Master’s thesis

Video telephony in an IP-based set-top box environment

by

Robert Högberg

LITH-IDA/DS-EX--04/036--SE

2004-04-08
Master’s thesis

Video telephony in an IP-based set-top box environment

by

Robert Högberg

LITH-IDA/DS-EX--04/036--SE

2004-04-08

Supervisor: Henrik Carlsson
Examinator: Petru Eles

Kreatel Communications AB
Linköping Institute of Technology
Videotelefi för IP-baserade set-top-boxar

Video telephony in an IP-based set-top box environment

Robert Högberg

This thesis evaluates and shows an implementation of a video telephony solution for network connected set-top boxes based on the SIP protocol for managing sessions.

Unlike other video telephony implementations the set-top box does not handle both audio and video, but only video. A separate phone is used to handle audio. To maintain compatibility with other video telephony implementations, which expect a single SIP device with both audio and video capabilities, a mechanism to merge the audio (SIP-phone) and video (set-top box) into a single entity was developed using a back-to-back user agent.

Due to the set-top boxes' limited hardware it could be impossible to have video compression and decompression performed by the set-top boxes. However, numerous performance tests of compression algorithms showed that the computational power available in the set-top boxes is sufficient to have acceptable frame rate and image quality in a video telephony session. A faster CPU or dedicated hardware for video compression and decompression would however be required in order to compete with dedicated video telephony systems available today.

The implemented video telephony system is based on open standards such as SIP, RTP and H.261, which means interoperability with other video telephony implementations, such as Microsoft's Windows Messenger 4.7, is good.
På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Robert Högberg
Abstract

This thesis evaluates and shows an implementation of a video telephony solution for network connected set-top boxes based on the SIP protocol for managing sessions.

Unlike other video telephony implementations the set-top box does not handle both audio and video, but only video. A separate phone is used to handle audio. To maintain compatibility with other video telephony implementations, which expect a single SIP device with both audio and video capabilities, a mechanism to merge the audio (SIP-phone) and video (set-top box) into a single entity was developed using a back-to-back user agent. Every video telephony call passes through the back-to-back user agent and if the back-to-back user agent notices that either party does not have video capabilities it will try to contact a set-top box to have video capabilities that way.

Due to the set-top boxes’ limited hardware it could be impossible to have video compression and decompression performed by the set-top boxes. However, numerous performance tests of compression algorithms showed that the computational power available in the set-top boxes is sufficient to have acceptable frame rate and image quality in a video telephony session. A faster CPU or dedicated hardware for video compression and decompression would however be required in order to compete with dedicated video telephony systems which have superior image quality and frame rate compared to what is possible with the set-top box today.

The implemented video telephony system is based on open standards such as SIP, RTP and H.261, which means interoperability with other video telephony implementations, such as Microsoft’s Windows Messenger 4.7, is good.
Acknowledgements

I wish to express my deepest gratitude to Kreatel Communications AB for letting me perform this very exciting master thesis work at the company, to everyone at Kreatel and especially my supervisor, Henrik Carlsson, for the valuable support, ideas, opinions and suggestions.

I would also like to thank professor Petru Eles at Linköping Institute of Technology for helping me improve this thesis.
## Contents

1 Introduction ........................................................................................................ 1  
   1.1 Background .................................................................................................. 1  
   1.2 Purpose ......................................................................................................... 1  
   1.3 Limitations ................................................................................................... 1  
   1.4 Methods ........................................................................................................ 2  
   1.5 Sources ......................................................................................................... 2  
   1.6 Structure ....................................................................................................... 3  
   1.7 Glossary ....................................................................................................... 3  

2 IP based set-top boxes ........................................................................................ 9  
   2.1 Hardware ...................................................................................................... 9  
   2.2 Software environment .................................................................................. 10  

3 Video telephony .................................................................................................. 11  

4 SIP ....................................................................................................................... 13  
   4.1 SIP operations ............................................................................................ 15  
      4.1.1 Setting up a session ............................................................................. 15  
      4.1.2 Modifying a session ............................................................................. 15  
      4.1.3 Terminating a session ......................................................................... 16  
      4.1.4 Location registration .......................................................................... 16  
      4.1.5 Capability discovery ........................................................................... 16  
   4.2 SIP building blocks ..................................................................................... 17  
      4.2.1 User agent ............................................................................................ 17  
      4.2.2 Proxy ..................................................................................................... 17  
      4.2.3 Registrar ............................................................................................... 18  
      4.2.4 Back-to-back user agent ...................................................................... 18  
   4.3 Locating SIP users ..................................................................................... 19  
   4.4 SIP message structure ................................................................................ 20  
   4.5 SDP ............................................................................................................ 22  
      4.5.1 Media negotiation ................................................................................. 24  

5 Media transport ................................................................................................... 27  
   5.1 RTP ............................................................................................................ 27  
   5.2 RTCP ......................................................................................................... 27  

6 Image processing ................................................................................................. 29  
   6.1 Colour spaces ............................................................................................. 29  
      6.1.1 RGB ..................................................................................................... 29  
      6.1.2 YUV ..................................................................................................... 29  
   6.2 Video Compression .................................................................................... 30  
   6.3 Intra encoding ............................................................................................. 31  
      6.3.1 Discrete Cosine Transform (DCT) ....................................................... 32  
      6.3.2 Quantization ........................................................................................ 32  
      6.3.3 Run-length encoding ........................................................................... 32  
      6.3.4 Huffman coding .................................................................................. 33
1 Introduction

1.1 Background
Video telephony, where users can communicate both through audio and video, has been available for many years. However, since the equipment has been expensive and the networks installed in most users’ homes have not been capable of handling the bandwidth needed for video telephony, it has not got any widespread use. As more people get broadband connected these days it is believed that video telephony will become more popular in the future.

For manufacturers of network connected devices it is of course interesting to be able to offer their customers as many services as possible in a single device since it can lead to increased revenue and popularity of the device. Because of this, video telephony could be a useful feature to add to networked set-top boxes.

1.2 Purpose
The purpose of this thesis work has been to implement and evaluate a video telephony solution for broadband connected set-top boxes. The set-top box should handle only video while a standard IP-telephone takes care of the audio. This separation between audio and video is mainly because it should be possible to have an audio conversation without any interaction from the set-top box or even when the box is turned off.

To perform this task there are a few questions that need to be answered:
- What different standards are used today in video telephony and which can be of use in this project?
- How can video telephony be implemented in a set-top box?
- What limitations in the implementation need to be done because of limited hardware?
- Can interoperability with other IP-telephony systems be achieved?
- How can video be made an optional part of a call?

1.3 Limitations
The final product is not supposed to be deployed to customers, but is intended only as an evaluation of what can be done with the currently available set-top boxes and give the set-top box designers hints to how future set-top boxes need to be designed to support video telephony.

Because of this, security aspects have not been taken into consideration, which is something that needs to be implemented before public use. Without any security it is possible to receive, hijack or terminate other people’s calls, for example.
1.4 Methods
This work has consisted of literature studies, practical evaluations and an implementation part.

The literature studies focused on the different IP-telephony and video compression standards available, in order to investigate which would be the most suitable for this project and its requirements. Available software for video telephony was also studied and compared to see which could possibly be of use and to get familiar with IP-telephony.

Before implementation could start it was necessary to evaluate different compression algorithms available to see how well they suit the specific hardware found in the set-top box. Should the hardware available not be enough to perform the video compression, the system would have to be designed with video compression handled by a separate unit. For these evaluations a series of benchmark tests were conducted.

The final step of this work was the implementation part where the theories and conclusions drawn in the previous parts were tested and the video telephony system was designed and constructed. Approximately half of the project time was spent implementing.

1.5 Sources
Of most use have been the various RFC documents covering SIP, SDP and RTP. Since these documents are considered the official standard documents for the mentioned protocols and have been accessed from the IETF website, these sources must be considered trustworthy.

Most documents covering video compression standards are somewhat aged and may not be up to date, but since standards do not change, the information available should be correct. It is possible that new information covering new compression techniques is missing, but I find it unlikely that the hardware used in this project would be capable of handling the new compression techniques and none of the existing video telephony applications seem to use them anyway.

Information about H.261 and H.263 has exclusively been gathered from the Internet, but is from lecture notes from universities, which I think would prove the information’s credibility.
1.6 Structure

First, important technologies used in this thesis work are described in chapters 2 through 6 in order to familiarize the reader with them. Chapters 7 and 8 describe various design problems with proposed solutions and are followed by chapter 9 describing the implementation of the video telephony system. Finally, chapter 10 concludes the whole project.

1.7 Glossary

ADSL
Asymmetric Digital Subscriber Line. “A data communications technology that enables faster data transmission over copper telephone lines than a conventional modem can provide.” (Wikipedia, 2004)

API
Application Programming Interface. A set of functions used to communicate between different software.

B2BUA
Back-to-back user agent. A SIP user agent monitoring calls between UAs. It is part of the call and can therefore modify and terminate sessions.

Callee
The user being called by the caller.

Caller
The user initiating a call.

CIF

Codec
COder and DECoder. A term used to describe a device or program capable of encoding and decoding an information stream. (Wikipedia, 2004)

DCT
Discrete Cosine Transformation. A mathematical transformation that transforms a signal into the frequency domain. A two-dimensional DCT is commonly used in video compression algorithms.

Dialog
“A dialog is a peer-to-peer SIP relationship between two UAs that persists for some time. A dialog is established by SIP messages, such as a 2xx response to an INVITE request.” (Rosenberg et al., 2002)
DNS
Domain Name System. DNS “/.../ is a core feature of the Internet. It is a distributed database that handles the mapping between host names (domain names), which are more convenient for humans, and the numerical IP address, which a computer can use directly.” (Wikipedia, 2004)

G.711
An audio compression standard commonly used in telephony applications. It defines the two compression algorithms μ-law and a-law. (Wikipedia, 2004)

G.723
An audio compression algorithm commonly used in telephony applications.

GUI
Graphical User Interface. An interface shown to the user, which the user can use to manipulate an application’s behaviour.

GNU
GNU’s Not Unix. A collection of programs and system tools which together with the Linux kernel form a Unix-like operating system. The software is free software released under any of the GPL or LGPL licenses. GNU can be found at http://www.gnu.org.

GOP
Group Of Pictures. In a video sequence consisting of intra and inter encoded frames a GOP consists of one intra frame and the inter frames preceding the next intra frame.

GPL
GNU General Public License. A license used for free software. In short it specifies that source code of a program must be publicly available and any derivatives of the work must also use GPL as license. The full license is available at http://www.gnu.org/licenses/gpl.html.

H.261
A video compression algorithm designed for ISDN networks.

H.263
A video compression algorithm based on H.261. It provides equal image quality to H.261, but at much lower bit rate.

H.323
A protocol presented by ITU defining a way to implement video conferencing and IP-telephony.

HTTP
HyperText Transfer Protocol. This application protocol is used to request information from the WWW (World Wide Web). It was designed by IETF and can be found in RFC 2068.
IETF
Internet Engineering Task Force. IETF is “/…/ charged with developing and promoting Internet standards. It is an open, all-volunteer organization, with no formal membership or membership requirements.” (Wikipedia, 2004)

ISDN
Integrated Services Digital Network. A digital telephone system, which can give users network access with speeds between 64 and 2048 kbps. (Wikipedia, 2004)

ITU
International Telecommunication Union. ITU “/…/ is an international organization established to standardise and regulate international radio and telecommunications.” (Wikipedia, 2004)

LGPL
GNU Lesser General Public License. A software license similar to GPL, but LGPL lets the user link program code to LGPL licensed code without using LGPL for the program itself. Because of this, LGPL is often used for software libraries. Any changes made to the LGPL code must however be made publicly available, just as for GPL. The full license is available at http://www.gnu.org/licenses/lgpl.html.

Linux
An operating system kernel. Among other things, it manages all running processes on the machine, includes drivers for hardware devices and provides a set of system calls, which processes can use to access hardware for example. To have a fully functional operating system the Linux kernel is often used together with programs from the GNU project.

Method
In SIP methods are used to specify requests. “The method is the primary function that a request is meant to invoke on a server. The method is carried in the request message itself. Example methods are INVITE and BYE.” (Rosenberg et al., 2002)

MPEG
Motion Picture Experts Group. “/…/ a small group charged with the development of video and audio encoding standards.” (Wikipedia, 2004) The compression algorithm MPEG-2 is used to store movies on DVDs and also to transmit TV digitally.

QCIF
Quarter Common Intermediate Format. An image format just as CIF, but only one fourth in size resulting in a resolution of 176x144 pixels. This is the recommended resolution for video telephony sessions.
RFC
Request For Comments. A series of documents released by IETF describing technologies and standards used on the Internet. SMTP, HTTP and SIP are examples of protocols defined in RFCs.

RTCP
Real-time Transfer Control Protocol. This protocol is used to control and monitor RTP data streams. See section 5.2 for more information.

RTP
Real-time Transfer Protocol. A protocol providing functions useful when transmitting real-time data. See section 5.1 for more information.

SDP
Session Description Protocol. SDP is defined in RFC 2327 and specifies a way to describe sessions. It can be used to describe sessions initiated by SIP. See also section 4.5.

SIP
Session Initiating Protocol. RFC 3261 describes the latest version of this protocol and it defines how sessions between users can be set up and torn down over the Internet. SIP can be used for video telephony/conferencing sessions. See chapter 4.

SMTP
Simple Mail Transport Protocol. The protocol defined in RFC 822. It defines the language that e-mail servers speak and how e-mail can be delivered to users.

Speex
An audio compression algorithm designed to be efficient and unencumbered by patents. (Wikipedia, 2004)

STB
Set-top box. A multimedia device connected to a user’s TV set that delivers digital TV transmissions, music, games and other multimedia services to the user.

Transaction
“/…/ a SIP transaction consists of a single request and any responses to that request, which include zero or more provisional responses and one or more final responses.” (Rosenberg et al., 2002)

UA
User Agent. In SIP a user agent is defined as “A logical entity that can act as both a user agent client and user agent server.” (Rosenberg et al., 2002).

