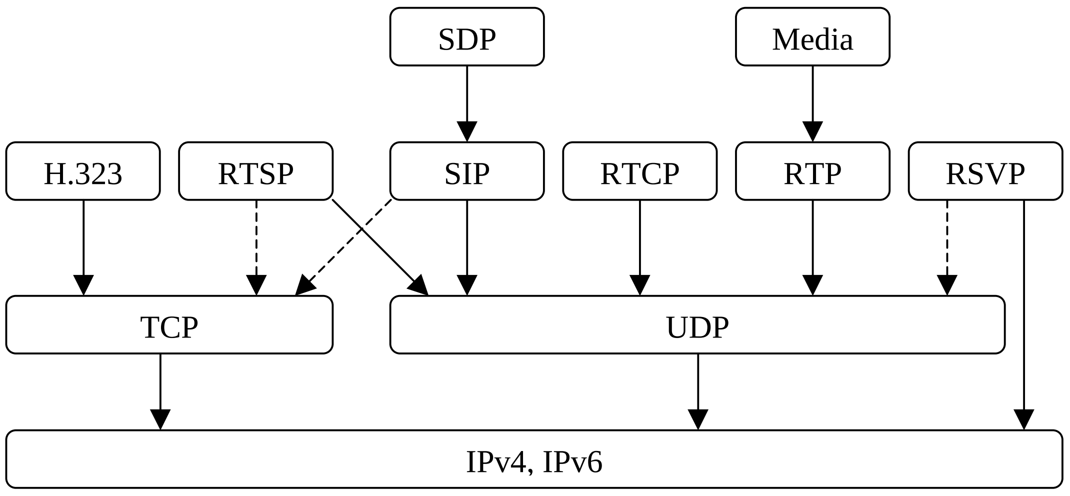

Figure 2: The Internet telephony protocol architecture

4.1 The surrounding protocols

Here we will explain the surrounding protocols in the architectural overview above. The protocols are independent parts and stand alone, but can use and can be used by other protocols. The SIP protocol for example uses the SDP to specify which media to be used in the conference that is being set up. SIP then uses RTP and RTCP to transport the media and to control the media connection.

4.1.1 IP

The Internet Protocol (IP), is part of the TCP/IP protocol suite, and is the most widely used internetworking protocol. IP is a connectionless protocol, which means that there is no established connection between the end points that are communicating. Each packet that travels through the Internet is treated as an independent unit of data without any relation to any other unit of data. It is for the protocol above to keep track of the order between packets and for guaranteeing that the packet arrived, if so wanted. It uses IP-addresses, which is a 32-bit or a 128-bit for the new version IPv6 global internet address. The address is coded to allow a variable allocation of bits to specify network and host. This encoding provides flexibility in assigning addresses to hosts and allows a mix of network sizes on an internet [15]. There are several protocol beneath the IP to handle the network and the physical connection, but they are not mentioned here.

4.1.2 TCP

The Transmission Control Protocol (TCP), is known as a connection-oriented protocol, which means that a connection is established and maintained until such time as the message or messages to be exchanged by the application programs at each end have been exchanged. TCP is responsible for ensuring that a message is divided into the packets that IP manages and for reassembling the packets back into the complete message at the other end. In the OSI communication model, TCP is in layer 4, the Transport Layer [15].

4.1.3 UDP

The User Datagram Protocol (UDP), is known as a connectionless protocol and uses the Internet Protocol to send a data unit (called a datagram) from one computer to another. Unlike TCP, however, UDP does not provide the service of dividing a message into packets (datagrams) and reassembling it at the other end. Specifically, UDP doesn't provide sequencing of the packets that the data arrives in. This means that the application program that uses UDP must be able to make sure that the entire message has arrived and is in the right order. Network applications that want to save processing time because they have very small data units to exchange (and therefore very little message reassembling to do) may prefer UDP to TCP [15].

4.1.4 RTP/RTCP

The Real-time Transport Protocol (RTP), is a transport protocol for real-time data, including audio and video [9], [10]. It can be used for, among others, interactive services such as Internet telephony. RTP consists of a data and a control part. The latter is called RTCP.

The data part of RTP is a protocol providing support for applications with real-time properties such as continuous media (e.g., audio and video), including timing reconstruction, loss detection, security and content identification.

RTCP provides support for real-time conferencing of groups of any size within an internet. This support includes source identification and support for gateways like audio and video bridges as well as multicast-to-unicast translators. It offers quality-of-service feedback from receivers to the multicast group as well as support for the synchronization of different media streams.

4.1.5 RSVP

Resource Reservation Protocol (RSVP), is a protocol that allows channels or paths on the Internet to be reserved for the multicast (one source to many receivers) transmission of video and other high-bandwidth messages. RSVP is part of the Internet Integrated Service (IIS) model, which ensures: best-effort service, real-time service, and controlled link-sharing.

The basic routing philosophy on the Internet is "best-effort," which serves most uses well enough but isn't adequate for the continuous stream transmission required for video and audio programs over the Internet. With RSVP, people who want to receive a particular Internet service can reserve bandwidth through the Internet in advance of the program and be able to receive it at a higher data rate and in a more dependable data flow than usual. When the program starts, it will be multicast to those specific users who have reserved routing priority in advance. RSVP also supports unicast (one source to one destination) and multi-source to one destination transmissions [12], [24].

4.1.6 RTSP

The Real Time Streaming Protocol (RTSP), is an application-level protocol to control the delivery of data with real-time properties. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video. Sources of data can include both live data feeds and stored clips. This protocol is intended to control multiple data delivery sessions, provide a means for choosing delivery channels such as UDP, multicast UDP and TCP, and provide a means for choosing delivery mechanisms based upon RTP [11].

4.1.7 H.323

H.323 is the International Telecommunication Union's protocol for multimedia communication. It is an umbrella recommendation from the ITU that sets standards for multimedia communications over Local Area Networks (LANs) that do not provide a guaranteed Quality of Service (QoS). The H.323 consists of several protocols used for different purposes and work together. It includes H.245 for control, H.225.0 for connection establishment, H.332 for large conferences, H.450.1 H.450.2 and H.450.3 for supplementary services, H.235 for security and H.246 for interoperability with circuit-switched services. Figure 3 below shows how some of them are related. H.323 started out as a protocol for multimedia communication on a LAN segment, but has evolved to fit the more complex needs of Internet telephony [22].

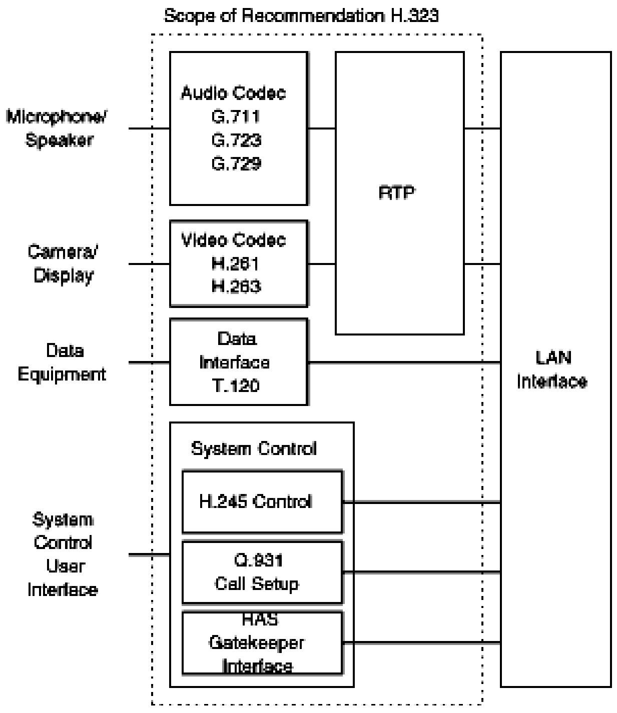

Figure 3: The H.323 components [22]

Below is the official description composed by the ITU:

Recommendation H.323 describes terminals, equipment, and services for multimedia communication over local area networks (LAN) which do not provide a guaranteed quality of service. H.323 terminals and equipment may carry real-time voice, data, and video, or any combination, including videotelephony.
The LAN over which H.323 terminals communicate may [consist of] a single segment or ring, or it may [comprise] multiple segments with complex topologies. It should be noted that operation of H.323 terminals over the multiple LAN segments (including the

Internet) may result in poor performance. The possible means by which quality of service might be assured in such types of LANs/Internetworks is beyond the scope of this recommendation.
H.323 terminals may be integrated into personal computers or implemented in standalone devices such as videotelephones. Support for voice is mandatory, while data and video are optional, but if supported, the ability to use a specified common mode of operation is required, so that all terminals supporting that media type can interwork. H.323 allows more than one channel of each type in use. [21].

5 The Session Initiation Protocol

The Session Initiation Protocol signaling protocol, for creating, modifying and terminating sessions [1], [2], [3]. These sessions can be multimedia conferences, Internet telephone calls and similar application consisting of one or more media types as audio, video, whiteboard etc.. SIP invitations, used to create sessions, carry session descriptions, which allow participants to agree on a set of compatible media types. Participants can be a human user, a “robot” (e.g. media server) or a gateway to another network. These can communicate via multicast or via a mesh of unicast relations, or a combination of these.

SIP is currently under development within the Internet Engineering Task Force (IETF) Multiparty Multimedia Session Control (MMUSIC) working group. They have based SIP on some other protocols coming out of the IETF, in particular the Simple Mail Transfer Protocol (SMTP) and Hyper Text Transfer Protocol (HTTP). Like these SIP is a textual protocol based on the client-server model, with requests generated by one entity (the client), and sent to a receiving entity (the server) which responds them. A request invokes a method on the server and can be sent either over TCP or UDP. The most important SIP method, of the currently six, is the INVITE method, used to initiate a call between a client and a server

This chapter is based on the SIP draft “draft-ieft-mmusic-sip-09” [1], the latest version when writing this is “draft-ieft-mmusic-sip-12” [3]. Additional information is collected from [4], [5], [6], [8], [13], [28].

5.1 SIP components

There are two components in a SIP system, user agents and network servers. A user agent is an end system that acts on behalf of a user. Usually it consists of two parts, a client and a server, as the user probably is wishing to both be able to call and to be called. The client part, User Agent Client (UAC), is used to initiate a SIP request. The server part, User Agent Server (UAS), receives requests and returns responses on behalf of the user.

There are two kinds of network servers, proxy servers and redirect servers. A SIP proxy server forwards requests to the next server after deciding which it should be. This next server could be any kind of SIP server, the proxy does not know and does not have to know. Before the request has reached the UAS it may have traversed several servers. These will be traversed in reverse order by the response. As a proxy server issues both requests and responses it contains both a client and a server.

The other network server type, redirect server, does not forward requests to the next server. Instead it sends a redirect response back to the client containing the address of the next server to contact.

5.2 Addressing and naming

To be able to locate and invite participants there has to be a way the called party could be addressed. The entities addressed by SIP are users at hosts, identified by a SIP URL.

The SIP URL has an email-like identifier of the form user@host. Where the user part can be a user name, a telephone number or a civil name. The host part is either a domain name or a numeric network address. In many cases a user's SIP URL could be guessed from the users email address. Examples of SIP URLs could be:

- sip:patrik@example.com
- sip:beagleboy@176.7.6.1

This URL may well be placed in a web page, so that clicking on the link, as in the case mail URLs, initiates a call to that address.

When using the email address SIP has to resolve the name@domain to user@host. This could lead to different addresses depending on time of the day, media to be used and so on.

5.3 Locating a Server

When a client wishes to send a request it first obtains the address of the participant to be contacted. If the address consists of a numeric IP address the client contact the SIP server there. Else if the address is of the form name@domain the client has to translate the domain part to an IP address where a server may be found. This is done with a DNS lookup. Once the IP address is found the request is sent using either UDP or TCP.

5.4 Locating a User

When the SIP server receives a request it has to locate the user in its domain. The user's location could be of different kind. He could for example be logged in at zero to many hosts or at a different domain. To find the user's actual location there is an outside SIP entity, a location server. Being asked for a user's location the location server returns a list of zero to many locations where the user could be found.

The location can dynamically be registered with the SIP server. The user may also install call handling features at the server. This is done sending a REGISTER request, explained later.

If the location of the user was not found the server sends a response to the client indicating this. Otherwise the action taken by the server varies with the type of the SIP server:

- **Proxy server**: A SIP proxy server can send the request in sequence or in parallel to the locations listed.
- **Redirect Server**: A SIP redirect server can return a response with the list placed in Contact headers. Then the client can send directly to the users location(s).

These two modes are exemplified in section 5.6.

5.5 SIP messages

There are two kinds of SIP messages, requests and responses. Clients issues requests and servers answer with responses. These, requests and responses, include different headers to describe the details of the communication.

Unlike other signaling protocols such as H.323, SIP is a text-based protocol. This makes a SIP header largely self-describing and minimizes the cost of entry. Since most values are textual, the space penalty is limited to the parameter names.

If not designed carefully, text-based protocols can be difficult to parse due to their irregular structure. SIP tries to avoid this by maintaining a common structure of all messages and their header fields, allowing a generic parser to be written. Request and response use a generic-message format which consists of a start-line, one or more header-fields ("headers"), an empty line indicating the end of the header fields, and an optional message-body.

SIP was designed for character-set independence, so that any field can contain any ISO 10646 character. Together with the ability to indicate languages of enclosed content and language preferences of the requester, SIP is well suited for international use.

To make SIP signaling more secure, encryption and authorization can be used. Encryption can for example be used to prevent packet sniffers and other eavesdroppers from seeing who is calling whom. Authorization is used to prevent an active attacker from modifying and replaying SIP requests and responses.

5.5.1 Header-fields

Messages use header-fields to specify such things as caller, callee, the path of the message, type and length of message body and so on. Some of the header fields are used in all messages, the rest is used when appropriate. A SIP application does not need to understand all these headers, though it is desirable. The entity receiving simply silently ignores headers that it does not understand. The order in which the headers appear is generally of no importance, except for the Via field and that hop-by-hop headers appear before end-to-end headers.

There are 37 headers, listed in figure 4, and they can be divided into four different groups of headers:

- **General header fields** apply to both request and response messages.
- **Entity header fields** define information about the message body or, if no body is present, about the resources identified by the request.
- **Request header fields** act as request modifiers and allow the client to pass additional information about the request, and about the client itself, to the server.
- **Response header fields** allow the server to pass additional information about the response which cannot be placed in the Start-Line (in responses called Status-Line).

These header fields give information about the server and about further access to the resource identified by the Request-URI

General-headers	Entity-headers	Request-headers	Response-headers
Call-ID	Content-Encoding	Accept	Allow
Contact	Content-Length	Accept-Encoding	Proxy-Authenticate
CSeq	Content-Type	Accept-Language	Retry-After
Date		Authorization	Server
Encryption		Contact	Unsupported
Expires		Hide	Warning
From		Max-Forwards	WWW-Authenticate
Record-Route		Organization	
Timestamp		Priority	
To		Proxy-Authorization	
Via		Proxy-Require	
		Route	
		Require	
		Response-Key	
		Subject	
		User-Agent	

Figure 4: SIP Headers

In figure 5, some of the most fundamental headers are shortly explained, the rest can be found in [1].

Call-ID	Uniquely identifies a particular invitation or all registrations of a client.
Contact	Contains location(s), that are used in different purposes depending on the message.
Content-Length	Indicates the message body length in bytes
Content-Type	Indicates the media type of the message body
CSeq	(Command Sequence) Uniquely identifies a request within a Call-ID
From	Indicates the initiator of the request
Require	Used by clients to tell the user agent server about options that the server expects the server to support in order to properly process the request.
Subject	Indicates the nature of the call.
To	Specifies the recipient of the request.
Via	Indicates the path taken by the request so far.

Figure 5: Some headers

5.5.2 Requests

The request is characterized by the Start-Line, called Request-Line. It starts with a method token followed by a Request-URI and the protocol version. There are six different kinds of requests in the current version of SIP (version 2.0). They are referred to as methods and are here listed with their functionality.

- **INVITE:** The INVITE method indicates that the user or service is being invited to participate in a session. For a two-party call, the caller indicates the type of media it is able to receive as well as their parameters such as network destination. A success response indicates in its message body which media the callee wishes to receive.

- **ACK:** The ACK request confirms that the client has received a final response to an INVITE. It may contain a message body with the final session description to be used by the callee. If the message body is empty, the callee uses the session description in the INVITE request. This method is only used with the INVITE request.

- **BYE:** The user agent client uses BYE to indicate to the server that it wishes to release the call.

- **CANCEL:** The CANCEL request cancels a pending request, but does not affect a completed request. (A request is considered completed if the server has returned a final response).