UAC
User Agent Client. A SIP entity that generates SIP requests. The role of a UAC is specified for each transaction. (Rosenberg et al., 2002)
UAS
User Agent Server. A SIP entity that receives SIP requests and produces SIP responses. The role of a UAS is specified for each transaction. (Rosenberg et al., 2002)

UDP
User Datagram Protocol. A simple transport protocol used on top of IP (Internet protocol). UDP is connectionless and does not have error control, flow control or use retransmissions (Tanenbaum, 2002).

URI
Uniform Resource Identifier. “A URI is a short string of characters that conform to a certain syntax. The string indicates a name or address that can be used to refer to an abstract or physical resource.” (Wikipedia, 2004) In SIP a URI is used to describe a user’s identity.

USB
Universal Serial Bus. A high-speed serial data bus normally used to connect external devices to a PC.
2 IP based set-top boxes

A set-top box (STB) offers multimedia services to home users through their TV set. Examples of services are TV or radio transmissions, games, video on demand (the user can start watching movies or TV programs whenever he wants and not only on fixed times) and Internet web browsing. Some STBs only receive information that they decode, but the STB used in this project is broadband connected and can therefore transmit data, which allows for interactive services such as web browsing, chat and video on demand.

As mentioned, each STB is connected to a high-speed network, as can be seen in Figure 1, and servers in this network provide the STB with the information it wants, such as TV transmissions, movie streams, games and Internet services. To lower the demands on the network’s bandwidth somewhat, multicasting can be used to broadcast TV channels. By using multicasting each TV channel only needs to be sent out once no matter how many users that are watching the channel. Each channel has its own multicast address to which the STBs can listen. The STBs always have to notify the multicast server of their interest in a certain channel and should the multicast server notice that no one is listening to a certain multicast address it will stop the broadcasting and stop the bandwidth waste. (Tanenbaum, 2002)

2.1 Hardware

A set-top box is really nothing else than a very specialized computer. To keep cost down non-essential components have been removed and the hardware used is the simplest possible. A typical STB contains a CPU, RAM, a flash memory
to boot from and an MPEG decoder. The MPEG decoder is used to decompress digital TV transmissions, which are often broadcasted in MPEG-2 format. Since MPEG decompression is a computational intensive operation it is preferable to have a dedicated hardware decoder rather than a fast general purpose CPU doing the decoding. Then, the main CPU only has to manage the GUI, networking and other simple administrative tasks, which means that a very simple and cheap CPU can be used.

The STB used for this project uses AMD’s (Formerly National Semiconductor’s) Geode SC1200 CPU, which runs at 266 MHz, is x86-compatible and has integrated sound and video capabilities (Advanced Micro Devices, Inc., 2004). The MPEG decoder integrated in the STB is from the EM8400 series developed by Sigma Designs (Sigma Designs Inc., 2004). Since this STB gets all its information through a broadband connection it also has a high-speed network interface and a USB 1.0 interface is available to connect external devices.

Contrary to a computer, there is no keyboard or mouse, but all user input is normally handled through a remote control instead. Instead of a computer monitor a normal TV is used as display and unnecessary parts such as hard disk drives and CD players have been removed.

### 2.2 Software environment

As the STBs are miniature computers the same software that runs on computers runs on the STBs. It is important though that the software used is efficient and lightweight since the hardware resources in the STB are very limited.

The STB used in this project runs the operating system GNU/Linux, which can easily be customized to only include the parts necessary for the STB to do its work. This results in efficient use of the available hardware resources. GNU/Linux is also available for free, which helps keep licensing costs down to a minimum.

When developing software the limited hardware needs to be taken into consideration as well. To keep memory and CPU usage down, it is a good idea to try to keep the number of libraries used in a program to a minimum and also make sure that libraries of reasonable size and speed are used. Other than that, the development is just like developing software for PCs running GNU/Linux.
3 Video telephony

Video telephony is ordinary telephony, but with video added to it, which lets the users not only hear, but also see each other. A related term is video conferencing, which typically refers to audio and video sessions with more than two participants. In video conferencing there could be two groups of people using a single camera for each group or there could be a separate camera for each participant. To have compatibility between different video telephony or video conferencing systems it is important that standards are defined and used for communication.

When transmitting audio there are standards such as G.711, G.723.1 and Speex used. Each with its own advantages and disadvantages, of course. G.711 defines simple compression standards, which gives low coding latency, but requires much bandwidth compared to G.723.1 and Speex, which are good compressing codecs with higher latency and lower bit rate. Also audio quality differs between the different codecs although they all give quality equal to or better than normal telephone systems. Bit rates for audio are generally lower than 64 kbps and using Speex coding the bit rate can be as low as 2.15 kbps (Xiph.Org Foundation, 2003).

H.261 and H.263 are two of the most used video codecs. Because of the high bandwidth demands of uncompressed video, heavy compression is necessary before video can be transmitted over most networks. H.261 was constructed for use with ISDN networks, which have a capacity of 64 kbps up to 30*64 kbps. Because of this, H.261 is also called px64 where p is in the range 1-30 (Furht et al., 1995). H.263 can produce a video stream with similar quality to H.261 while using 2.5 times less bandwidth (Schaphorst, 1996). To have acceptable video quality, certain frame rates and image resolutions need to be met and since video telephony and video conferencing show different views of the users, the demands for each system are different.

In H.261 the image resolutions CIF and QCIF are defined and those are commonly used for video telephony and video conferencing purposes. CIF (Common Intermediate Format) is defined as the resolution 352x288, while QCIF (Quarter CIF) is a quarter of the size of a CIF frame resulting in the resolution 176x144. For video telephony a close up view of the user is used showing only the user’s head and shoulders. For this a QCIF resolution is adequate while for video conferencing, where a whole room of people needs to be seen, CIF is the proposed resolution (Schaphorst, 1996). Recommended frame rates used for the different sessions are 5-10 and 15-30 frames per second for video telephony and video conferencing respectively (Furht et al., 1995).
These frame rates, however, are only recommendations and to have high quality video a frame rate of 25-30 frames per second is needed.

For setting up and tearing down telephony sessions a few different networking protocols have been suggested. There are especially two protocols that compete to become the de facto standard for Internet telephony and these are H.323 and SIP. The two protocols are not compatible and even though they share some similarities they are in many ways each other’s opposite.

H.323 is the older of the two protocols and is therefore the one most used today. It was presented by ITU (International Telecommunications Union) in 1996 and is a very complete standard defining exactly how video telephony calls are handled, which ensures good interoperability between applications using H.323, but also restricts what can be done with the protocol. ITU did put a lot of features into H.323 from the start though with the hope that it would satisfy people’s needs for a long time. (Tanenbaum, 2002)

SIP, on the other hand, is a more lightweight protocol that tries not to be as strict as H.323, but more flexible. SIP was presented by IETF (The Internet Engineering Task Force), which has designed many of the protocols used throughout Internet such as HTTP and SMTP and ideas from these protocols were reused in SIP. Just like these protocols, SIP is a text-based protocol that is easily decoded and understood by humans, unlike H.323, which uses binary coding for its messages. SIP is also designed with Internet in mind, which makes SIP more Internet friendly than what H.323 is.

SIP’s flexibility comes from the fact that SIP only handles setting up, modifying and tearing down sessions. It does not define what kind of session it can handle, which means that SIP can be used for video telephony, but also for video game sessions, chat sessions or about anything. Due to this, different SIP implementations of video telephony may not be compatible. To help ensure compatibility between SIP implementations there are events arranged several times a year, which anyone with a working SIP implementation may attend to test his implementation against other implementations (Session Initiation Protocol Interoperability Tests, 2004).

For this project SIP is the protocol used and this is mainly because of its flexibility and simplicity. SIP also seems to have a bright future with many new SIP applications popping up on the market such as firewalls, proxy servers, gateways, hard phones and soft phones (telephony software for computers). An extensive list of SIP products can be found in iptel.org (2004).
4 SIP

All information in this section is based on Rosenberg et al. (2002) unless otherwise noted.

SIP uses a client-server model for its communications. A SIP client sends requests and the receiving SIP server generates a response to this request and sends it back. One request and its resulting responses is defined as one SIP transaction.

There are six different request types, called methods, used in SIP: INVITE, ACK, CANCEL, BYE, REGISTER and OPTIONS (Tanenbaum, 2002). This is the bare minimum of message types that a SIP implementation would have to implement and these are the only ones defined in the SIP standard. SIP is however not limited to only those methods, but implementers are free to use their own methods if needed. There also exist standardized extensions to SIP that contain additional methods. Examples of extensions are support for call-forwarding and presence notifications handled by the methods REFER and SUBSCRIBE respectively (Sparks, 2003; Roach, 2002).

The responses used by SIP to answer requests are similar to those defined in the well-known Internet protocol HTTP, which means that numerical values are used together with a human readable error description to describe the responses. The responses range between 100 and 699 and are grouped into six groups consisting of 100 responses each. Each one of these groups contains responses with similar meaning and the meaning of each response group can be seen in Table 1 below.

Table 1: SIP error code classification.

<table>
<thead>
<tr>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>100-199</td>
<td>Provisional responses.</td>
</tr>
<tr>
<td>200-299</td>
<td>Successful responses.</td>
</tr>
<tr>
<td>300-399</td>
<td>Redirection responses.</td>
</tr>
<tr>
<td>400-499</td>
<td>Client error responses.</td>
</tr>
<tr>
<td>500-599</td>
<td>Server error responses.</td>
</tr>
<tr>
<td>600-699</td>
<td>Global error responses.</td>
</tr>
</tbody>
</table>

Provisional responses tell the client that a final response cannot be generated right now, but will be sent at a later time. A typical provisional response is the “180 Ringing” response, which is used to inform the client that the server is ringing the user and we do not yet know whether the user will accept or decline the call. All responses other than provisional responses are called final responses.
Only one successful response is defined by SIP and that is “200 OK” and it simply means that the request received was accepted.

The 3xx series redirects the client to send the request elsewhere. This could be because the user to whom the request was meant has got a new address.

400, 500 and 600 series are used to signal an error somewhere. In case of a client error there is something wrong with the request, server errors mean the server was not capable of processing a request while global errors indicate that the request would have failed regardless of where it was sent. Examples from the three groups are “404 Not Found” used when a client tries to contact a non-existing user, “503 Service Unavailable” could be used when the server is overloaded or in maintenance and “603 Decline” tells the client that the requested user has declined the request.

The first response in each group (100, 200, 300 and so on) is a general response without any additional information added to it. For example 100 is specified as “Trying”, but no information is given about what really is tried and why a final response could not be generated immediately.
4.1 SIP operations

This section will describe what operations SIP can perform and what methods and responses are used to perform these operations.

4.1.1 Setting up a session

To initiate a session, SIP uses a three-way handshake consisting of an INVITE request, a response and an ACK request. Figure 2 shows a variation of this communication where a provisional response is used to tell User1 to hold for a final response. The provisional response is optional, but is often used when establishing telephone sessions. Once User2 has received the ACK a dialog is established between the two users.

The CANCEL method could be used by User1 to cancel the invitation before it gets a final response. So prior to receiving the “200 OK” response a CANCEL request could have been sent.

4.1.2 Modifying a session

Once a session has been set up, the users may want to renegotiate session parameters without terminating the session and reinitialize it. Modification is also accomplished by sending an INVITE request to the other user, but is called a re-INVITE since a dialog already has been established. Either of the two users may initiate the re-INVITE by sending an INVITE request to the other party containing the new information. Should the user that receives the re-invitation not accept the changes a “488 Not Acceptable Here” response is sent back and no changes to the session are made.
4.1.3 Terminating a session

Not surprisingly, the BYE method is used to terminate sessions and the signalling required is very simple as Figure 3 shows. Immediately, as a BYE request is sent, the sender considers the dialog closed, which means that the receiver cannot deny a session termination request. The receiver must send a “200 OK” response anyway to tell the sender of the BYE request that it has received the request and that it does not need to receive retransmissions of the request.

4.1.4 Location registration

For a user to receive any calls it is important that he announces his location to a registration service or it may be hard to locate the user. Locating users and registration will be described in sections 4.3 and 4.2.3 respectively. However, for the registration, the REGISTER method is used and sent to the registration server as seen in Figure 4. The registration server is called registrar. The REGISTER request is sent outside dialogs, which means that it is neither necessary to INVITE the registrar nor send a BYE request after the registration.

4.1.5 Capability discovery

In some cases it might be useful to know what the other end is capable of before a call is actually made. For example, if a user does not want to have a conversation with someone that lacks video capabilities, he does not want to bother the other end by calling and hanging up immediately anyway. For such situation it is possible to probe a user by sending an OPTIONS request. The recipient will process an OPTIONS request like an INVITE request would be processed. With the exception that no dialog will be generated and the other end will not be alerted of an incoming call.
4.2 SIP building blocks

To have SIP work well in a network there are many different elements that can be used. This section will describe the different types of SIP elements available and what their purpose is.

4.2.1 User agent

A User agent (UA) is the most common building block of a SIP network. User agents are the SIP elements between which sessions are established. SIP phones and SIP compatible instant messaging platforms are examples of UAs.

When talking about user agents it is common to distinguish between a user agent client (UAC) and a user agent server (UAS). The easy way to explain the differences is to say that UASes are user agents that receive requests while UACs are user agents sending requests. Within a dialog between two user agents it is possible that both agents will act as both UAC and UAS. For example, user A wants to make a call to user B. User A then sends an INVITE request to user B thus acting as a UAC. User B, on the other hand, receives this request and produces a response as a UAS. Once the users have had enough of each other, user B might terminate the session by sending a BYE request which means that this time user B is the UAC sending the request while user A is UAS and answers to the request.

4.2.2 Proxy

A proxy is a server element that receives SIP messages, checks their destinations and from certain predefined routing rules forwards the messages to another proxy closer to the final destination or directly to the intended recipient.

There are two main types of proxy servers: stateless and stateful.