- **OPTIONS:** The OPTIONS method solicits information about capabilities, but does not set up a connection.

- **REGISTER:** Conveys information about a user's location to a SIP server.

After the headers following the Request-Line the request may contain a message body which is separated from the headers with an empty line. The message body is always a session description and if present the type of Internet media in it is indicated by the Content-Type header field.

SIP request example:

INVITE sip:pgn@example.se SIP/2.0 Via: SIP/2.0/UDP science.fiction.com From: Fingal <sip:ffl@fiction.com> To: Patrik <sip:pgn@example.se> Call-ID: 1234567890@science.fiction.com CSeq: 1 INVITE Subject: lunch at La Empenada? Content-Type: application/sdp Content-Length: ... v=0 o=ffl 53655765 2353687637 IN IP4 123.4.5.6 s=Chorizo c=IN IP4 science.fiction.com m=audio 5004 RTP/AVP 0 3 5	An INVITE request from ffl@fiction.com to pgn@example.se. As the Content-Type points out the message body contains an SDP session description (explained in chapter 6). The session description indicates that ffl wants to receive RTP audio at port 5004 of payload format 0 (PCMU), 3 (GSM) or 5 (DV14). And that this will be received at science.fiction.com.

5.5.3 Responses

After receiving and interpreting a request message, the recipient responds with a SIP response message, indicating the status of the server, success or failure. The responses can be of different kinds and the type of response is identified by a status code, a 3-digit integer. The first digit defines the class of the response. The other two have no categorization role. The six different classes that are allowed in SIP are here listed with their meaning. These classes can be categorized by provisional and final responses. A provisional response is used by the server to indicate progress, but does not terminate a SIP request. A final response terminates a SIP request.

1xx	Informational	Provisional
2xx	Success	Final
3xx	Redirection	Final
4xx	Client Error	Final
5xx	Server Error	Final
6xx	Global Failure	Final

SIP applications are not required to understand the meaning of all registered response codes, though it is desirable. However applications must be able to recognize the class of the response and treat any unrecognized response as being the x00 response code of the class.

SIP response example:

SIP/2.0 200 OK Via: SIP/2.0/UDP sippo.example.se Via: SIP/2.0/UDP science.fiction.com From: Fingal <sip:ffl@fiction.com> To: Patrik <sip:pgn@example.se> ;tag=25443232 Call-ID: 1234567890@science.fiction.com CSeq: 1 INVITE Content-Type: application/sdp Content-Length: ... v=0 o=pgn 4858949 4858949 IN IP4 198.7.6.5 s=Ok c=IN IP4 pepperoni.example.se m=audio 5004 RTP/AVP 0 3	A response message sent from pepperoni.example.se to sippo.example.se. This message is sent in response to the INVITE above. The status code, 200, in the Start-Line indicates success. When it arrives at sippo.examle.se, the local proxy server, it will remove the first Via field and forward the message to science.fiction.com, indicated by the next Via field.

5.6 Basic Operation

As previously mentioned there are two different ways of handling an incoming SIP request at a SIP server. These are here exemplified using the most important SIP operation, inviting a participant to a call, to show the basic operation of SIP. Figure 6 and figure 7 each shows a fiction example with simplified messages and the provisional responses left out. The user *ffl* at host *science.fiction.com* wants to invite user *pgn*. He obtains *pgn*'s address from his email address, of form name@domain, which is *pgn@example.se*. The client then translates the domain part to a numeric IP address, by a DNS lookup, where a server may be found. This results in the server *sippo* of domain *example.se*. An INVITE request (the same as in the request example in 5.5.2) is then generated and sent to this server (1). The Server accepts the invitation and contacts his location server for a more precise location (2). The location server returns one location of *pgn*, which is at host *pepperoni* (3). These steps are the same for both proxy and redirect server.

In the proxy case (figure 6) the server then issues an INVITE request to the address given by the location server (4). The user agent server at *pepperoni* alerts the user (5), who is willing to accept the call. The acceptance is returned to the proxy server, by a 200 status response (6) (the same as in the response example in 5.5.3). A success response is then sent by the proxy to the original caller (7). This message is confirmed by caller, with an ACK request (8). The ACK request is then forwarded to the callee (9).

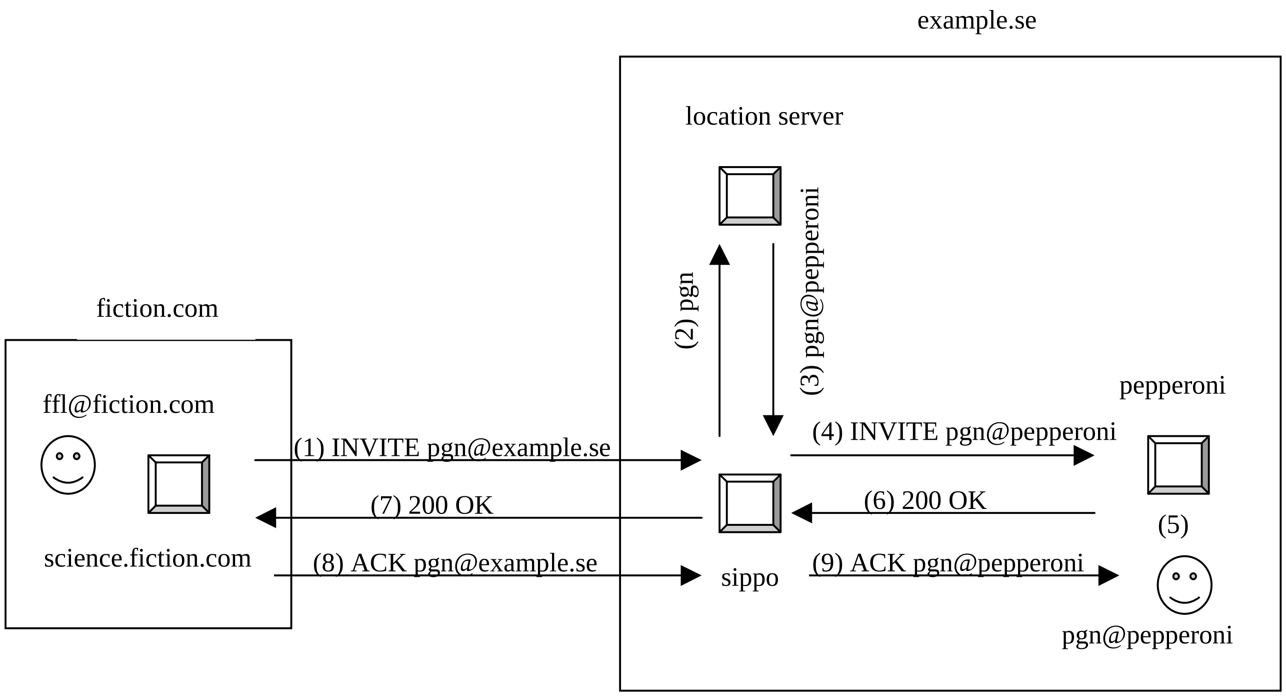

Figure 6: Example of INVITE for SIP proxy

In the redirect case (figure 7) the server returns a redirection response of class 300 giving the address to contact in the Contact header field (4). The caller acknowledges the response with an ACK request to the server (5). The caller issues a new INVITE request with the same Call-ID but a higher CSeq number. This is sent to the address given by the server (6). In this case the call succeeds and a response indicating this is sent to the caller (7). The signaling is completed with an ACK from the caller (8).

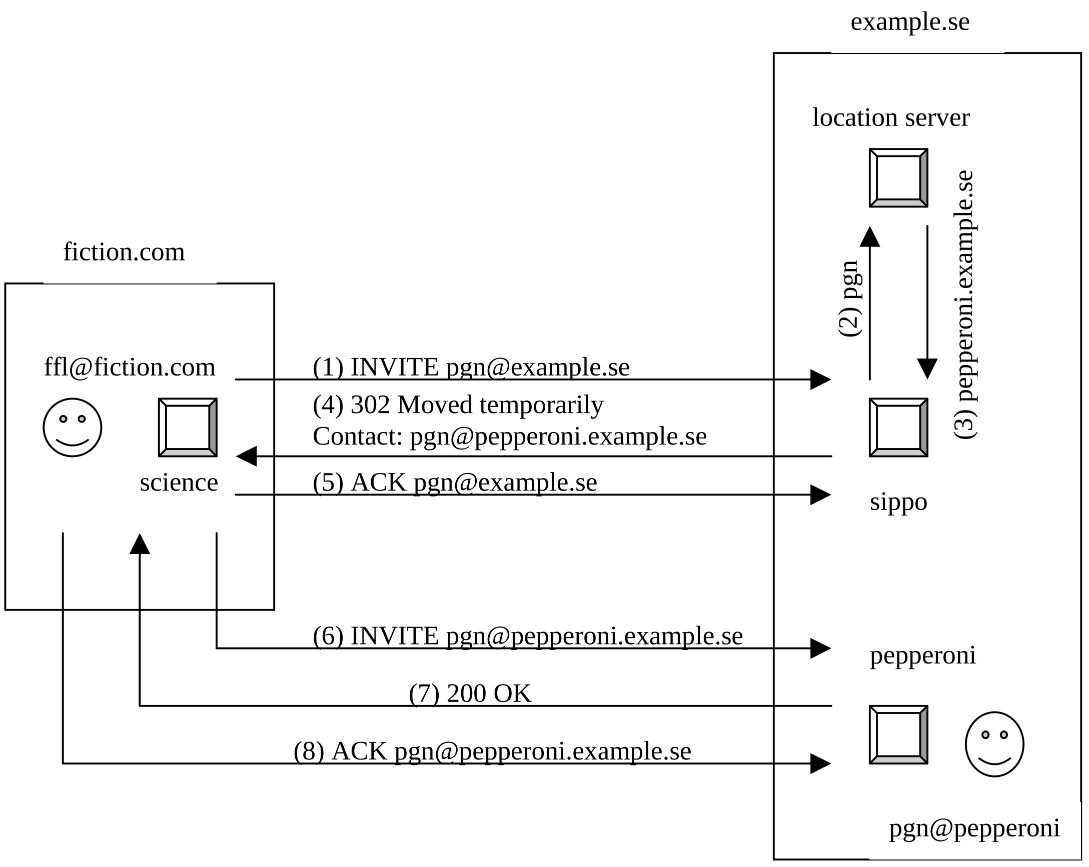
example.se
location server
(2) pgn
(3) pepperoni.example.se
fiction.com
ffl@fiction.com
science
(1) INVITE pgn@example.se
(4) 302 Moved temporarily
Contact: pgn@pepperoni.example.se
(5) ACK pgn@example.se
sippo
(6) INVITE pgn@pepperoni.example.se
pepperoni
(7) 200 OK
(8) ACK pgn@pepperoni.example.se
pgn@pepperoni

Figure 7: Example of INVITE for redirect server

After a successful invitation as in these two examples, the necessary parameters for a session are determined. In the case of RTP audio communication these are the other side's IP-address and port, and the audio codecs to be used. When the RTP connection is set up it is possible to have a conversation. During this conference, either party can change the session by reinviting the other party or for example invite a third party to the conference.

5.7 Extensibility

Since IP telephony is still immature it is most likely that there will be need for additional signaling capabilities in the future. This makes extensibility as well as compatibility among versions key issues. In SIP this is well supported. For example unknown headers and values are ignored. Clients can indicate what feature the server must understand in the Require header. If the server does not support the named feature the request will send a error response with a list of feature that it does understand. These feature can be registered with the Internet Assigned Number Authority (IANA) by any creator of a feature.

5.8 Services

Like in standard telephony, services in Internet telephony is an important issue. SIP does not exactly explain how these should be implemented, but with headers and methods SIP provides elements to construct services. With SIP and its Call Control extension [5] many services similar to the Intelligent Network (IN) services may be implemented, for example Call transfer, Call forwarding and Conference calling. These and many other services are described in *Implementing Intelligent Network Services with the Session Initiation Protocol* [7].

The Session Initiation Protocol: Providing Advanced Telephony Services [4] includes some examples of services. A copy of the Person mobility section is her included to illustrate some features of SIP (the figure is not the same).

“SIP supports advanced personal mobility services. An example of a mobility service is given in" figure 8.

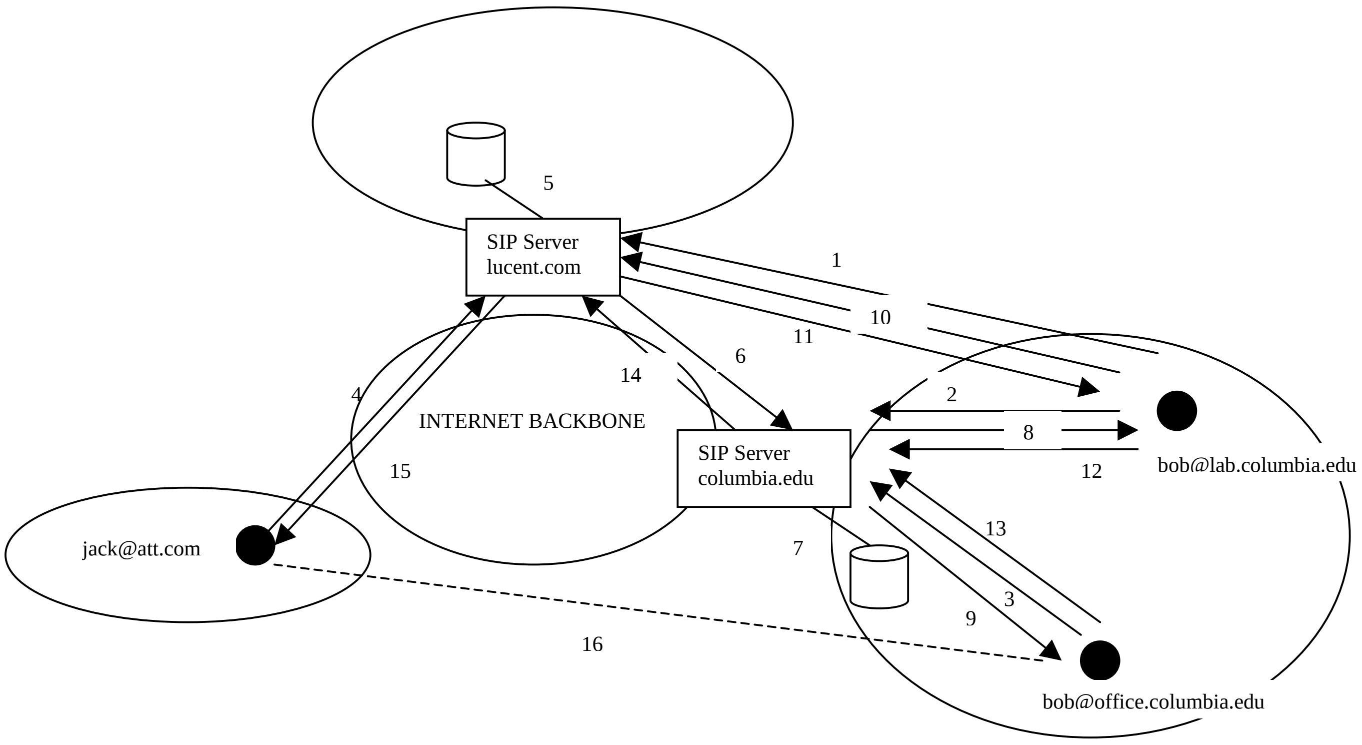

Figure 8: SIP Mobility Example

“Here, a user of the system, Bob, maintains an office at Lucent. However, as an assistant professor at Columbia University, Bob also has a lab and an office there. Bob publishes a single IP telephony phone address for himself, bob@lucent.com. On a day when Bob is at Columbia, he sends a REGISTER message to the Lucent SIP server (1), listing his Columbia address, bob@columbia.edu as a forwarding address. Once at Columbia, Bob register's both his lab machine bob@lab.columbia.edu (2) and his once bob@office.columbia.edu with the Columbia registration server (3). Last time Bob was at Columbia, he set up his lab's computer to automatically forward calls back to his Lucent address. Forgetting about this, Bob restarts his client in the lab with the same configuration.

Later in the day, jack@att.com places a call to bob@lucent.com. Using DNS, the caller resolves lucent.com to the address of the Lucent SIP server, which receives the call request (4). The server checks its registration and policy databases (5), and decides to forwards the request to bob@columbia.edu. To do so, it looks up columbia.edu in DNS, and obtains the address of the main Columbia SIP server. It then forwards the request there (6). Once arriving at the Columbia server, it, too, does a policy and database lookup (7), and determines that there are two potential contacts for Bob. So, the server forks, and sends a call request to both the lab and once machines simultaneously (8,9). This causes the office phone to ring. The lab phone receives the request, and according to its outdated configuration, forwards it back to Lucent (10). Using the loop detection capabilities in SIP, the Lucent server determines that an error has occured, and returns an error response to the lab machine (11). It, in turn, returns an error code to the Columbia server (12).