Stateless

A stateless proxy is the simplest of the proxies. It does nothing else but what has been described above. It receives a message and forwards it to where it thinks it should go. It can generate error responses if it does not know how to forward a message or fails to do so.

Stateful

A stateful proxy understands the notion of transactions and once it receives a request from a UA it will handle the request as a UAS would do and also start a UAC instance, which forwards the message to another proxy or host. This means that the proxy itself manages retransmissions of requests inside its UAC instance while the UAS instance can generate provisional responses and error responses in case something goes wrong when sending the request (time-out, unknown destination, network error, for example).
A stateful proxy also has the possibility to fork requests, which means that it can send one incoming request to multiple destinations. This can be useful in cases where a user has several SIP devices which all have registered their different locations under the same name. When an INVITE comes for that user, all devices will ring and the user can use the device closest to him to answer the call. The proxy then CANCELS all the INVITEs sent to the other devices and only one session is set up.

4.2.3 Registrar
The registrar receives REGISTER requests and registers the user in its database. The entries in the database contain bindings between a user’s SIP address and the user’s contact information, i.e. host and port of the machine the user wishes to be contacted through. A registrar is often integrated in proxy servers. That way the proxy can use the registrar’s database to help in routing messages to the correct locations.

4.2.4 Back-to-back user agent
A back-to-back user agent (B2BUA) is quite similar to a proxy server, and is often mistaken as such. There is one very important difference though. The B2BUA controls and is part of the whole session while a proxy often only helps establishing the session and has no way of controlling the session, i.e. injecting SIP requests in the dialog.

A B2BUA consists of two UAs, which work together, back to back. One UA waits for incoming requests and the other UA is used to send requests. When a request is received the UA receiving the request will pretend to be the UA for which this request is meant while it tells the other UA to modify the same request and send it to the true recipient. The UA that forwards the request to the true recipient will however start a new SIP transaction with itself as the sender. This way the B2BUA effectively hides the true identities of the two UAs from each other and they are forced to talk to the B2BUA in the future. They think the B2BUA is the other user in the transaction/dialog. A proxy server does not generate a new request for an incoming request, but simply forwards the original. Figure 6 shows how two dialogs are used when the two users are communicating through a B2BUA while only one dialog is established when a proxy is involved.
A common use for a B2BUA is to control calls that are paid in advance. When a call is set up between users the B2BUA is contacted and acts as a middleman throughout the whole call. Should the caller exceed his credit the B2BUA sends a BYE request to each party and the session has been terminated.

### 4.3 Locating SIP users

Internet friendliness was mentioned earlier as one of SIP’s advantages and that is partly because of the addressing scheme used by SIP. SIP addresses are very similar to e-mail addresses in that they consist of a username followed by “@” and a domain or host name. What differs a SIP address from an e-mail address is that a SIP address contains the string “sip:” in the beginning. A typical SIP address may look like this: sip:robert@foobar.se.

The same approach used to locate e-mail recipients is used to locate SIP users when initiating a session. This means that when a user is to be contacted, the SIP server for the host or domain mentioned in the recipient’s SIP address is looked up through normal DNS queries and the request is sent there. Suppose a request would be sent to sip:robert@foobar.se. The request would then be sent to the SIP server for foobar.se and the proxy server of foobar.se would then try to locate user robert, by asking the registration server, and forward the request to his registered contact address.
4.4 SIP message structure
As stated earlier, SIP messages are human readable and a typical SIP message from sip:john@foobar.se inviting sip:robert@foobar.se to a session may look like this:

1) INVITE sip:robert@foobar.se SIP/2.0
2) Max-Forwards: 10
3) Via: SIP/2.0/UDP 192.168.1.193;branch=z9hG4bKb986.d9359434.0
4) Via: SIP/2.0/UDP 192.168.1.189:5060
5) From: sip:john@foobar.se;tag=785249902
6) To: <sip:robert@foobar.se>
7) Call-ID: 4052730853@192.168.1.189
8) CSeq: 1 INVITE
9) Contact: <sip:john@192.168.1.189:5060>
10) Content-Length: 250
11) Content-Type: application/sdp
12)
13) <Message body>

Here, line 1-11 constitute the main SIP message, line 12 marks the end of the SIP message and the beginning of the message body, which starts in line 13. The headers found in lines 2-11 are only a subset of all headers available in SIP and the ones listed are the most commonly used.

Let us go through this message line by line:

**INVITE sip:robert@foobar.se SIP/2.0**
This is the request line of the message. It specifies that this message is an invitation to a session for user robert at host/domain foobar.se. If this message had been a response, this request line would have been replaced with a status line containing the response code. All other lines of the message are common for both requests and responses.

**Max-Forwards: 10**
This is the first header of this SIP message. There is no special reason as to why this header is the first in the message since it does not matter in what order the different headers appear in the message. Max-Forwards limits the number of SIP elements that this message may pass. Each proxy that routes this message reduces this number by one and if the number reaches zero an error message is returned to the sender. This prevents the message from being caught in an infinite loop between badly configured proxy servers.
Via: SIP/2.0/UDP 192.168.1.193;branch=z9hG4bKb986.d9359434.0
and
Via: SIP/2.0/UDP 192.168.1.189:5060
The Via-headers document the route the message has taken from its origin. We here see that the message was originally sent by 192.168.1.189 and then it passed 192.168.1.193 before I caught it and put it in this report. The branch parameter identifies the transaction which this message is part of. The reason the first Via-header does not have a branch parameter is because the sender of the message conforms to an older SIP standard where this parameter was not mandatory.

From: sip:john@foobar.se;tag=785249902
The From-header shows the sender of the message. The tag parameter is used to identify which dialog this message is part of. The reason for why the tag parameters are needed and we cannot manage with only the Call-ID header (described below) to identify a message’s dialog is that some proxies are forking meaning that a single invitation with one Call-ID might give rise to several dialogs.

To: <sip:robert@foobar.se>
Of course this header shows whom the message is meant for. A tag parameter can be seen here too, but as this is an initial invitation we do not know the remote user’s tag so we let the recipient add it once it sends the response.

Call-ID: 4052730853@192.168.1.189
This header helps to identify which call this message belongs to. A unique Call-ID is generated for each new call that is made.

CSeq: 1 INVITE
CSeq stands for Command Sequence and is composed of a number and the method name of that transaction. The CSeq number is incremented for each new request made. By checking the CSeq of an incoming message it is possible to see if this is an old request arriving late and to which we already have answered or if it is an unanswered message.

Contact: <sip:john@192.168.1.189:5060>
Once a session has been established it is more convenient and efficient to communicate directly with the other party and bypass proxies along the way if possible. The contact header describes how the user can be contacted directly in the future. Some proxy servers may insist on remaining in the message path even after the session has been established though, but then it would have to insert a Record-Route header in the message before forwarding it.
**Content-Length: 250**
Tells the recipient the length of the attached body in bytes.

**Content-Type: application/sdp**
Tells the recipient what type of body is attached to this message. In this case there is an SDP body attached.

### 4.5 SDP

SDP stands for Session Description Protocol and is often used to describe sessions handled by SIP. Handley & Jacobson (1998) describes how SDP works and that SDP includes, among other things, information on what the session name is, who the session owner is, available medias and on which IP and port it expects these medias.

SDP messages are also human readable and an SDP message describing a session containing both audio and video can look like this:

```plaintext
v=0
o=john 1075375300 1075375300 IN IP4 192.168.1.193
s=A call
c=IN IP4 192.168.1.250
t=1075375300 1075378900
m=audio 42798 RTP/AVP 0 4
c=IN IP4 192.168.1.189
a=rtpmap:0 PCMU/8000
a=rtpmap:4 G723/8000
m=video 44466 RTP/AVP 31
a=rtpmap:31 H261/9000
```

Once again, let us have a closer look:

**v=0**
The value of v defines what version of SDP this message conforms to. 0 being the version number of SDP defined in RFC 2327 (Handley & Jacobson, 1998).

**o=john 1075375300 1075375300 IN IP4 192.168.1.193**
Here the origin of the session is defined. User john at host 192.168.1.193 generated this session. The fields that contain 1075375300 are session ID followed by session description version and they are used to identify this session and to check how recent a session description is.
s=A call
This is where the session’s name is set and in this case it is set to the extremely informative string “A call”.

c=IN IP4 192.168.1.250
This header holds the connection information. It defines which host that is listening for this session. Sometimes there are multiple media streams within a session that are supposed to go to different recipients. In such a case a c-header can be inserted directly into the media section (see below) for each media.

t=1075375300 1075378900
A session may only exist for a certain period in time. This is the place to define the time when a session exists. Values are given as decimal representation of Network Time Protocol time values, which is the number of seconds elapsed since January 1st 1900. First value sets the session start time while the second time defines the session’s end. This session lasts for an hour.

m=audio 42798 RTP/AVP 0 4
c=IN IP4 192.168.1.189
a=rtpmap:0 PCMU/8000
a=rtpmap:4 G723/8000
This is a media section. It describes a media that can be accepted and where this media is listened for. This media section describes an audio stream using either PCMU (µ-law as defined in G.711) or G.723 encoding and data is expected on port 42798 and delivered by the RTP protocol (section 5.1). Since there exists a c-header in this media section it will take precedence over the global c-header, which means that this audio stream is expected by host 192.168.1.189.

m=video 44466 RTP/AVP 31
a=rtpmap:31 H261/90000
Here is another media section, but this one describes a video stream of H.261 encoded video, which is expected on port 44466 of host 192.168.1.250 according to the global c-header.
4.5.1 Media negotiation

When two UAs wish to talk to each other it is quite essential that both UAs can hear and understand each other or there will not be much point in communicating. Because of this, an offer/answer model, defined in Rosenberg & Schulzrinne (2002), is used in SIP using SDP to figure out what common support for media and codecs there is for UAs in a dialog.

First step is generating an offer. This offer contains the medias that the UA supports and thinks should be suitable for the session. This offer is expressed as an SDP message and is included in the body of a SIP message. When the other end receives this offer it compares the suggested medias and codecs to its preferences and generates an answer describing what medias will be used in the session. The answer contains a copy of all the media sections received in the offer, but the medias that are not accepted by the answerer will have the port number set to zero while the other accepted medias have valid port numbers, of course. The example in Table 2 shows how one video stream and one audio stream is offered and the answerer chooses to accept audio using G.723 encoding while declining video.

<table>
<thead>
<tr>
<th>Offer:</th>
<th>Answer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>m=audio 42798 RTP/AVP 0 4</td>
<td>m=audio 27449 RTP/AVP 4</td>
</tr>
<tr>
<td>c=IN IP4 192.168.1.189</td>
<td>a=rtpmap:4 G723/8000</td>
</tr>
<tr>
<td>a=rtpmap:0 PCMU/8000</td>
<td>m=video 0 RTP/AVP 31</td>
</tr>
<tr>
<td>a=rtpmap:4 G723/8000</td>
<td>a=rtpmap:31 H261/90000</td>
</tr>
<tr>
<td>m=video 44466 RTP/AVP 31</td>
<td></td>
</tr>
</tbody>
</table>

The answer is put in an SDP message inside the body of a SIP message and sent back. If the medias suggested in the offer are not acceptable at all, a SIP error response is sent back. When a session is initiated with SDP bodies like this, SIP clearly defines in what messages it is allowed to put the offer and answer. There are two different scenarios: Either the UAC sends the offer and the UAS answers or the other way around, but in either case the offer and answer has to be part of the same INVITE transaction.

If the UAC sends the offer it must be included in the INVITE request and the UAS must send the answer in the following 2xx response (if the invitation is accepted, of course).
Should the UAC not wish to suggest an offer to the UAS in its INVITE request, the UAS may choose to do so in its 2xx response. If the UAS has sent an offer in the 2xx response the ACK, from the UAC, must include its answer.

It is possible to use this negotiation procedure both during the initial invitation to the session and also at a later point where a session is already established and a re-invitation is sent. This makes the sessions very flexible.
5 Media transport

When using SDP to describe what medias that are available in a session, only UDP and RTP are the protocols defined as possible media transport protocols (Handley & Jacobson, 1998). For this project RTP will be used to have interoperability with other implementations. RTP also adds sequence numbers and other useful headers to the video stream, which UDP does not. This makes RTP preferable to UDP.

5.1 RTP

This is the Real-time Transport Protocol, which is defined in Schulzrinne et al. (1996). It was designed for use with real time data streams such as audio and video, where low latency is more important than safe delivery of the data. As RTP is an application protocol, which normally is used on top of UDP, it cannot guarantee any quality-of-service. What RTP does is that it adds headers (that are useful for real-time data streams) to the real-time data before it is sent over the network as normal UDP packets. (Tanenbaum, 2002)

The headers that RTP adds help the receiver to reconstruct the original media stream. Since no quality-of-service is guaranteed, RTP packets may arrive delayed or in the wrong order at the receiver. Packages may also be lost along the way. To handle these problems, RTP adds a sequence number to each packet sent and this number is increased by one for each new packet, which makes it easy for the receiver to rearrange the incoming packets in order. A timestamp is also added in each RTP header. This timestamp tells the receiver which time instant this packet’s contents were sampled, which the receiver can use to play back the media in the right pace.

RTP can also be used to multiplex several media streams such as audio and video into one single stream of UDP packets.

5.2 RTCP

RTCP is used in conjunction with RTP to monitor RTP transmissions. Its main purpose is to send reports on how well RTP streams are received and sent from and to the network. If the network’s bandwidth would decrease, for example, RTCP would send a report telling about the increased packet loss/delay, which the receiver could use to adapt its media stream to the new bandwidth. Possible countermeasures to the bandwidth drop are increasing the compression of the media and lowering the sample or frame rate.
6 Image processing

6.1 Colour spaces

Colour spaces are used to express colours in computer applications. Using only three numbers it is possible to express every single colour in the world. These numbers form a three-dimensional colour space containing all the colours (Travis, 1991). There exist many different colour spaces, each with its own advantages and disadvantages, but only two colour spaces often used in computer graphics and image compression will be described in this section.