In the meantime, Bob answers the phone in his office, sending an acceptance response back to the Columbia server (13). Having now received both responses, it forwards the call acceptance back to the Lucent server (14), which forwards the request back to the original caller (15). At this point, the Lucent and Columbia servers can destroy all call state if they so choose. Both future call transactions may proceed directly between the caller and Bob without passing through the intermediate servers (16). The example illustrates a number of salient features of SIP. First, it shows how a call request can cause a “hunt" for a user to ensue, hopping between multiple servers until the final target of the call is found. Secondly, it demonstrates the loop detection features of SIP. Third, it demonstrates how a server can fork requests to speed up the process of contacting the desired user. Finally, it demonstrates how SIP network servers are used in a call only for the initial transaction. Note, however, that this does not lessen their importance in delivering rich mobility services.”

6 Session Description Protocol

The Session Initiation Protocol (SIP) is used to invite users to multimedia conferences. However it only states how communication between inviter and invitee, addressing and user location should be done. There is also need to describe a multimedia session within a SIP request. The protocol used for this purpose is the Session Description Protocol (SDP) [8].

6.1 Overview

The purpose of SDP is to convey information about media streams in multimedia sessions to allow the recipients of a session description to participate in a session [8].

Thus SDP includes:

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

Additional information may be:

- Information about the bandwidth to be used by the conference.
- Contact information for the person responsible for the session.

6.2 SDP Specification

An SDP session description consists of a number of lines of text of the form <type>=<value>. The <type> is always exactly one character and is case-significant. The <value> is a structured text string whose format depends on the <type>. It will also be case-significant unless a specific field defines otherwise. White space is not permitted either side of the ’=’ sign. In general the <value> is either a number of fields delimited by a single space character or a free format string.

An announcement consists of a session-level section followed by zero or more media-level sections. The session-level part starts with a ’v’ line and continues to the first media-level section. The media-level description starts with an ’m’ line and continues to the next media description or end of the whole session description. In general, session-level values are the default for all media unless overridden by an equivalent media-level value. Examples of SDP session descriptions can be found last in this chapter and in section 5.5.

Here follows an overview of the permitted type fields. Some lines in each description are required and some are optional but all must appear in exactly the order given here. Optional items are marked with a ’*’.

Type	Description
v	Protocol version
o	Owner/creator and session identifier
s	Session name
i*	Session information
u*	URI of description
e*	Email address
p*	Phone number
c*	Connection information – not required if included in all media
b*	Bandwidth information
z*	Time zone adjustments
k*	Encryption key
a*	Zero or more session attribute lines

Session Description

Type	Description
t	Time the session is active
r*	Zero or more repeat times

Time Description

Type	Description
m	Media name and transport address
i*	Media title
c*	Connection information – optional if included at session-level
b*	Bandwidth information
k*	Encryption key
a*	Zero or more media attribute lines

Media description

SDP parsers must completely ignore any announcement that contains a <type> letter that it does not understand. The ’attribute’ mechanism is the primary means for extending SDP and tailoring it to particular applications or media. Some attributes have a defined meaning but others may be added on an application-, media- or session-specific basis. A session directory must ignore any attribute it doesn’t understand.

The connection (’c=’) and attribute (’a=’) information in the session-level section applies to all the media of that session unless overridden by connection information or an attribute of the same name in the media description. For instance, in the example below, each media behaves as if it were given a ’recvonly’ (receive only) attribute.

An example SDP description is:

```
v=0
o=molgan 2890844526 2890842807 IN IP4 194.240.47.217
s=imaginary friends
e=molgan@imaginary.com
c=IN IP4 194.47.240.194
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 31
m=application 32416 udp wb
```

For detailed information see [8].

7 SIP vs. H.323

In this chapter SIP and H.323 will be compared. This is mostly based on the paper [16], which is written by spokesmen for the SIP. We have not found any comparisons written by any spokesmen for H.323. The comparison will be done first with an introduction and then the protocols will be compared in a few different aspects [22], [25]. The last section of this chapter is a discussion based on some articles written. They are a little out of date, but they give a pretty good view of the market and the race between SIP and H.323 [17], [18], [19], [20].

7.1 Introduction

To provide useful services, Internet telephony requires a set of control protocols for connection establishment, capabilities exchange and conference control. The SIP and the H.323 are two protocols that meet these needs.

The ITU H.323 is a series of recommendations that defines protocols and procedures for multimedia communications on the Internet. H.323 is mostly based on the ITU multimedia protocols, which preceded it, including H.320 for Integrated Services Digital Network (ISDN), H.321 for B-ISDN and H.324 for GSTN terminals. The encoding mechanisms, protocol fields and basic operations are somewhat simplified versions of the Q.931 signaling protocol.

The SIP, developed in the MMUSIC working group of the IETF, takes a different approach to Internet telephony signaling. It reuses many of the header fields, encoding rules, error codes and authentication mechanisms of HTTP [14].

As both of the protocols will likely use RTP to exchange its data, the QoS of the Internet telephony will not be influenced by the choice of signaling protocol.

7.2 Complexity

If we compare the protocols in the aspect of complexity, H.323 is the most complex of the two protocols. The total sum of the base specifications alone is around 700 pages. SIP, on the other hand, along with its call control extensions and session description protocols totals merely around 130 pages. These numbers tend to get out of date as the protocols are continuously improved, but they give a measure for comparison.

H.323 defines hundreds of elements, while SIP has only 37 headers, each with a small number of values and parameters. H.323 uses a binary representation for its messages, which are based on Abstract Syntax Notation One (ASN.1) and the packed encoding rules (PER). ASN.1 generally requires special code-generators to parse. SIP encodes its messages as text, similar to HTTP and RTSP. This textual encoding simplifies debugging, allowing manual entry and analyzing of messages.

Another advantage of SIP is that it uses a single request that contains all necessary information, while many of the H.323 services require interaction between the several protocol components that are included in the standard.

7.3 Extensibility

Extensibility is a key metric for measuring an IP telephony signaling protocol. As with any heavily used service, the features provided evolve over time as new applications are developed. This makes compatibility among versions a complex issue.

SIP has learned the lessons of HTTP and SMTP (which both have evolved over time), and built in a rich set of extensibility and compatibility functions. By default, unknown headers and values are ignored. Using the Require header, clients can indicate named feature sets that the server must understand. If the server does not support any of the features listed in the header, the server returns an error code and lists the set of features it does not understand.

To further enhance extensibility, numerical error codes are hierarchically organized, as in HTTP. There are the six basic classes, each of which is identified by the hundreds digit in the response code. Basic protocol operation is dictated solely by the class, and terminals need only understand the class of the response. The other digits provide additional information that can be useful but not critical. This allows for additional features to be added by defining semantics for the error codes in a class, while achieving compatibility.

The textual encoding means that header fields are self-describing. As new header fields are added in various different implementations, developers in other corporations can determine usage just from the name, and add support for the field.

H.323 provides extensibility mechanisms as well. These are generally nonstandardParam fields placed in various locations in ASN.1. These parameters contain a vendor code, followed by an opaque value that has meaning only for that vendor. This allows different vendors to develop their own extensions. However it has its limitations. First, extensions are limited only to those places where a non-standard parameter has been added. If a vendor wishes to add a new value to an existing parameter, and there is no placeholder for a non-standard element, it cannot be added. Secondly, H.323 has no mechanism for allowing terminals to exchange information about which extensions each support. As the values in non-standard parameters are not self-describing, this limits interoperability among terminals from different manufacturers.

7.4 Scalability

Scalability is also important as the use of Internet and its services tend to grow. Here the protocols are compared in different levels:

Large Numbers of Domains: As H.323 was originally meant to be used on a single LAN, it has some problems with the scalability even though the newest version defines the concept of zones, and defines procedures for user location across zones for email names. It provides no easy way to perform loop detection in complex multi-domain searches, it can be done statefully by storing messages but this is not scalable. SIP, however, uses a loop detection method by checking the history of the message in the Via header fields, which can be performed in a stateless manner.

Server Processing: In an H.323 system, both telephony gateways and gatekeepers will be required to handle calls from a multitude of users. Similarly, SIP servers and gateways will need to handle many calls. A SIP transaction through several servers and gateways can be either stateful or stateless. This means that large, backbone servers that handle a lot of traffic can be stateless to reduce the memory requirements. This is combined with the ability of using UDP, as UDP does not require any connection state. H.323, on the other, requires its gatekeepers to be stateful. They must keep call state for the entire duration of a call. Furthermore, the connections are TCP based, which means that a gatekeeper must hold its connections throughout a call.

Conference Sizes: H.323 has support for multiparty conferences with multicast data distribution, but it requires a central control point, called a Multipoint Controller, MC, for processing all signaling, for even the smallest conferences. This tends to become a bottleneck for larger conferences. SIP has no need for a central MC and the conference coordination is fully distributed. This improves scalability and complexity but gives a service provider less control.

7.5 Services

Roughly SIP and H.323 provides the same services, even if new services always are added. In addition to call control services, both SIP and H.323 provide capabilities exchange services. In this regard, H.323 provide a much richer set of functionality. Terminals can express their ability to perform various encodings and decodings based on parameters of the codec, and based on which other codecs are in use. SIP only uses basic receiver capability indication. This means that SIP sends a list of the encodings supported and it is for the other side to choose any subset of these.

The service of personal mobility is also supported by both protocols, but H.323's support for this is more limited. SIP can both redirect and proxy incoming requests to a number of locations using any arbitrary URL. Information about language spoken, business or home, mobile phone or fixed, and a list of callee priorities, can be conveyed for each location. SIP also supports, multi-hop "searches" for a user. This means that the servers can proxy the request to one or more additional servers in search of the callee. A SIP server can also proxy the request to multiple servers in parallel, called forking proxy, which makes the search operation more rapid. H.323 can redirect a caller to try several other addresses. Here it is not possible to express preferences, nor can the caller express preferences in the original invitation. H.323 was not engineered for wide area operation, it does support call forwarding, but as mentioned before it has no mechanism for loop detection. H.323 does not allow a gatekeeper to proxy a request to multiple servers either.

H.323 support various conference control services. SIP does not provide conference control, relying instead on other protocols for this service

7.6 Discussion

The race between H.323 and SIP is still pretty open and the question is who will survive, or if they will coexist. Another alternative is that neither of them survive and a new standard takes over, maybe a merge of the two. Jeff Ford, chief technology officer of Inter-Tel Inc., says that H.323 will win the race as it has so many corporations backing it and SIP only has a few [17]. Also, H.323 is already out there which gives it a pretty big head start. Another opinion from Henry Sinnreich, senior engineer in MCI's Internet Engineering Organization, is that SIP is still a young technology, so we have to give it a chance to grow [18]. The article [18] also says that "Some have suggested that SIP is an attempt by MCI—which of course may see Internet telephony as a threat to its core circuit-switched long distance business—to derail the Internet telephony industry.", but we do not believe the IETF would be involved in such a matter.

Even if SIP seems to be the least complex of the two protocols it has to win the confidence in the market. Also, the IETF does not carry the same weight as the ITU does. The ITU has a long, rich history of specifying standards all the way down to the physical interface, while the IETF only deals with Layer 3 and above [17]. This probably gives the ITU an advantage in the race .

Some critics to H.323 say that the standard is too oriented to the old circuit-switching model and too expensive to implement. Henry Sinnreich says "We already have a phone network, and we already have the Internet. Now we have to build an H.323 network that does everything differently." [17].

What SIP offers that H.323 does not is generality and simplicity, explains Henning Schulzrinne [18].

Schulzrinne says that the industry of today does not understand all the features it wants to offer with Internet telephony in the future. This is why it is so important that a protocol allows easy adding of new features, as SIP does. He also says, there is no reason not to implement both protocols because once you have implemented H.323, SIP is like a walk in the park [18].

8 Our implementation

This chapter will explain what we have done during this project including the tools used. The mistakes made during the project will also be pointed out and lessons learned will be passed on.

8.1 The implementation

There are several steps of implementation for a SIP client/server as described in the drafts. The steps builds on a minimal implementation, which contains the essential elements. These are defined as follows and are copied from the draft [1]:

Client

All clients must be able to generate the INVITE and ACK requests. Clients must generate and parse the Call-ID, Content-Length, CSeq, From and To headers. Clients must also parse the Require header. A minimal implementation must understand SDP (RFC 2327). It must be able to recognize the status code classes 1 through 6 and act accordingly.

The following capability sets build on top of the minimal implementation described in the previous paragraph. In general, each capability listed below builds on the ones above it:

> **Basic:** A basic implementation adds support for the BYE method to allow the interruption of a pending call attempt. It includes a User-Agent header in its requests and indicate its preferred language in the Accept-Language header.
>
> **Redirection:** To support call forwarding, a client needs to be able to understand the Contact header, but only the SIP-URL part, not the parameters.
>
> **Negotiation:** A client must be able to request the OPTIONS method and understand the 380 (Alternative Service) status and the Contact parameters to participate in terminal and media negotiation. It should be able to parse the Warning response header to provide useful feedback to the caller.
>
> **Authentication:** If a client wishes to invite callees that require caller authentication, it must be able to recognize the 401 (Unauthorized) status code, must be able to generate the Authorization request header and must understand WWW-Authenticate response header.
> If a client wishes to use proxies that require caller authentication, it must be able to recognize the 407 (Proxy Authentication Required) status code, must be able to generate the Proxy-Authorization request header and understand the Proxy-Authenticate response header.

Server

A minimally compliant server implementation must understand the INVITE, ACK; OPTIONS and BYE requests. A proxy server must also understand CANCEL. It must parse and generate, as appropriate, the Call-ID, Content-Length, Content-Type, CSeq, Expires, From, Max-Forwards, Require, To and Via headers. It must echo the CSeq and the Timestamp headers in the response. It should include the server header in its response.

From these steps we have chosen to implement the user agent as a minimal implementation with the added support for the BYE request. The proxy server is implemented as a minimal and stateless proxy server. The whole implementation is based on the draft -09 [1], but some improvements from the draft -11 [2] are also used. Due to lack of time the full requirements for a minimal implementation are not satisfied, the shortage is documented but not a part of this report.

The functionality of our system:

- Connect a two party call.
- Pending calls are prompted but denied.
- Register at the Location Service, this is done outside of the SIP.
- Proxying incoming requests at the proxy server.
- Forwarding incoming responses at the proxy server.

The client/server was built to run on UDP, as this is the preferred transport of SIP messages [1], and only supporting unicast. The Location Service and the registration was built as separate parts of the system, outside of the SIP, and run on TCP. We chose TCP for the registration because it is reliable and we did not want to handle the retransmitting to achieve a reliable connection ourselves. The Location Service is implemented as a separate database and runs on the same computer as the proxy server. Therefore no network communication is needed between the two. Figure 9 below shows an overview of our system.

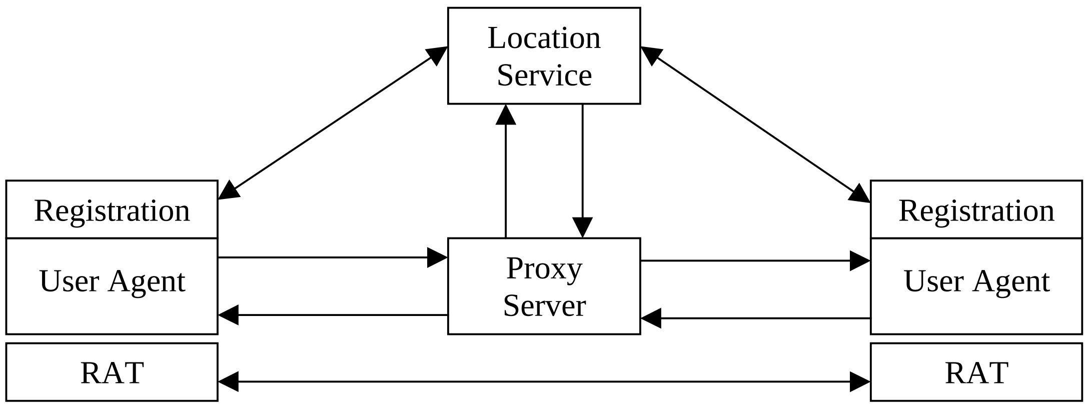

Figure 9: System overview

Because of firewalls and our system not being a full implementation it would not be possible to connect a call outside of Sigma Exallon System AB's internal network.