6.1.1 RGB

In the RGB colour space the colours red, green and blue are used to define colours. A colour monitor or TV uses these three colours when rendering a picture. To paint a pixel black all three colour components should be turned off while turning all components up to max paints the pixel white. The dashed line in Figure 7 shows where grey colours are found. (Travis, 1991)

6.1.2 YUV

Another popular colour space is the YUV colour space. It is different from the RGB colour space in that it does not work with colours but intensities. The Y stands for luminance and defines how bright a pixel is while U and V specify the chrominance, which holds the colour information. YUV originates from the time when colour TV made its entrance into the world. Before colour TV, only the Y-data was transmitted to the users’ TVs. Instead of transmitting completely different data at a different frequency to the colour TV users, the engineers tried to just add the colour information to the Y-data. What they came up with was the chrominance levels found in the U and V components. Because of this, it is possible even today to use a black and white TV to watch TV transmissions. Those TVs simply do not know of the colour data and ignore it. (Tanenbaum, 2002)

The YUV colour space is advantageous even outside the TV world. The human eye is more sensitive to changes in the luminance than in the chrominance components and this means that colour information can be removed without the viewer noticing any difference, which leads to smaller image sizes making it ideal for image compression. YUV data also compresses better than RGB normally does.

Conversion between RGB and YUV is a lossless operation, meaning that no picture information is lost.
There are many different flavours of YUV, which only differ in how much and which of the colour information is removed and how the different Y, U and V components are stored in memory. To show how the formats differ, two of the YUV formats are described below.

**YUV444**
No colour is removed in this format, which means that for every pixel there is one Y, one U and one V sample.

**YUV420**
This is the most used YUV format for image compression. It only contains a quarter of the colour information compared to a YUV444 coded image, which means that even before the compression the image size is only 50% of the original. Figure 8 shows a 4x4 pixel image encoded in YUV420 format and the colour information available in it. As can be seen each pixel has its own Y sample, while four neighbouring pixels share the same U and V components. The colour information in U₁ and V₁ belongs to the Y values of Y₁, Y₂, Y₅ and Y₆. (Schimek & Dirks, 2003)

### Figure 8: Colour information in a YUV420 encoded picture.

#### 6.2 Video Compression
An image of CIF resolution with 24 bits colour information/pixel requires 352*288*24/8 = 304,128 bytes for storage. A video telephony session using the lowest recommended frame rate of 5 frames per second and CIF sized images would require a network capable of handling 5*304,128 = 1,520,640 bytes each second. Since this amount of bandwidth is not acceptable for video telephony sessions, compression is necessary. Most broadband connections offered these days could not even handle the bandwidth required and especially not asymmetric connections such as ADSL where the available bandwidth for sending data is usually much lower than the bandwidth available for receiving data.

Most compression schemes that are used for video today are lossy, meaning that image quality is lost in the compression process. This is however often acceptable since the achieved compression ratio greatly exceeds the ratio of lossless compression algorithms and in many cases the lost image quality is not noticeable.

There are many different approaches to video compression, but this section will describe how the most common compression algorithms used in video telephony work. For video telephony H.261 and H.263 are the most common algorithms.
since they have the ability to produce very low bit rate streams. Even though MPEG-1 and MPEG-2 are not common compression algorithms in video telephony applications the following compression method is applicable to those algorithms as well.

Image data is expected in YUV format for this compression technique and compression is divided into two parts; intra and inter encoding. Intra encoding compresses a complete frame while inter encoding means that only the changes from the previous frame are compressed and included in the resulting data. Since consecutive frames in a video telephony session are similar, inter encoding gives a drastically decreased bit rate. Ideally, it would be enough to send one intra encoded frame first in the video session and only use inter encoded frames for the rest of the frames. This would however give problems if packets are lost or errors appear in the data stream. Errors would propagate from frame to frame and they might never disappear. To ensure that the receiving side has a proper image, intra frames are sent every now and then as Figure 9 shows. The more inter frames used the higher compression ratio is achieved, but on the other hand the image quality may degrade and CPU utilization rise. The number of inter frames inserted between intra frames is called Group Of Pictures (GOP) size.

For both intra and inter encoding the frame is divided into blocks prior to the compression process. This is done for the Y, U and V data separately and normally blocks of size 8x8 samples are used. Figure 10 shows a typical block and each pixel’s value. This block will be used to exemplify the different steps in the intra encoding section below.

### 6.3 Intra encoding

Intra encoding compresses a complete frame without any references to previous frames. The algorithm used here is also often used when compressing still images. In fact, the steps used in the intra coding can also be found in JPEG encoding.
6.3.1 Discrete Cosine Transform (DCT)

For each block a discrete cosine transform is performed, which puts the samples in the frequency domain. Since the spatial frequency of a block normally is low (Colour changes are smooth and rarely abrupt.) this transformation puts most of the coefficients close to the DC component, which has frequency zero and is an average value of the whole block. The transformation is theoretically lossless, but rounding errors may occur due to the resulting floating-point numbers from the transformation. (Tanenbaum, 2002)

![Figure 11: DCT coefficients. The DC component is located in the upper left corner and frequencies increase towards the lower right corner.](image)

6.3.2 Quantization

This is the step in the compression where details disappear from the image and thus makes the algorithm lossy. Each of the values in the DCT matrix is divided by a pre specified value, the quantization value, resulting in some coefficients getting rounded off to zero. Using a higher quantization value gives more zero-valued coefficients, which in turn gives higher compression ratio at the expense of worse image quality. In Figure 12 a quantization value of 10 has been used for all coefficients, but some compression algorithms use a separate quantization value for each coefficient. This can be used to suppress high frequencies more than low frequencies. (Tanenbaum, 2002)

![Figure 12: DCT coefficients after quantization.](image)

6.3.3 Run-length encoding

At this stage most DCT coefficients are located in one corner of the matrix while most other coefficients are zero-valued. Using a zigzag pattern when scanning the values, as shown in Figure 13, it is possible to gather most of the zeroes in the end of the number sequence. Applying run-length encoding to the resulting number sequence, it is possible to achieve efficient compression. (Tanenbaum, 2002)
Scanning the quantized DCT coefficients from previous section using the zigzag pattern the following sequence is obtained: 146, -3, -4, -2, -5, -2, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.

Run-length encoding reduces this sequence to something along the lines of: 146, -3, -4, -2, -5, -2, 7 zeroes, -1, 4 zeroes, -1, 45 zeroes, which is more efficient than expressing every single value separately.

### 6.3.4 Huffman coding

As a final step, Huffman coding is applied which reduces the number of bits used to encode frequently appearing values while uncommon values get a longer bit-string. (Tanenbaum, 2002)

### 6.4 Inter encoding

Consecutive frames in a video stream often have many similarities. Looking at a typical video telephony session, where one has a head-and-shoulders view of the other party, there is often quite little motion. Facial expressions and head movements are the main changes between frames while the background remains unchanged. This can be taken advantage of to achieve high compression since only blocks that have changed need to be encoded and compressed. These frames, containing only information about changes since the previous frame, are called inter frames.

Two different approaches to inter frame coding will be described here: conditional replenishment and motion compensation.

#### 6.4.1 Conditional replenishment

For conditional replenishment every block in the frame is compared to the corresponding block in previous frame and only if a significant change is detected that block is compressed. Conditional replenishment is easy to implement and the speed is high compared to motion compensation described below, but the resulting bit rate is higher than for motion compensation.
6.4.2 Motion compensation

Objects moving around in the picture cause many blocks to change between frames, which in turn triggers re-encoding of all those blocks. It would be more efficient if we could detect an object’s movement and simply tell the receiver how the object has moved relative to the previous frame. This technique is called motion compensation and is frequently used in video compression.

When it is detected that a block has changed since the previous frame, the previous frame is searched for patterns found in the present block that is to be encoded. If a suitable match is found in the previous frame a motion vector will be used to describe how the pattern has moved compared to the prior frame. These steps are described in Figure 14 below. Since it is probably impossible to find an exact match, the differences between the two blocks are compressed and transmitted along with the motion vector to the recipient. By compressing only differences, less compressed data is produced than what would have been the case if the block had been compressed without any reference to a previous frame. (Video compression, 2000)

![Figure 14: Simplified view of motion compensation showing how motion vectors are used to express a block’s movement.](image)

Searching a picture for specific patterns can be a very time consuming task and can be done in many different ways. Best compression would of course be achieved if the whole frame is searched for matches, but since high compression speed often is desired searches are often performed in a limited area close to the current block’s position. Exactly how the search is performed is up to the implementer to decide.

Decoding motion vectors is on the other hand a simple task. The fact that compression takes much longer time than the decompression makes inter frame encoding using motion compensation an asymmetric task.
7 Project requirements

Before starting the system design it is necessary to define what the final system should be capable of and what it is expected to do.

The following list of primary requirements was constructed:

- It should be possible to have audio and video communication between two STBs with an acceptable quality.
- SIP should be used for session management, if possible, since it would give great flexibility, be future proof and allow for added services such as chat, presence notifications and other nifty features.
- To take advantage of the high availability of cheap USB cameras in the market, a regular USB 1.0 compatible camera should be used to deliver the video to the STB. USB is also a convenient way to attach a camera to the STB. The alternative would be to access the camera through the network.
- Audio is handled outside of the STB in a standalone IP-telephone. This enables users to make audio calls without being dependent of the STB. The STB can be turned off.
- The user should be presented with the possibility to add video to an initiated audio call.
- Free, open source software should be used when possible to keep licensing costs down.
- Existing standards for video telephony should be used as much as possible to make interoperability with existing video telephony systems possible.

Additionally, if technically possible and time allows, the following functionality should be implemented:

- It should be possible to add and remove video to and from an initiated audio session with great flexibility. Video could, for example, be added several times during a call, but only when needed/desired.
- The system should be compatible with other video telephony systems.
- The user should be able to initiate calls from the STB by selecting users from a graphical phone book interface.
- Other sessions should be possible to add to the audio/video sessions. Chat, game or whiteboard sessions are examples.
8 System design

8.1 SIP communication

It was decided at an early stage that a separate SIP phone should be used to handle the audio. This has the advantage that it is possible to use the phone without any STB involvement and even when the STB is turned off. This is especially important for emergency calls, which always have to work. It also eases some parts of the implementation since no audio compression or echo cancellation needs to be implemented, for example. However, this also introduces problems.

When a call is made from the audio UA we would like the ability to automatically add video to the session using the STB. For this to work there has to be some kind of connection between the audio and video UA. We cannot modify the audio UA to have it report to the video UA whenever it makes a call, but since all outgoing calls pass through a proxy server it is possible to add some intelligence to the proxy, which handles the STB notification.

In an addition to the SIP standard the method REFER is defined. REFER can be used to have one user initiate a call between two other users. It is called call-transfer. If a proxy would detect an audio call being initiated it could send a REFER request to the corresponding STB, asking it to call up the other STB. This would get the effect that we wanted: an audio call would also start a video session. Figure 15 shows how REFER is used for this scenario. Note that a REFER request must be sent inside a dialog so a dialog between the proxy and STB

Unfortunately, this only solves part of the problem. In this case there are two SIP sessions established between two different pairs of UAs, which means that there is still no connection between the audio and video parts. Using the proxy to terminate the video session in a similar way as it was established is, however, not possible. Only STB\textsubscript{1} or STB\textsubscript{2} may terminate the session since they are the dialog endpoints.

Also, this would only work well in a homogenous world where every user has an STB and a separate audio SIP phone. If a UA with both audio and video support would have been dialled there would not be an STB corresponding to that UA to call and so video would not be possible for that call.
The solution to our problems is a Back-to-back UA (B2BUA), which has the ability to modify sessions between UAs and also tear them down. A B2BUA also effectively hides the real identities of the involved UAs, which can be used to merge two UAs into one entity and also split one UA into two. This really helps interoperability since an audio and video capable UA could be fooled into believing that it has a session with a single UA while in fact the B2BUA has merged the audio UA and video UA into one single entity as shown in Figure 16.

Normaly, a B2BUA only manages two sessions, one to each of the involved UAs. However, in the case where STBs handle video and audio UAs handle audio, the B2BUA needs to be able to handle four simultaneous sessions. For this reason the required B2BUA consists of four UAs as shown in Figure 17. Of
course, should either the caller or callee use a UA capable of both audio and video, its corresponding Video UAC would be unused for that call.

For the B2BUA to work in all situations it must be able to determine the capabilities of the parties involved in the call. This can easily be accomplished by checking the SDP bodies supplied when a session is initiated. Re-INVITEs can also be sent to discover capabilities even after a session has been established. In the cases where the caller and callee have the same capabilities it is easy for the B2BUA to just forward the SDP bodies between the two users. When either side uses a UA capable of both audio and video, the B2BUA has to merge and split SDP bodies.

8.2 SIP implementations

Even though SIP is a relatively small standard compared to H.323 it contains a lot of rules that have to be obeyed. This makes implementing SIP from scratch a huge undertaking, but luckily there exist libraries to help in developing SIP applications.

To keep licensing costs down, free SIP implementations are preferable and the free SIP stacks that were found are oSIP (The GNU oSIP library, 2004), dissipate (DiSSiPaTe, 2004), osipua (Linphone Telephony on Linux, 2004), Vovida (Vovida.org, 2004) and eXosip (The eXtended osip library, 2004). Both dissipate and eXosip uses GPL, which makes it impossible to use those SIP implementations in a commercial environment with closed source. This leaves us with Vovida, osipua and oSIP.

Vovida SIP stack is a huge implementation primarily intended for SIP server implementations. This makes it unsuitable for embedded systems such as STBs since the binaries produced are several tens of megabytes in size.

Osipua is a library built on top of oSIP in order to provide the developer with a higher level of abstraction. Osipua is part of the SIP softphone Linphone, but is no longer in development as it will be replaced by eXosip (Morlat, 2003).
oSIP is a low level SIP stack giving the user great flexibility, but also requires a lot of development effort to get a working UA. oSIP is currently under active development, some documentation is available and support is available through a mailing list. oSIP is also independent of other libraries with the C-library as its only dependency making it small.

Concluding the SIP implementation evaluation, the two main candidates are osipua and oSIP. Since oSIP is in active development and there is support available it comes out as the better SIP implementation of the two, although it requires much development effort.

8.3 Video input
For the STB to get video capabilities, a USB camera is supposed to be used as stated in the initial requirements. The camera also has to fulfil the following requirements:

- Linux compatible
- Acceptable image quality and frame rate
- Low CPU utilization

Finding a USB connected camera is not a problem these days since most cameras sold for PCs use the USB interface. The main problem is finding one that works with Linux. To my knowledge there is not a single camera manufacturer that develops Linux drivers for their devices, but fortunately this does not mean that there are no cameras that work with Linux. Skilled programmers have in their spare time been working on getting their cameras to work with Linux and in many cases they have succeeded and made their work available to the public. Sadly, it takes some time for the drivers to be developed, meaning that most new cameras released to the market have no Linux support and once the drivers have been developed it could well happen that that camera model is no longer sold.

One problem with USB 1.0, which most cameras and the STB uses, is that it has a signalling rate of only 12 Mbps, which is a very real limitation in the world of video where high data rates are common and often required. The limited data rate of USB sets limits on what resolution and frame rate combinations can be used. To overcome these limitations some camera manufacturers compress the video data before it is sent over the USB and the camera driver decompresses the video stream once it has passed the USB. The decompression, of course, has to be performed by the main CPU raising the CPU utilization, which we had hoped to keep low.
Since we also plan on compressing the video data it would be preferable to have image data delivered by the camera in YUV format and thus removing the need for a colour space conversion before compression.

A significant difference between camera models is the list of supported image resolutions. The more expensive models support resolutions of 640x480, but even the cheapest support CIF resolution, which is enough for our application.

A camera giving CIF sized video in uncompressed YUV format would be ideal. The USB bandwidth would then limit the frame rate to approximately ten, which is good enough. However, no camera with the above specifications was available for purchase so a compromise was needed and a camera giving uncompressed RGB data was chosen. This means an extra colour space conversion needs to be performed by the CPU.

### 8.4 Video compression

The STBs were not designed with video telephony in mind and therefore there is no hardware support for video compression and the main CPU has very limited computational power. To see what the STB is capable of, a series of performance tests were performed evaluating several video compression algorithms. Should the CPU not be able to handle compression or decompression, those tasks would have to be moved to a server.

The different compression algorithms tested were H.261, H.263, H.263+ and MPEG-1. The H.26x algorithms are designed specifically for video telephony with low bit-rate as the primary goal and they are the most commonly used algorithms for video telephony. MPEG-1 was included in the test because there is an MPEG-decoder built into the STB, which could be used to free the main CPU from video decompression. Using MPEG-1 would hurt interoperability with existing video UAs though.

#### 8.4.1 Codec implementations

There are several different implementations available of the video compression algorithms mentioned above. Many of them share the same origin though, which means that fundamentally they are the same.

For H.261 there is really only one implementation available and it has its roots in VIC (UCB/LBNL Video Conferencing Tool (vic), 2004), a video conferencing application. The version evaluated, however, was extracted from the open source project OpenH323 (OpenH323 Project, 2004). The decoder included in this implementation supports both intra and inter frames making it fully compatible with other H.261 implementations. The encoder, on the other hand, only sends inter frames. It will use a conditional replenishment algorithm
to detect blocks that have changed since last frame and encode these blocks using intra encoding. Intra coding is used on block level and not frame level, which reduces the risk of distorted images and lowers bit rate. The produced H.261 stream is compliant with the standard (intra frames and motion compensation are optional), but since no motion compensation is used a higher bit rate can be expected. This H.261 implementation has been around for a long time and is known to work well with other H.261 implementations such as Microsoft’s NetMeeting (Microsoft Corporation, 2004b).

There exist a few different H.263 implementations and two were evaluated for this project.

OpenH323 contains also an H.263 codec, although the implementation is rather limited. The decoder does not support motion vectors, so it will not work well and produce distorted pictures with other implementations that make use of motion vectors. The encoder can encode intra and inter frames, but the inter frame encoding will not make use of motion vectors, but conditional replenishment, which probably results in a higher bit rate. This H.263 implementation is known to work with some other implementations, but not with all.

FFMPEG (FFMPEG, 2004) is a library containing a large number of video codecs. From this library H.263, H.263+ and MPEG-1 implementations were evaluated. All these implementations support both encoding and decoding intra and inter frames (with motion compensation), but unfortunately they all have interoperability issues. The implementations of H.263 and H.263+ do not produce frames suitable to embed in RTP packets while MPEG-1 simply is not a standard video telephony codec.

8.4.2 Codec benchmarks
To compare the different compression algorithms a small program was put together. The program compresses and decompresses the same video sequence over and over again, but with different algorithms and compression settings and outputs performance statistics. Valuable performance measurements that were collected for each algorithm were the time spent compressing and decompressing frames and also the amount of compressed data produced.

To get a fair comparison of the different codecs the same video sequence was used for all the tests. For that reason a one-minute video was recorded as reference video. As it is interesting to know how a codec would perform in a video telephony application the recorded video tries to imitate such a session, but to also test extreme situations the video contained scenes with heavy and almost no motion.
In a real video telephony application there are several tasks other than the video compression/decompression that take significant time such as:

- Capturing images from a camera.
- Colour space conversion – All compression algorithms used expect YUV420 data. Should the camera not deliver images in this format a conversion is necessary.
- Rendering the decompressed image on the TV screen.

To have the benchmark results resemble reality as much as possible it is important to take these operations into consideration when calculating frame rates. Each of these operations takes a constant time to perform for a given resolution and the times were measured separately prior to the compression benchmarks.

Values that were varied during the tests were the resolution, quantization value and GOP size. The resolutions used during the tests were CIF and QCIF since they were the only resolutions supported by all codecs and as stated earlier these are the resolutions suggested for video conferencing and video telephony. The quantization value was varied between 1-30, which is the range used by all codecs. Intra frames were expected to be faster to compress than inter frames because of the lack of motion vector search. To verify this, GOP sizes of zero and five were used in these tests. Also, we cannot control what type of frames we receive (could be only intra or contain inter frames), which means that it is important to check that the decompression of inter frames can be done in reasonable time.

When estimating the quality of the decompressed video the following grading was used:

5 – Perfect quality. There are no visible differences between the uncompressed video and the compressed video.
4 – Acceptable quality. Blocks from the compression are noticeable, but not distracting.
3 – Mediocre quality. Blocks are easily noticed and distracting, but image is still good enough to make out details such as eye movements.
2 – Bad quality. Details seen in pictures of grade three are lost and the image is really blurry.
1 – Terrible quality. Even large details are now hard to see.
0 – No picture. The compression/decompression was unable to handle the supplied settings.

Only grades 4 and 5 were considered useable for video telephony.

Results from the benchmark runs can be found in Appendix 1.
8.4.3 Benchmark results

Common observations
CIF is the recommended resolution for group conferences while QCIF is considered to be sufficient for video telephony where normally only head and shoulders of the user are shown (Schaphorst, 1996). This was confirmed during these tests, which used a typical video telephony session as reference. The added detail when using the higher resolution CIF was actually unnecessary. On the other hand, the computational power to handle CIF images is four times that of QCIF handling, which would suggest that it is preferable to use QCIF resolution and high quality compression instead of CIF with lower quality compression to get an acceptable frame rate and picture quality in video telephony sessions where computational power is limited.

OpenH323’s H.261
This is the fastest algorithm, but it is also the one that gives the worst picture quality of all the compression algorithms: blocks in the picture are noticeable even at the highest quality settings. The compression time is more or less constant regardless of the quality setting and it is mainly the decompression time that is affected by the changed quantization value, which is the expected behaviour. It is mainly the DCT operations that take time to perform in the compression and decompression operations. When compressing a frame the DCT is performed prior to the quantization and is thus not affected by the quality setting. When decompressing a frame the quantization during compression has removed some of the DCT coefficients making the reverse DCT operation quicker when low quality is used.

The worst-case scenario for the encoder is when there is a lot of motion since it then has to re-encode every single block in the frame. Benchmarks show that frame rate may very well be cut in half if there is heavy motion compared to “normal” motion.

Worth noting is that because the decoder was decoding frames encoded by the encoder that only produces inter frames, the benchmarks for the decoder do not show how well it handles intra frames and motion vectors. Also, since this implementation does not use intra frames a GOP size setting is not applicable for this implementation.

OpenH323’s H.263
This is the slowest algorithm in the tests and it is mainly because of the compression, which takes a lot of time. Especially inter frames seem hard to encode and since the bit rate is really high if only intra frames are used, it is hard
to find a good use for this algorithm unless lots of spare bandwidth or processor cycles is available.

Quality is fairly good with this implementation if a low quantization value is chosen, but for some reason part of the picture seems to be decompressed later than the rest of the picture, which means that this part is lagging behind the rest of the picture.

**FFMPEG’s H.263 and H.263+**
The results for FFMPEG’s H.263 and H.263+ implementations are more or less identical. The major difference between these codecs is that H.263+ supports any resolution between 4x4-1152x2048 while H.263 is locked to just a few resolutions (Pelletier, 2000).

These implementations are faster than OpenH323’s H.263 implementation, but they are not as fast as the H.261 implementation making them unsuitable for CIF resolution. Speed is however relatively independent of the level of motion, which makes these algorithms predictable. Even in these implementations inter frame encoding takes more time than intra encoding, but for some unclear reason the bit rate remains the same regardless of the GOP size.

Worth mentioning is that the picture quality is perfect for any quality setting even though the compression and decompression times decrease slightly with lower quality demands.

**FFMPEG’s MPEG-1**
Both compression and decompression times were measured for this implementation, but since the STBs have a built in hardware MPEG decoder only the compression times were included when calculating frame rates (decompression was expected to take no time at all). Encoding speed is really good and not too far behind the H.261 encoder. In fact, it seems to handle heavy motion better than the H.261 encoder. Really interesting is the fact that using inter frames does not affect the encoding speed much at all while the decompression time is cut in half!

Picture quality is perfect and the bit rates are competitive with the H.26x implementations if inter encoding is used, which was not expected.

**8.4.4 Benchmark conclusions**
Since interoperability with other video telephony systems is desired there is really only one codec to choose and that is the H.261 implementation. Since this codec also has good performance and can produce good quality images (with low quantization values) this codec is the overall “winner” in these tests.
9 Implementation

Two distinct components were developed in this project. One part was the UA running on the set-top box that handled the video and the other part was a B2BUA used to merge and split SIP sessions. The B2BUA could be integrated into the STBs, but was for this project a separate unit running on a normal Linux PC. This was because audio calls would otherwise be dependent on the STB, which was something we tried to avoid by using a separate audio phone. This means that the network layout we first saw in section 2 needs to be extended with two new servers: a SIP proxy/registrar and the B2BUA as Figure 18 shows.

![Network layout with video telephony offered for the STB households.](image)

Figure 18: Network layout with video telephony offered for the STB households.
Clip art images copyright Microsoft Corporation. Used with permission.

9.1 Video user agent

9.1.1 SIP

The implemented video UA cannot initiate calls itself, but is merely called up by the B2BUA when an audio call is outgoing or a video call is incoming. To be able to receive calls the video UA has to register its location though and this is the first thing the UA does when it starts up. After this it just waits for incoming calls.

When a call is received it sends a provisional response (180 Ringing) to the B2BUA telling it that it is notifying the user of the incoming call at the same time as it also starts to flash the TV screen in order to attract the user’s attention. Having an audio signal to alert the user is not really necessary since that is handled by the audio UA instead.
The user is presented with two options. Either he may choose to decline the video session or he may accept it. Is the session declined nothing happens but the screen stops to flash and the B2BUA is told about the declination by a “603 Declined” response. After the declination it is not possible to add video to the session again. Should the user however accept the video session a “200 OK” response is sent to the B2BUA and the UA starts to exchange video with the opposite side as is described in the following video handling section. For simplicity, video is exchanged throughout the whole call and should the user tell the video UA to terminate the video session it will cause the B2BUA to drop the audio session as well.

All SIP communication done by the video UA is handled through the oSIP library.

**9.1.2 Video handling**

There are many different actions that have to be performed from the instant of capturing a single image from the camera until it can be viewed on the receiver’s TV. Here is an overview of the whole system including both the transmitting and the receiving end.

![Video system overview](Figure 19)

Clip art images copyright Microsoft Corporation. Used with permission.

As the image compression/decompression and RTP packing/unpacking operations are similar but only reversed they will only be described once.

**9.1.3 Image capturing**

To access imaging devices in Linux the Video for Linux API (Video4Linux HQ, 2004) is used. It lets you query device capabilities and set your own image
preferences before capturing starts. Frame rate, image resolution, colour space, brightness, contrast and hue are examples of properties that can be set for a device. Once device properties have been set Video for Linux is also used to fetch frames from the device of the specified format.

The camera chosen was the “Labtec webcam” produced by Labtec (Labtec.com, 2004). This camera is a cheap and simple model and has acceptable support in Linux. Only resolutions up to CIF are supported and a colour depth of only 8 bits per pixel is used. Video data is transferred uncompressed over USB and therefore limits the frame rate to roughly eight frames per second. This camera is however good enough for a basic video telephony application.

The most noticeable disadvantages of this camera and its driver are:

- The frame rate is locked to eight frames per second. It is not possible to change this, which would be preferable to do if video compression cannot keep up with this frame rate.
- If a resolution lower than CIF is chosen the camera driver will not deliver a scaled version of the CIF sized frame, but instead crop the image resulting in narrower field of vision.
- RGB is the only colour space supported, which is not suitable for video compression.