8.2 Chronology

The project started with a general study of all the material that we found relevant. This included SIP, SDP, RTP, H.323, HTTP and others. The first decisions to make were to decide which platform to use and what programming language to develop in. We choose Windows NT 4.0 as our platform, because this is the common platform used at Sigma Exallon Systems AB where the project was executed and the fact that we are most experienced with this platform. We chose Java as our programming language. The reasons for this is platform independence and the fact that SIP is text-based makes it easier with a language with good support for text-processing. The choice of language also made the platform decision less important. Another early decision was to let an external application handle the audio communication and to concentrate on the SIP signaling.

When a theory base was established we started to draw up requirements of what we wanted our system to be able to do. We only set some basic requirements such as connect a two party call and prototyped our system within time. One requirement from Sigma Exallon Systems AB was the use of Object Oriented Design. As we were new to this concept we choose to use a program called Rational Rose, RR, for the design [29]. We choose this because RR includes a straight forward step by step method and introduction to Object Oriented Design and to the software itself. Our experience of RR is that it is an easy tool to use and it is easy to get an overview of the whole design. Afterwards, the use of RR seems a bit as a mistake and studying the concepts of client/server solutions would seem more appropriate than to put a lot of time on the Object Orientation. This might have made the design less Object Oriented, but it would probably have given us a better design. Studying the overall concept of client/server solutions would probably have improved the end result of our application too.

When the design was done, the programming begun. We started with a small client and a server, just to be able to get a packet from one computer to another. This was very efficient and the threshold to learn Java and at the same time develop an application did not seem difficult. Our experience with Java is very good. It is a powerful tool for this kind of projects and includes a lot of support for both network communication and text processing.

The end result was a system where it is possible to make two-party calls on the Sigma Exallon Systems AB network. We also increased our knowledge level on general Internet protocols and the abilities of developing it. The concept of object orientation is introduced to us, as well as Java-programming, threads and client/server solutions.

8.3 The Audio application

The application we chose for the media communication is called Robust-Audio Tool (RAT) [23], see appendix C for the copyright note. A snapshot of the operating window for the RAT is shown in figure 10 below. This window shows a multiparty call where the highlighted name is currently transmitting to the other participants.

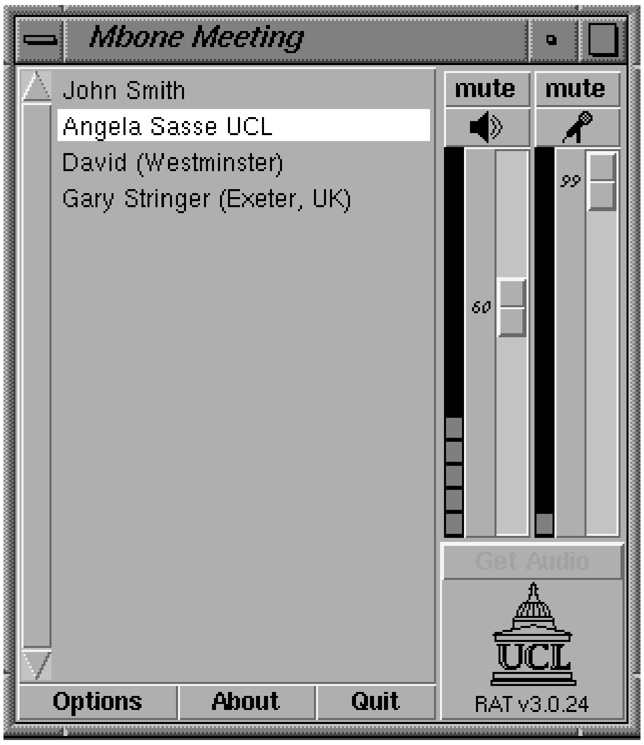

Figure 10: The RAT operating window

It is possible to communicate with full duplex if the soundcard supports it, otherwise half duplex is used. These options are set in the ‘Options’ menu in the RAT operating window. It is not possible to chose full duplex if the soundcard does not support it, and this is set automatically when the RAT is starting. The default option is full duplex if available. In the mode of half-duplex the option: ”Net mutes mike” or “Mike mutes net” is chosen.

This program supports five different audio codecs, as shown in box 1 below. These codec names are used as options when the RAT is started to set which codec to use. These codecs can be set as in parameters when starting the RAT. Each codec is represented by a Payload Type (PT) value in the SDP’s m field. The PTs are shown in appendix B and are copied from [10]. These PTs are registered with the Internet Assigned Numbers Authority, IANA [31]. A new RTP payload format specification may be registered with the IANA by name.

CODECs
Five types of audio encoding are currently possible with RAT, although more are in development. The types of encoding are:

Name	Bit rate	Description
L16	128 kbit/s	Linear PCM at 16 bits per sample
PCM	64 kbit/s	µ-law companded PCM at 8 bits per sample (G711)
DVI	32 kbit/s	Intel's DVI ADPCM at 4 bits per sample
GSM	13.2 kbit/s	EDSI Group Systeme Mobile CODEC
LPC	5.8 kbit/s	Ron Zuckerman's 10 pole LPC CODEC

Box 1: RAT CODECs [30]

The RAT also needs an IP-address and a port number to know where to send the outgoing audio. These are set as in parameters when the RAT is started. One restriction with this program is that both end users must talk on the same port number. Also as specified in the RTP protocol definition, RTP data is to be carried on an even UDP port number and the corresponding RTCP packets are to be carried on the next higher (odd) port number. The port pair may be allocated randomly by a session management program. A single fixed port number pair cannot be required because multiple applications using this profile are likely to run on the same host. Here, port numbers 5004 and 5005 have been registered for use with this profile for those applications that choose to use them as the default pair [9].

9 Conclusion

The matter of the future for Internet telephony is hard to predict, it is for the companies to decide. One important aspect is the ability to get paid for providing an Internet telephony service. An easy way to do this would increase the interest of companies looking for opportunities. But, using HTTP does not cost anything and neither does e-mail. Who would want to pay for Internet telephony. Big operators for the PSTN are already starting to lower their call rates in order to be able to compete with the future Internet telephony. If no one gets paid for providing the service then no one will handle the maintenance and QoS will be difficult to guarantee.

MCI is one of Session Initiation Protocol’s biggest backers, but also companies like 3Com, Lucent and DynamicSoft have put out commercial products. The amount of companies backing SIP is crucial for survival. The low complexity level of the SIP, compared to H.323, should make companies interested. To us it seems like SIP is the better solution for running on the Internet, but to connect with the PSTN, H.323 seems to be the stronger of the two.

Our experience, only having worked with SIP, is that it is an easy protocol to understand and even though we have never worked with projects like this, it was possible to execute it without any major problems. The application developed was also implemented successfully even though it was not a full implementation due to lack of time.

10 References

[1] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg, “SIP: Session Initiation Protocol”, Internet Draft, Internet Engineering Task Force, September 1998, work in progress.

[2] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg, “SIP: Session Initiation Protocol”, Internet Draft, Internet Engineering Task Force, December 1998, work in progress.

[3] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg, “SIP: Session Initiation Protocol”, Internet Draft, Internet Engineering Task Force, January 1999, work in progress.

[4] H. Schulzrinne and J. Rosenberg, "The Session Initiation Protocol: Providing Advanced Telephony Services Across the Internet," Bell Labs Technical Journal,
1999.

[5] H. Schulzrinne and J. Rosenberg, “SIP call control services”, Internet Draft, Internet Engineering Task Force, February 1998, work in progress.

[6] H. Schulzrinne and J. Rosenberg, “Signaling for internet telephony”, Technical Report CUCS-005-98, Columbia University, New York, February 1998.

[7] J. Lennox, H. Schulzrinne and T. F. La Porta, “Implementing Intelligent Network Services with the Session Initiation Protocol”, Columbia University Computer Science Technical Report CUCS-002-99, January 1999.

[8] V. Jacobson and M. Handley, “SDP: Session Description Protocol”, RFC 2327, Internet Engineering Task Force, April 1998, work in progress.

[9] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," RFC 1889, Internet Engineering Task Force, Jan. 1996.

[10] H. Schulzrinne, “RTP profile for audio and video conferences with minimal control”, Internet draft, to eventually obsolete RFC 1890, Internet Engineering Task Force, November 1998.

[11] H. Schulzrinne, R. Lanphier, and A. Rao, "Real time streaming protocol (RTSP)," RFC 2326, Internet Engineering Task Force, Apr. 1998.