The Labtec camera delivers eight frames per second and since H.261 (the compression algorithm we will choose in section 9.1.5) can handle that frame rate at both QCIF and CIF resolution there is little point in capturing QCIF sized frames from the camera. In situations where the camera is located far away from the user, which might be the case if the camera is positioned on the TV for example, it might actually be preferable to use QCIF resolution since there would then be a kind of zoom effect when the camera driver crops the image to QCIF resolution. That would however be camera dependent (not all camera drivers crop) so in order to have good image quality, no matter what camera that is used, CIF is the frame resolution used to capture frames from the camera.

9.1.4 Colour space conversion

As stated in the previous section, the camera cannot deliver frames of YUV format, which means that a colour space conversion is required before the video compression can start. The FFMPEG library contains functions to perform colour space conversions and it easily handles the RGB to YUV420 conversion.
9.1.5 Video compression

H.261 from OpenH323 was chosen for the video compression. This was mainly because it was the only implementation that is known to work well with other implementations and it was the fastest of all evaluated codecs. To have acceptable image quality, a quantization value of five was set giving relatively little compression and a bit rate of around 200 kbps at eight frames per second. Since all STBs are connected to broadband connections capable of digital TV transmissions of up to 5 Mbps a bit rate of 200 kbps should not be a problem. There might be a problem if the other user does not have a broadband connection, but that would have been the case even with maximum compression.

9.1.6 RTP packaging

Turletti & Huitema (1996) defines the procedure of packaging H.261 packets into RTP packets. To make use of the nice features of RTP every H.261 packet produced by the encoder was put into RTP packets before being sent over the network. This way the receiver can easily decode the video stream and play it back properly.

The receiver implemented in the video UA does not use much of the RTP header data when decoding the stream though. It assumes that the packets it receives are in order and that there is little delay. Because of this there is no packet buffer in the receiver, which could be used when reordering packets or inserting delayed packets before the actual playback starts. The receiver decodes each packet as soon as it arrives and updates the TV screen once it detects that it has received a complete frame. Each frame boundary is marked in the RTP header.

Support for RTCP was not implemented neither in the sender nor receiver, meaning that no feedback on the video stream’s condition is possible in either direction.

9.1.7 Video rendering

When an H.261 packet has been decompressed at the receiver’s side it is time to render the frame on the TV screen. At this stage it is finally possible to make use of hardware built into the set-top box. The integrated MPEG decoder has the ability to read uncompressed YUV420 frames directly from memory and display them on the TV. All that is needed is to write the decoded data to a specified memory area and tell the chip the size of the image stored there and it will automatically render the image on the TV. Another nice feature of the MPEG decoder is that it can rescale video data to any resolution in hardware. This makes it possible to zoom the video stream to cover a large part of the TV screen without any loss in performance. Of course both QCIF and CIF sized frames can be zoomed to cover a large part of the screen.
9.2 B2BUA

Modifying an existing proxy server to have the functionality of a B2BUA would have been convenient because then we could have had access to its internal registration database and could also use some of the already existing forwarding functionality. However, it was quickly realized that this would require too much work, so a standalone B2BUA was developed instead.

A combined proxy and registrar was needed anyway to keep track of the UAs’ true locations. The proxy was programmed not to send requests to the final destination, but instead pass every request meant for an audio UA directly to the B2BUA forcing the B2BUA into the communication path of every call. Before the proxy passed the request on to the B2BUA it rewrote the URI of the request to the UA’s contact information located in the registration database. This means that the B2BUA does not need to contact the proxy again to deliver the request to the correct destination.

As with all B2BUAs there is one UA receiving incoming requests, thus acting as a UAS, and one UA initiating a call with the incoming request’s real recipient, in the way a UAC does. However, in this B2BUA there are two additional UAs, which initiate sessions with the STBs in case the caller or callee does not support video (see Figure 21). To ease development, the UA developed for the video UA was reused for each of the four UAs found in the B2BUA.

<table>
<thead>
<tr>
<th>Server side</th>
<th>Client side</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video UAC</td>
<td>Video UAC</td>
</tr>
<tr>
<td>UAS</td>
<td>UAC</td>
</tr>
</tbody>
</table>

Figure 21: User agents in the B2BUA.
The different combinations of UAs involved in a video telephony session and which need to be supported by the B2BUA are:

- **Audio UA → Audio UA**
  Two audio UAs communicate without any video involved.

- **Audio UA + Video UA → Audio UA + Video UA**
  Audio is handled by two audio UAs while video is taken care of by STBs running video UAs.

- **Audio UA + Video UA → Audio&Video UA**
  An audio UA and an STB running a video UA have an audio and video session with an ordinary UA that has both audio and video support.

- **Standard SIP videophone → Standard SIP videophone**
  Two ordinary UAs with both audio and video support should be able to communicate through the B2BUA. This is not really a requirement, since it does not involve any STBs, but it would be nice to have this functionality for completeness and to prove that the SIP implementation is working.

Let us go through each one of these scenarios and have a look at the SIP traffic generated and the decisions made by the B2BUA when all sessions are set up. The figures are slightly simplified since the proxy/registrar that forwards all messages to/from the B2BUA is not shown in the figures. To be able to easily contact an STB that belongs to an audio UA the STB’s video UA is given a SIP address similar to the audio UA’s, but with “_STB” appended. This way the B2BUA knows that the STB belonging to sip:robert@foobar.se has the address sip:robert_STB@foobar.se. It would be possible to give the STB and audio UA the same SIP address and use the forking functionality of stateful proxy servers to send requests to both UAs, but since it only complicates things without really adding any functionality the names were kept separate. Dashed lines in the figures below show communication going to the user’s STB while “non-dashed” lines show communication with the audio UA.
9.2.1 Audio UA → Audio UA

When an INVITE request arrives at the B2BUA from User1’s audio UA the B2BUA inspects the SDP body and notices that there is only one media section containing the audio stream. Since User1 might want to add video to this call the B2BUA will send an invitation to User1_STB, but it will also send an audio only INVITE to the original recipient, User2. Since the B2BUA does not have any video information from User2 or User2_STB it cannot offer User1_STB a valid SDP body, but instead it specifies a video address of 0.0.0.0 in the offer sent to User1_STB. User1_STB will interpret this as a video offer, but it will put the video stream on hold and not start any transmissions. In this case the invitation of User1_STB is declined and the B2BUA gives up trying to add video to the session.

When User2 receives the audio invitation a “180 Ringing” response is sent back and the user is alerted. The B2BUA forwards the provisional response to User1. Once the call is accepted User2 returns “200 OK”, which B2BUA acknowledges and forwards to User1. The SDP offer and answer included in the initial INVITE from User1 for User2 and in the “200 OK” response from User2 are left unchanged by the B2BUA, which means that media will go straight between User1 and User2 without any involvement from the B2BUA. Once User1 receives the “200 OK” response the session is established and User1 and User2 can have an audio conversation.
9.2.2 Audio UA + Video UA → Audio UA + Video UA

User1/User1 STB → B2BUA → User2/User2 STB

INVITE User2
100 Trying

INVITE User1_STB
200 OK
ACK
180 Ringing

INVITE User2
200 OK
180 Ringing

ACK
200 OK

ACK
200 OK

ACK

This is the most complex scenario as it involves all four UAs in the B2BUA. User1 starts by sending an INVITE request for Phone2, which is received by the B2BUA. As in the previous example the B2BUA notices that User1 does not support video and so B2BUA calls User1’s STB to add video. An audio invitation is also sent to User2. This time User1_STB accepts the invitation, which the B2BUA takes a note of. As soon as User2 accepts the call the B2BUA sends “200 OK” to User1 and attaches the media description received from User2. At this point an audio session between User1 and User2 has been
established. The B2BUA knows however that the server side has support for video through User1_STB. This triggers the B2BUA to ask User2 if it can also support video. We initially only asked User2 for audio since that was what User1 offered, but User2 may also be capable of video. User2 declines this new media though and the B2BUA tries to contact User2_STB instead with User1_STB’s media information. This succeeds so now User2_STB can start send video to User1_STB, but User1_STB does not yet know where User2_STB expects to receive video since it was put on hold with an IP address of 0.0.0.0 specified in the SDP message’s video media section. The B2BUA tells User1_STB of the new information it has acquired from User2_STB by sending a re-INVITE request containing a new video media section in the SDP offer. At this point both an audio and a video session has been established and media flows between the involved UAs. No media passes through the B2BUA in this scenario either.
This session is set up in a similar way as above. When an audio session has been established between User1 and User2 the B2BUA will try to re-INVITE User2 with the new video information it has received from User1_STB. The B2BUA merges the audio information it has received in User1’s SDP offer and the video information received in User1_STB’s SDP answer into a single SDP offer for User2. This is possible since each media section in an SDP message can contain its own c-header and IP address as described in section 4.5. This way an Audio&Video UA can be told to send audio and video to two different hosts relieving the B2BUA from handling the media streams.
9.2.4 Standard SIP videophone → Standard SIP videophone

When two audio and video capable UAs communicate through the B2BUA there is very little communication needed. The B2BUA does not need to call up any STBs or rewrite any SDP packets. It will receive an INVITE request from User1, notice, through SDP inspection, that User1 supports both audio and video and offer User2 both audio and video by forwarding User1’s SDP offer to User2. Once User2 establishes a session with the B2BUA the B2BUA will send a “200 OK” response to User1 containing User2’s SDP answer.
10 Conclusions

With this project we aimed for a video telephony system that should work with existing network connected set-top boxes and where SIP was used to manage the multimedia sessions.

As SIP is such a flexible protocol with no requirements on the sessions it manages, it was never a problem to use SIP for video telephony sessions. Since there are open standards defining how video streams are embedded into RTP streams it was also easy to get the video streams standard compliant and working with other video telephony implementations.

Thanks to the flexibility of SIP it was also possible to use a B2BUA to merge two separate SIP UAs into one single entity with both audio and video support. Without this possibility it would have been impossible to get interoperability between the split audio and video UAs and an ordinary UA with support for both audio and video such as Microsoft’s Windows Messenger (Microsoft Corporation, 2004a).

The limited hardware available in the set-top boxes was found sufficient for video telephony, although barely. Using the H.261 codec to compress the video data, the set-top box could handle CIF sized video streams of eight frames per second, being well within the proposed frame rate range for video telephony (5-10). Since QCIF is the recommended resolution for video telephony even higher frame rates would be possible if QCIF were to be used instead of CIF, but then another camera would have to be used since the Labtec camera has a limited frame rate of approximately eight. Using QCIF with the Labtec camera would only be a waste of processor cycles.

The implemented system fulfils all primary requirements set up in chapter 7. The optional interoperability requirement is also satisfied since interoperability has been confirmed with SIP telephony applications like Microsoft’s Windows Messenger, Cisco’s ATA 186 (Cisco Systems Inc., 2004), Ubiquity’s SIP UA (Ubiquity Software Corporation, 2004) and Ahead’s SIPPS (Ahead Software, 2004).
10.1 Future enhancements

As mentioned in the introduction, the applications developed in this project are not meant for public use, which means that there are many things that can be improved and in some cases really need to be improved before public use. Here follows a list of the most important and useful changes possible.

Since not all of the optional requirements were fulfilled future work could be aimed at fulfilling those. This would involve:

- Adding more flexibility to the video management. Currently it is only possible to add video when the audio session is initiated. Should the STB be turned off at that moment or the user would decline video it is impossible to add video at a later instance during the call.
- Implementing a phone book in the video UAs GUI. Since all calls are currently initiated by an audio UA the user has to remember or look up phone numbers to make calls. The phone book would make it easier to make calls since it would allow the user to initiate calls by simply choosing a user’s name from a list.
- Adding more sessions to a call. A call could be extended with game or chat sessions and not be limited to only audio and video.

Also, the B2BUA developed for this project has several limitations. The biggest is probably that it only supports one simultaneous call, which of course cannot be tolerated if it should be able to serve a large network of set-top boxes. It also has problems forwarding error messages between its server and client parts since only a small subset of the error messages is supported. Should the client side receive an unsupported error message the server side will send out a different error message.

It would be much preferred to have all video decompression and compression performed by hardware instead of putting these tasks on the main CPU. This would give better image quality, higher frame rates and lower video latency.

During the implementation of this project there has been work performed on the FFMPEG library and its H.263 and H.263+ implementations, which has resulted in that these codecs should now produce standard compliant video streams (Tardy, 2003). To extend the abilities and improve interoperability of the video UA it might be a good idea to implement also FFMPEG’s H.263 codec.

As was mentioned earlier in this report, RTCP is not used to control the RTP sessions established for the video, but supporting RTCP would be a good idea. Since the STBs have limited computation capacity they cannot handle video streams with too high frame rate and instead of the STBs throwing away much
of the video information that it does not have time to decode it would be preferable to instead tell the sender, through RTCP, to slow down the video stream.

RTCP can also be used to synchronize RTP streams. Currently, video and audio streams are sent independently and directly between the involved UAs. This could result in bad synchronization between audio and video. By having audio and video passing through the B2BUA it could be possible to synchronize audio and video before it is forwarded to the receiver.

As was mentioned in section 9.1.6, no buffering of video data is performed in the video UA and hardly any checks are performed on the RTP headers on the received data. This could result in distorted decoded images and strange behaviour. This should really be taken care of.

The video UA is currently a completely separated application running on the STB, which means that it interacts badly with other applications available for the STB. To be able to receive video calls while watching TV, for example, there needs to be some communication between the video UA and the TV application so that the user gets notified of the incoming call and can answer it.

Although SIP seems to have a bright future ahead of it H.323 is still what many existing video telephony systems are using. To connect SIP phones to other types of telephony networks a gateway is needed to handle translation between the different networks’ protocols. There exist gateways that can convert between SIP and H.323, but these gateways handle only audio. A possible future enhancement could be to construct a SIP H.323 gateway that can handle audio, video and all other session types implemented in the video telephony system.

Finally, if a user is located far away from the camera, it is hard to get a good, detailed view of the user. To remedy this it would be nice to have a zoom function in the video UA. This would either require a camera capable of zooming or a camera supporting high resolutions so that zooming can be done in software.
11 Bibliography


64
Tardy, Guilhem (2003). [*OpenH323*] H.263(+) codec now RFC2190 (NetMeeting) compliant and else. [www]


Appendix 1: Video benchmarks

The benchmarks on the following pages show the compression speed of different video compression algorithms and implementations. Two sets of benchmark runs were conducted: One for CIF resolution and one for QCIF.

Prior to the tests the time taken to capture a frame from the camera, to convert it to YUV420 colour space and to render it on the TV were measured for each resolution. The results were used to calculate the estimated frame rates. The measured times are:

<table>
<thead>
<tr>
<th></th>
<th>CIF [ms]</th>
<th>QCIF [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame capture</td>
<td>22,5</td>
<td>5,74</td>
</tr>
<tr>
<td>Colour space conversion</td>
<td>28,73</td>
<td>7,14</td>
</tr>
<tr>
<td>TV rendering</td>
<td>7,47</td>
<td>1,94</td>
</tr>
</tbody>
</table>

When calculating the bit rate a frame rate of 8 frames per second was assumed.
## CIF resolution

<table>
<thead>
<tr>
<th>OpenH323 H.261 CIF</th>
<th>Compression time [ms]</th>
<th>Decompression time [ms]</th>
<th>Estimated frame rate</th>
<th>Bit rate [kbps]</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Shortest</td>
<td>Average</td>
<td>Longest</td>
<td>Shortest</td>
<td>Average</td>
</tr>
<tr>
<td>q = 1</td>
<td>0,87</td>
<td>43,67</td>
<td>148,92</td>
<td>0,05</td>
<td>27,74</td>
</tr>
<tr>
<td>q = 5</td>
<td>0,96</td>
<td>41,54</td>
<td>158,93</td>
<td>0,05</td>
<td>18,17</td>
</tr>
<tr>
<td>q = 10</td>
<td>0,88</td>
<td>38,37</td>
<td>136,34</td>
<td>0,05</td>
<td>13,90</td>
</tr>
<tr>
<td>q = 15</td>
<td>0,88</td>
<td>38,45</td>
<td>154,47</td>
<td>0,05</td>
<td>11,34</td>
</tr>
<tr>
<td>q = 20</td>
<td>0,87</td>
<td>38,33</td>
<td>133,74</td>
<td>0,05</td>
<td>9,66</td>
</tr>
<tr>
<td>q = 25</td>
<td>0,85</td>
<td>38,04</td>
<td>131,54</td>
<td>0,05</td>
<td>8,31</td>
</tr>
<tr>
<td>q = 30</td>
<td>0,87</td>
<td>38,16</td>
<td>133,43</td>
<td>0,05</td>
<td>7,37</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OpenH323 H.263 CIF</th>
<th>Compression time [ms]</th>
<th>Decompression time [ms]</th>
<th>Estimated frame rate</th>
<th>Bit rate [kbps]</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Shortest</td>
<td>Average</td>
<td>Longest</td>
<td>Shortest</td>
<td>Average</td>
</tr>
<tr>
<td>q = 1, gop = 0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>q = 1, gop = 5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>q = 5, gop = 0</td>
<td>254,40</td>
<td>281,43</td>
<td>313,22</td>
<td>48,46</td>
<td>71,79</td>
</tr>
<tr>
<td>q = 5, gop = 5</td>
<td>258,01</td>
<td>372,56</td>
<td>534,69</td>
<td>25,22</td>
<td>60,72</td>
</tr>
<tr>
<td>q = 10, gop = 0</td>
<td>239,79</td>
<td>260,63</td>
<td>284,06</td>
<td>45,73</td>
<td>58,54</td>
</tr>
<tr>
<td>q = 10, gop = 5</td>
<td>241,90</td>
<td>343,55</td>
<td>463,16</td>
<td>24,44</td>
<td>48,39</td>
</tr>
<tr>
<td>q = 15, gop = 0</td>
<td>233,70</td>
<td>252,24</td>
<td>281,53</td>
<td>44,37</td>
<td>53,29</td>
</tr>
<tr>
<td>q = 15, gop = 5</td>
<td>234,62</td>
<td>333,86</td>
<td>429,56</td>
<td>24,14</td>
<td>42,80</td>
</tr>
<tr>
<td>q = 20, gop = 0</td>
<td>229,65</td>
<td>245,88</td>
<td>281,40</td>
<td>41,08</td>
<td>50,57</td>
</tr>
<tr>
<td>q = 20, gop = 5</td>
<td>232,33</td>
<td>328,74</td>
<td>403,96</td>
<td>23,00</td>
<td>40,40</td>
</tr>
<tr>
<td>q = 25, gop = 0</td>
<td>225,50</td>
<td>241,97</td>
<td>278,29</td>
<td>40,21</td>
<td>48,73</td>
</tr>
<tr>
<td>q = 25, gop = 5</td>
<td>227,48</td>
<td>326,38</td>
<td>397,73</td>
<td>25,68</td>
<td>38,55</td>
</tr>
<tr>
<td>q = 30, gop = 0</td>
<td>225,98</td>
<td>239,98</td>
<td>270,12</td>
<td>40,44</td>
<td>47,42</td>
</tr>
<tr>
<td>q = 30, gop = 5</td>
<td>227,04</td>
<td>324,74</td>
<td>381,85</td>
<td>24,74</td>
<td>37,41</td>
</tr>
<tr>
<td>FFMPEG H.263 CIF</td>
<td>Compression time [ms]</td>
<td>Decompression time [ms]</td>
<td>Estimated frame rate</td>
<td>Bit rate [kbps]</td>
<td>Quality</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------</td>
<td>-------------------------</td>
<td>----------------------</td>
<td>----------------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td>Shortest</td>
<td>Average</td>
<td>Longest</td>
<td>Shortest</td>
<td>Average</td>
</tr>
<tr>
<td>q = 1, gop = 0</td>
<td>82,78</td>
<td>85,46</td>
<td>108,24</td>
<td>46,88</td>
<td>47,78</td>
</tr>
<tr>
<td>q = 1, gop = 5</td>
<td>124,69</td>
<td>128,15</td>
<td>158,51</td>
<td>46,97</td>
<td>47,90</td>
</tr>
<tr>
<td>q = 5, gop = 0</td>
<td>79,85</td>
<td>83,19</td>
<td>110,52</td>
<td>42,16</td>
<td>42,89</td>
</tr>
<tr>
<td>q = 5, gop = 5</td>
<td>119,97</td>
<td>122,94</td>
<td>145,56</td>
<td>42,26</td>
<td>43,30</td>
</tr>
<tr>
<td>q = 10, gop = 0</td>
<td>78,70</td>
<td>81,56</td>
<td>104,08</td>
<td>41,14</td>
<td>41,88</td>
</tr>
<tr>
<td>q = 10, gop = 5</td>
<td>117,55</td>
<td>121,17</td>
<td>143,56</td>
<td>41,15</td>
<td>41,90</td>
</tr>
<tr>
<td>q = 15, gop = 0</td>
<td>78,00</td>
<td>80,39</td>
<td>115,98</td>
<td>40,68</td>
<td>41,40</td>
</tr>
<tr>
<td>q = 15, gop = 5</td>
<td>116,47</td>
<td>119,13</td>
<td>141,93</td>
<td>40,64</td>
<td>41,43</td>
</tr>
<tr>
<td>q = 20, gop = 0</td>
<td>77,68</td>
<td>80,68</td>
<td>103,90</td>
<td>40,43</td>
<td>41,18</td>
</tr>
<tr>
<td>q = 20, gop = 5</td>
<td>116,49</td>
<td>118,93</td>
<td>142,17</td>
<td>40,45</td>
<td>41,15</td>
</tr>
<tr>
<td>q = 25, gop = 0</td>
<td>79,15</td>
<td>81,34</td>
<td>101,10</td>
<td>40,29</td>
<td>40,94</td>
</tr>
<tr>
<td>q = 25, gop = 5</td>
<td>117,73</td>
<td>120,85</td>
<td>143,74</td>
<td>40,33</td>
<td>41,06</td>
</tr>
<tr>
<td>q = 30, gop = 0</td>
<td>77,55</td>
<td>79,81</td>
<td>103,03</td>
<td>40,24</td>
<td>40,95</td>
</tr>
<tr>
<td>q = 30, gop = 5</td>
<td>115,87</td>
<td>118,93</td>
<td>136,25</td>
<td>40,27</td>
<td>41,18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FFMPEG H.263+ CIF</th>
<th>Compression time [ms]</th>
<th>Decompression time [ms]</th>
<th>Estimated frame rate</th>
<th>Bit rate [kbps]</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Shortest</td>
<td>Average</td>
<td>Longest</td>
<td>Shortest</td>
<td>Average</td>
</tr>
<tr>
<td>q = 1, gop = 0</td>
<td>83,13</td>
<td>86,88</td>
<td>109,40</td>
<td>46,49</td>
<td>47,47</td>
</tr>
<tr>
<td>q = 1, gop = 5</td>
<td>125,58</td>
<td>130,13</td>
<td>156,91</td>
<td>46,59</td>
<td>47,29</td>
</tr>
<tr>
<td>q = 5, gop = 0</td>
<td>80,58</td>
<td>84,38</td>
<td>108,18</td>
<td>41,86</td>
<td>42,66</td>
</tr>
<tr>
<td>q = 5, gop = 5</td>
<td>120,63</td>
<td>123,42</td>
<td>146,19</td>
<td>41,88</td>
<td>43,23</td>
</tr>
<tr>
<td>q = 10, gop = 0</td>
<td>79,14</td>
<td>82,38</td>
<td>104,96</td>
<td>40,78</td>
<td>41,52</td>
</tr>
<tr>
<td>q = 10, gop = 5</td>
<td>118,79</td>
<td>122,00</td>
<td>156,21</td>
<td>40,82</td>
<td>41,58</td>
</tr>
<tr>
<td>q = 15, gop = 0</td>
<td>78,43</td>
<td>81,94</td>
<td>108,21</td>
<td>40,35</td>
<td>41,03</td>
</tr>
<tr>
<td>q = 15, gop = 5</td>
<td>117,51</td>
<td>121,33</td>
<td>143,35</td>
<td>40,39</td>
<td>41,14</td>
</tr>
<tr>
<td>q = 20, gop = 0</td>
<td>78,16</td>
<td>81,16</td>
<td>103,12</td>
<td>40,11</td>
<td>40,84</td>
</tr>
<tr>
<td>q = 20, gop = 5</td>
<td>117,64</td>
<td>121,49</td>
<td>144,40</td>
<td>40,16</td>
<td>40,93</td>
</tr>
<tr>
<td>q = 25, gop = 0</td>
<td>79,14</td>
<td>82,24</td>
<td>105,19</td>
<td>40,01</td>
<td>40,75</td>
</tr>
<tr>
<td>q = 25, gop = 5</td>
<td>117,84</td>
<td>122,08</td>
<td>156,08</td>
<td>40,02</td>
<td>40,70</td>
</tr>
<tr>
<td>q = 30, gop = 0</td>
<td>77,91</td>
<td>81,21</td>
<td>107,51</td>
<td>39,88</td>
<td>40,70</td>
</tr>
<tr>
<td>q = 30, gop = 5</td>
<td>117,12</td>
<td>120,67</td>
<td>142,33</td>
<td>40,00</td>
<td>40,70</td>
</tr>
<tr>
<td>FFmpeg MPEG1 CIF</td>
<td>Compression time [ms]</td>
<td>Decompression time [ms]</td>
<td>Estimated frame rate</td>
<td>Bit rate [kbps]</td>
<td>Quality</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------</td>
<td>-------------------------</td>
<td>----------------------</td>
<td>----------------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td>Shortest</td>
<td>Average</td>
<td>Longest</td>
<td>Shortest</td>
<td>Average</td>
</tr>
<tr>
<td>q = 1, gop = 0</td>
<td>79,56</td>
<td>81,21</td>
<td>104,39</td>
<td>42,26</td>
<td>44,73</td>
</tr>
<tr>
<td>q = 1, gop = 5</td>
<td>62,55</td>
<td>82,08</td>
<td>136,06</td>
<td>9,15</td>
<td>20,61</td>
</tr>
<tr>
<td>q = 5, gop = 0</td>
<td>78,10</td>
<td>80,25</td>
<td>108,26</td>
<td>40,29</td>
<td>42,38</td>
</tr>
<tr>
<td>q = 5, gop = 5</td>
<td>63,39</td>
<td>79,08</td>
<td>129,23</td>
<td>8,79</td>
<td>18,87</td>
</tr>
<tr>
<td>q = 10, gop = 0</td>
<td>77,80</td>
<td>79,64</td>
<td>103,41</td>
<td>38,55</td>
<td>41,05</td>
</tr>
<tr>
<td>q = 10, gop = 5</td>
<td>58,20</td>
<td>79,46</td>
<td>129,63</td>
<td>8,60</td>
<td>18,57</td>
</tr>
<tr>
<td>q = 15, gop = 0</td>
<td>77,43</td>
<td>79,26</td>
<td>97,00</td>
<td>39,13</td>
<td>41,41</td>
</tr>
<tr>
<td>q = 15, gop = 5</td>
<td>57,67</td>
<td>73,61</td>
<td>135,74</td>
<td>8,00</td>
<td>16,79</td>
</tr>
<tr>
<td>q = 20, gop = 0</td>
<td>77,18</td>
<td>79,12</td>
<td>103,31</td>
<td>38,58</td>
<td>41,16</td>
</tr>
<tr>
<td>q = 20, gop = 5</td>
<td>57,83</td>
<td>72,90</td>
<td>125,17</td>
<td>7,67</td>
<td>16,52</td>
</tr>
<tr>
<td>q = 25, gop = 0</td>
<td>77,19</td>
<td>79,05</td>
<td>101,93</td>
<td>32,26</td>
<td>40,98</td>
</tr>
<tr>
<td>q = 25, gop = 5</td>
<td>55,52</td>
<td>72,15</td>
<td>136,50</td>
<td>7,36</td>
<td>16,26</td>
</tr>
<tr>
<td>q = 30, gop = 0</td>
<td>77,07</td>
<td>79,25</td>
<td>101,47</td>
<td>32,82</td>
<td>41,09</td>
</tr>
<tr>
<td>q = 30, gop = 5</td>
<td>55,08</td>
<td>71,08</td>
<td>129,85</td>
<td>6,97</td>
<td>15,89</td>
</tr>
</tbody>
</table>
### QCIF resolution