[12] B. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reSerVation Protocol (RSVP) – version 1 functional specification," RFC 2205, Internet Engineering Task Force, Oct. 1997.

[13] Henning Schulzrinne and Jonathan Rosenberg, “Internet Telephony: Architecture and Protocols -- an IETF Perspective” July 2 1998.

[14] R. Fielding, J. Gettys, J. Mogul, H. Nielsen, and T. Berners-Lee, "Hypertext transfer protocol -- HTTP/1.1," RFC 2068, Internet Engineering Task Force, Jan. 1997.

[15] William Stallings, “Data and computer communications”, Fifth edition, 1997.Prentice Hall.

[16] H. Schulzrinne and J. Rosenberg, “A Comparison of SIP and H.323 for Internet Telephony”, Network and Operating System Support for Digital Audio and Video (NOSSDAV), Cambridge, England, July 1998.

[17] Dawn Bushaus, “Bellheads vs.’Netheads”, Tele.com, May 1998.
http://www.teledotcom.com/0598/features/tdc0598cover1_side1.html

[18] Paula Bernier, “The Standards Struggle. Will SIP be a Drain on H.323's momentum”, Sounding Board, May/June 1998.
http://www.soundingboardmag.com/articles/851feat2.html

[19] Stephania H. Davis, “Double standard”, IP.net, June 29, 1998
http://www.internettelephony.com/archive/internet1998/6.29.98IPNet/DavisIPN.htm

[20] Vince Vittore, “ H.323: IP telephony's panacea or Pandora's box?”, April 6 1998.
http://www.internettelephony.com/archive/4.06.98/nmnews.html

[21] Johan Zander, “IP telephony for dummies”, Master’s thesis, Lund Institute of Technology, Lund University, June 1998.

[22] A Primer on the H.323 Series Standard, Version 2.0.
http://www.databeam.com/h323/h323primer.html

[23] Robust Audio Tool.
http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/

[24] Whatis.com
http://www.whatis.com

[25] “Comments about H.323 and SIP” August 1998.
http://www.cs.columbia.edu/~hgs/sip/h323.html

[26] International Engineering Task Force, IETF.
http//:www.ietf.org

[27] International Telecommunications Union, ITU.
http//:www.itu.int

[28] SIP: Session Initiation Protocol.
http://www.cs.columbia.edu/~hgs/sip/

[29] Rational Software.
http//:www.rational.com

[30] “User Guide for RAT”, University College London, Computer Science Department, September 1998.

[31] Internet Assigned Numbers Authority (IANA)
http://www.iana.org

Appendix A: Wordlist

ASN.1	**A**bstract **S**yntax **N**otation **One** is a standard way to describe a message (a unit of application data) that can be sent or received in a network. ASN.1 is divided into two parts: (1) the rules of syntax for describing the contents of a message in terms of data types and content sequence or structure and (2) how you actually encode each data item in a message.
Client	A client is the requesting program or user in a client/server relationship. For example, the user of a Web browser is making client requests for pages from servers all over the Web.
Client/Server	Client/server describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfills the request.
DNS	The **D**omain **N**ame **S**ystem is the way that Internet domain names are located and translated into IP addresses.
H.323	H.323 addresses how audio, video and data communication should function over IP based networks, e.g. Intranets or the Internet.
HTTP	The **H**yper**T**ext **T**ransfer **P**rotocol is an application-level protocol used by the World-Wide Web to distribute global information using web browsers.
IETF	**I**nternet **E**ngineering **T**ask **F**orce is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the operation of the Internet. It is open to any interested individual.
Internet-Draft	Internet-Drafts are working documents of the Internet Engineering Task Force, its areas, and its working groups. Other groups may also distribute working documents as Internet-Drafts.
IP-address	**I**nternet **P**rotocol address is a 32-bit number, or 128-bit in IPv6, that identifies each sender or receiver of information that is sent in packets across the Internet.
ITU	**I**nternational **T**elecommunication **U**nion is an international organization within which governments and the private sector coordinate global telecom networks and services.

Port	In programming, a port is a "logical connection place" at an IP-address where the server application listens for incoming data.
Protocol	In information technology, a protocol is the special set of rules for communicating that the end points in a telecommunication connection use when they send signals back and forth. Protocols exist at several levels in a telecommunication connection.
Proxy-server	In an enterprise that uses the Internet, a proxy server is a server that acts as an intermediary between a workstation user and the Internet so that the enterprise can ensure security, administrative control, and caching service.
PSTN	**P**ublic **S**witched **T**elephone **N**etwork: The traditional way of telephony.
QoS	**Q**uality **o**f **S**ervice: On the Internet and in other networks, QoS is the idea that transmission rates, error rates, and other characteristics can be measured, improved, and, to some extent, guaranteed in advance.
RFC	**R**equest **F**or **C**omments: The IETF definition of this is: “ The Requests for Comments are a series of notes, started in 1969, about the Internet (originally the ARPANET). The notes discuss many aspects of computing and computer communication focusing in networking protocols, procedures, programs, and concepts, but also including meeting notes, opinion, and sometimes humor.”.
RSVP	**R**esource Re**s**er**v**ation **P**rotocol is a protocol that allows channels or paths on the Internet to be reserved for the multicast (one source to many receivers) transmission of video and other high-bandwidth messages.
RTCP	**RT**P **C**ontrol **P**rotocol: Monitors the QoS and carries information about the participants in an on-going session.
RTP	**R**eal-time **T**ransport **P**rotocol: Continuous media data like audio and video is carried in RTP data packets.
RTSP	**R**eal **T**ime **S**treaming **P**rotocol: For controlling delivery of streaming media.
SAP	**S**ession **A**nnouncement **P**rotocol: A protocol for advertising multimedia sessions via multicast.
SDP	**S**ession **D**escription **P**rotocol: A protocol for describing multimedia sessions.

Server	In general, a server is a computer program that provides services to other computer programs in the same or other computers.
SIP	**S**ession **I**nitiation **P**rotocol: A signaling protocol for setting up a media conference with two or more participants.
SMTP	**S**imple **M**ail **T**ransport **P**rotocol is a TCP/IP protocol governing electronic mail transmission and reception. The details of SMTP are in RFC 821 of the IETF.
TCP	**T**ransmission **C**ontrol **P**rotocol: A connection-oriented transmission protocol on an IP-based network.
UDP	**U**ser **D**atagram **P**rotocol: A connectionless transmission protocol on an IP-based network.
URI	**U**niform **R**esource **I**dentifier: A way of identifying a point of content on the internet. The most common URI is a URL.
URL	**U**niform **R**esource **L**ocator: The address of a file (resource) accessible on the Internet.
UTF-8	**U**SC (Universal Character Set) **T**ext **F**ormat, 8 bit encoding system of 16 bit text.
VoIP	**V**oice **o**ver **IP**: A term used in IP telephony for a set of facilities for managing the delivery of voice information using the Internet Protocol.

Appendix B: Table of payload types (PT)

PT	encoding name	media type	clock rate (Hz)	channels (audio)
0	PCMU	A	8000	1
1	1016	A	8000	1
2	G726-32	A	8000	1
3	GSM	A	8000	1
4	G723	A	8000	1
5	DVI4	A	8000	1
6	DVI4	A	16000	1
7	LPC	A	8000	1
8	PCMA	A	8000	1
9	G722	A	16000	1
10	L16	A	44100	2
11	L16	A	44100	1
12	QCELP	A	8000	1
13	unassigned	A		
14	MPA	A	90000	(see text)
15	G728	A	8000	1
16	DVI4	A	11025	1
17	DVI4	A	22050	1
18	G729	A	8000	1
19	CN	A	8000	1
20	unassigned	A		
21	unassigned	A		
22	unassigned	A		
23	unassigned	A		
24	unassigned	V		
25	CelB	V	90000	
26	JPEG	V	90000	
27	unassigned	V		
28	nv	V	90000	
29	unassigned	V		
30	unassigned	V		
31	H261	V	90000	
32	MPV	V	90000	
33	MP2T	AV	90000	
34	H263	V	90000	
35—71	unassigned	?		
72—76	reserved	N/A	N/A	N/A
77—95	unassigned	?		
96—127	dynamic	?		
dyn	RED	A		
dyn	MP1S	V	90000	
dyn	MP2P	V	90000	

Table 1: Payload types (PT) for standard audio and video encodings [10].

Appendix C: Copyright note for the Robust-Audio Tool