<table>
<thead>
<tr>
<th></th>
<th>Compression time [ms]</th>
<th>Decompression time [ms]</th>
<th>Estimated frame rate</th>
<th>Bit rate [kbps]</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Shortest</td>
<td>Average</td>
<td>Longest</td>
<td>Shortest</td>
<td>Average</td>
</tr>
<tr>
<td><strong>OpenH323 H.261 QCIF</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q = 1</td>
<td>0,35</td>
<td>11,96</td>
<td>39,94</td>
<td>0,03</td>
<td>7,77</td>
</tr>
<tr>
<td>q = 5</td>
<td>0,37</td>
<td>11,46</td>
<td>38,53</td>
<td>0,05</td>
<td>5,56</td>
</tr>
<tr>
<td>q = 10</td>
<td>0,36</td>
<td>10,42</td>
<td>35,39</td>
<td>0,05</td>
<td>4,42</td>
</tr>
<tr>
<td>q = 15</td>
<td>0,36</td>
<td>10,37</td>
<td>36,09</td>
<td>0,05</td>
<td>3,71</td>
</tr>
<tr>
<td>q = 20</td>
<td>0,37</td>
<td>10,30</td>
<td>36,11</td>
<td>0,04</td>
<td>3,23</td>
</tr>
<tr>
<td>q = 25</td>
<td>0,39</td>
<td>10,34</td>
<td>51,59</td>
<td>0,05</td>
<td>2,84</td>
</tr>
<tr>
<td>q = 30</td>
<td>0,39</td>
<td>10,27</td>
<td>35,13</td>
<td>0,03</td>
<td>2,56</td>
</tr>
</tbody>
</table>

<p>| <strong>OpenH323 H.263 QCIF</strong> |         |         |         |         |         |         |          |         |         |          |
| q = 1, gop = 0 | -       | -       | -       | -       | -       | -       | -         | -       | -       | 0        |
| q = 1, gop = 5 | -       | -       | -       | -       | -       | -       | -         | -       | -       | 0        |
| q = 5, gop = 0 | 65,76   | 75,98   | 100,86  | 11,13   | 21,27  | 30,21  | 10,9    | 8,9    | 6,9      | 221     | 4       |
| q = 5, gop = 5 | 67,38   | 97,36   | 160,42  | 7,64    | 15,51  | 29,10  | 11,1    | 7,8    | 4,9      | 81      | 4       |
| q = 10, gop = 0| 63,43   | 70,22   | 94,55   | 8,46    | 17,26  | 41,22  | 11,5    | 9,8    | 6,6      | 131     | 2       |
| q = 10, gop = 5| 64,00   | 89,80   | 138,99  | 7,58    | 12,66  | 33,13  | 11,6    | 8,5    | 5,3      | 47      | 2       |
| q = 15, gop = 0| 62,36   | 67,52   | 101,58  | 8,27    | 15,33  | 24,89  | 11,7    | 10,2   | 7,1      | 99      | 1       |
| q = 15, gop = 5| 62,39   | 87,03   | 128,50  | 7,41    | 11,42  | 22,98  | 11,8    | 8,8    | 6,0      | 35      | 1       |
| q = 20, gop = 0| 61,02   | 65,70   | 89,79   | 8,79    | 14,29  | 22,26  | 11,8    | 10,5   | 7,9      | 83      | 1       |
| q = 20, gop = 5| 61,26   | 85,26   | 124,55  | 7,30    | 10,72  | 19,58  | 12,0    | 9,0    | 6,3      | 30      | 1       |
| q = 25, gop = 0| 60,04   | 64,40   | 84,51   | 8,26    | 13,53  | 22,31  | 12,0    | 10,8   | 8,2      | 74      | 1       |
| q = 25, gop = 5| 60,64   | 84,08   | 117,55  | 7,11    | 10,20  | 17,06  | 12,1    | 9,2    | 6,7      | 26      | 1       |
| q = 30, gop = 0| 59,17   | 63,41   | 87,03   | 8,14    | 13,05  | 20,86  | 12,2    | 11,0   | 8,1      | 67      | 1       |
| q = 30, gop = 5| 59,83   | 83,33   | 117,56  | 7,10    | 9,89   | 16,09  | 12,2    | 9,3    | 6,7      | 24      | 1       |</p>
<table>
<thead>
<tr>
<th>FFmpeg H.263 QCIF</th>
<th>Compression time [ms]</th>
<th>Decompression time [ms]</th>
<th>Estimated frame rate</th>
<th>Bit rate [kbps]</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Shortest</td>
<td>Average</td>
<td>Longest</td>
<td>Shortest</td>
<td>Average</td>
</tr>
<tr>
<td>q = 1, gop = 0</td>
<td>21.35</td>
<td>22.94</td>
<td>36.28</td>
<td>12.29</td>
<td>12.51</td>
</tr>
<tr>
<td>q = 1, gop = 5</td>
<td>32.70</td>
<td>33.56</td>
<td>41.29</td>
<td>12.32</td>
<td>12.54</td>
</tr>
<tr>
<td>q = 5, gop = 0</td>
<td>20.65</td>
<td>21.25</td>
<td>45.72</td>
<td>10.88</td>
<td>11.08</td>
</tr>
<tr>
<td>q = 5, gop = 5</td>
<td>31.11</td>
<td>31.81</td>
<td>42.73</td>
<td>10.90</td>
<td>11.13</td>
</tr>
<tr>
<td>q = 10, gop = 0</td>
<td>20.06</td>
<td>20.66</td>
<td>31.79</td>
<td>10.49</td>
<td>10.72</td>
</tr>
<tr>
<td>q = 10, gop = 5</td>
<td>30.39</td>
<td>31.05</td>
<td>39.82</td>
<td>10.51</td>
<td>10.77</td>
</tr>
<tr>
<td>q = 15, gop = 0</td>
<td>19.77</td>
<td>20.54</td>
<td>44.18</td>
<td>10.37</td>
<td>10.56</td>
</tr>
<tr>
<td>q = 15, gop = 5</td>
<td>29.89</td>
<td>30.84</td>
<td>55.28</td>
<td>10.39</td>
<td>10.62</td>
</tr>
<tr>
<td>q = 20, gop = 0</td>
<td>19.87</td>
<td>20.52</td>
<td>31.88</td>
<td>10.28</td>
<td>10.48</td>
</tr>
<tr>
<td>q = 20, gop = 5</td>
<td>30.23</td>
<td>30.80</td>
<td>40.64</td>
<td>10.31</td>
<td>10.57</td>
</tr>
<tr>
<td>q = 25, gop = 0</td>
<td>20.25</td>
<td>20.76</td>
<td>29.23</td>
<td>10.25</td>
<td>10.44</td>
</tr>
<tr>
<td>q = 25, gop = 5</td>
<td>30.26</td>
<td>30.86</td>
<td>38.86</td>
<td>10.27</td>
<td>10.48</td>
</tr>
<tr>
<td>q = 30, gop = 0</td>
<td>19.94</td>
<td>20.34</td>
<td>28.21</td>
<td>10.19</td>
<td>10.41</td>
</tr>
<tr>
<td>q = 30, gop = 5</td>
<td>29.62</td>
<td>30.47</td>
<td>41.34</td>
<td>10.22</td>
<td>10.51</td>
</tr>
<tr>
<td>FFmpeg H.263+ QCIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Shortest</td>
<td>Average</td>
<td>Longest</td>
<td>Shortest</td>
<td>Average</td>
</tr>
<tr>
<td>q = 1, gop = 0</td>
<td>21.44</td>
<td>22.06</td>
<td>30.42</td>
<td>12.42</td>
<td>12.69</td>
</tr>
<tr>
<td>q = 1, gop = 5</td>
<td>32.97</td>
<td>33.57</td>
<td>42.27</td>
<td>12.44</td>
<td>12.69</td>
</tr>
<tr>
<td>q = 5, gop = 0</td>
<td>20.42</td>
<td>21.14</td>
<td>44.97</td>
<td>11.01</td>
<td>11.25</td>
</tr>
<tr>
<td>q = 5, gop = 5</td>
<td>31.13</td>
<td>31.84</td>
<td>40.57</td>
<td>11.03</td>
<td>11.30</td>
</tr>
<tr>
<td>q = 10, gop = 0</td>
<td>20.17</td>
<td>20.68</td>
<td>28.77</td>
<td>10.62</td>
<td>10.83</td>
</tr>
<tr>
<td>q = 10, gop = 5</td>
<td>30.50</td>
<td>31.08</td>
<td>40.24</td>
<td>10.64</td>
<td>10.92</td>
</tr>
<tr>
<td>q = 15, gop = 0</td>
<td>19.97</td>
<td>20.44</td>
<td>28.73</td>
<td>10.49</td>
<td>10.74</td>
</tr>
<tr>
<td>q = 15, gop = 5</td>
<td>30.25</td>
<td>30.84</td>
<td>38.85</td>
<td>10.51</td>
<td>10.74</td>
</tr>
<tr>
<td>q = 20, gop = 0</td>
<td>19.98</td>
<td>20.50</td>
<td>44.73</td>
<td>10.40</td>
<td>10.59</td>
</tr>
<tr>
<td>q = 20, gop = 5</td>
<td>30.09</td>
<td>30.87</td>
<td>54.73</td>
<td>10.42</td>
<td>10.65</td>
</tr>
<tr>
<td>q = 25, gop = 0</td>
<td>20.06</td>
<td>20.59</td>
<td>28.31</td>
<td>10.37</td>
<td>10.57</td>
</tr>
<tr>
<td>q = 25, gop = 5</td>
<td>30.22</td>
<td>30.86</td>
<td>55.19</td>
<td>10.39</td>
<td>10.61</td>
</tr>
<tr>
<td>q = 30, gop = 0</td>
<td>19.77</td>
<td>20.30</td>
<td>28.72</td>
<td>10.31</td>
<td>10.55</td>
</tr>
<tr>
<td>q = 30, gop = 5</td>
<td>29.95</td>
<td>30.53</td>
<td>38.35</td>
<td>10.35</td>
<td>10.51</td>
</tr>
<tr>
<td>FFMPEG</td>
<td>MPEG1 QCIF</td>
<td>Compression time [ms]</td>
<td>Decompression time [ms]</td>
<td>Estimated frame rate</td>
<td>Bit rate [kbps]</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------------------------</td>
<td>-------------------------</td>
<td>----------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>q = 1, gop = 0</td>
<td>20,53 21,05 45,70</td>
<td>8,90 12,15 20,14</td>
<td>28,3 27,9 16,5</td>
<td>134</td>
<td>5</td>
</tr>
<tr>
<td>q = 1, gop = 5</td>
<td>16,25 24,68 40,99</td>
<td>2,68 6,66 19,52</td>
<td>32,2 25,3 17,9</td>
<td>83</td>
<td>5</td>
</tr>
<tr>
<td>q = 5, gop = 0</td>
<td>20,08 20,66 45,09</td>
<td>8,76 11,07 18,50</td>
<td>28,7 28,2 16,7</td>
<td>112</td>
<td>5</td>
</tr>
<tr>
<td>q = 5, gop = 5</td>
<td>14,44 23,25 37,59</td>
<td>2,45 5,95 19,50</td>
<td>34,2 26,3 19,1</td>
<td>68</td>
<td>5</td>
</tr>
<tr>
<td>q = 10, gop = 0</td>
<td>19,77 20,43 44,64</td>
<td>8,87 10,82 18,80</td>
<td>28,9 28,4 16,8</td>
<td>88</td>
<td>5</td>
</tr>
<tr>
<td>q = 10, gop = 5</td>
<td>14,42 22,56 32,54</td>
<td>2,37 5,62 15,33</td>
<td>34,2 26,8 21,1</td>
<td>59</td>
<td>5</td>
</tr>
<tr>
<td>q = 15, gop = 0</td>
<td>19,82 20,32 43,99</td>
<td>8,90 10,72 18,70</td>
<td>28,9 28,5 17,0</td>
<td>76</td>
<td>5</td>
</tr>
<tr>
<td>q = 15, gop = 5</td>
<td>13,95 21,30 53,84</td>
<td>2,22 5,14 11,72</td>
<td>34,8 27,7 14,6</td>
<td>48</td>
<td>5</td>
</tr>
<tr>
<td>q = 20, gop = 0</td>
<td>19,60 20,24 31,26</td>
<td>8,94 10,69 18,41</td>
<td>29,1 28,5 21,7</td>
<td>69</td>
<td>5</td>
</tr>
<tr>
<td>q = 20, gop = 5</td>
<td>13,92 20,95 53,34</td>
<td>2,37 5,04 11,51</td>
<td>34,8 28,0 14,7</td>
<td>47</td>
<td>5</td>
</tr>
<tr>
<td>q = 25, gop = 0</td>
<td>19,83 20,27 28,61</td>
<td>8,87 10,60 18,72</td>
<td>28,9 28,5 23,0</td>
<td>63</td>
<td>5</td>
</tr>
<tr>
<td>q = 25, gop = 5</td>
<td>14,09 21,13 44,07</td>
<td>2,30 4,91 11,17</td>
<td>34,6 27,8 17,0</td>
<td>43</td>
<td>5</td>
</tr>
<tr>
<td>q = 30, gop = 0</td>
<td>19,74 20,18 29,14</td>
<td>8,91 10,52 18,67</td>
<td>28,9 28,6 22,7</td>
<td>57</td>
<td>5</td>
</tr>
<tr>
<td>q = 30, gop = 5</td>
<td>14,06 21,37 37,06</td>
<td>2,19 5,06 34,38</td>
<td>34,6 27,6 19,3</td>
<td>43</td>
<td>5</td>
</tr>
</tbody>
</table>