HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Electrical and Communications Engineering
Laboratory of Acoustics and Audio Signal Processing

Jussi Mutanen

A System for Real-Time High-Quality Audio Streaming

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Espoo, Aug 20, 2002

Supervisor: Professor Matti Karjalainen
Instructors: Martti Rahkila

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Jussi Mutanen
Name of the thesis:	A System for Real-Time High-Quality Audio Streaming
Date:	Aug 20, 2002 **Number of pages:** 67
Department:	Electrical and Communications Engineering
Professorship:	S-89
Supervisor:	Prof. Matti Karjalainen
Instructors:	Martti Rahkila, M.Sc.

Real-Time Streaming Protocol (RTSP) establishes and controls time-synchronized streams of continuous media in the Internet. The main aim of RTSP is to control multiple data sources, both live sessions and stored clips.

Real-time Transport Protocol (RTP) provides a real-time data transport for applications, which transmit audio, video or other real-time media. User Datagram Protocol (UDP) is usually chosen for the underlying transport protocol, due to its efficiency and best effort nature. Thus, delivery of RTP packets cannot be guaranteed and RTP packet loss may occur.

Ogg Vorbis is an open source, non-proprietary, and patent free compressed audio format for high-quality audio and music.

The thesis presents a streaming system for high-quality audio based on open and non-proprietary standards. The streaming system contains a server and several clients. RTSP is used for the signaling and Ogg Vorbis encoded audio is transported with RTP. A solution for Ogg Vorbis audio streaming using RTSP is made and an architecture is build for the evaluation of the solution.

Also the properties of Vorbis RTP streams are analyzed, simulations for the more efficent RTP packaging are done, and proposals for the congestion control and packet recovery protocol have been made.

Keywords: Audio streaming, Internet audio, Ogg Vorbis, RTP, RTSP

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Jussi Mutanen
Työn nimi:	Järjestelmä korkeatasoisen äänen reaaliaikaiseen siirtoon Internetissä
Päivämäärä:	20.8.2002 **Sivuja:** 67
Osasto:	Sähkö- ja tietoliikennetekniikka
Professuuri:	S-89
Työn valvoja:	Prof. Matti Karjalainen
Työn ohjaajat:	DI Martti Rahkila

Real-Time Streaming Protocol (RTSP) on Internetissä käytettävissä oleva protokolla, joka muodostaa ja ohjaa reaaliaikaisia mediavirtoja. RTSP on tarkoitettu ohjaamaan useita sekä reaaliaikaisia että tallennettuja lähetyksiä.

Real-time Transport Protocol (RTP) tarjoaa protokollan reaaliaikaisen äänen, kuvan tai muun tiedon siirtoa varten. User Datagram Protocol (UDP) on yleensä valittu siirtoprotokollaksi tämän tehokkuuden ja nopean palvelun ansiosta. Tällöin RTP pakettien perille menoa ei voida kuitenkaan taata ja RTP pakettien hukkuminen on mahdollista.

Ogg Vorbis koodausmenetelmä perustuu avoimeen koodiin ja se ei sisällä patenttioikeuksia. Se soveltuu hyvin korkeatasoisen äänen ja musiikin pakkaamiseen.

Tämä diplomityö esittelee järjestelmän korkeatasoisen äänen siirtoon Internetissä käyttäen avoimia standardeja. Järjestelmä sisältää palvelimen ja useita asiakkaita. RTSP:tä käytetään merkinantoon ja Ogg Vorbis muotoon pakattu ääni siirretään RTP:n avulla. Työssä esitetään ratkaisu Ogg Vorbis muotoon pakatun äänen siirtoon käyttäen RTSP protokollaa ja työssä esitetty ratkaisu arvioidaan toteutetun arkkitehtuurin avulla.

Työssä analysoidaan myös Vorbis RTP paketoinnin ominaisuuksia, tehdään simulaatioita tehokkaamnan RTP paketoimisen aikaansaamiseksi ja esitetään ratkaisuja sekä ruuhkanhallinnan että pakettien uudelleenlähetyksen protokollaksi.

Avainsanat: Audio streaming, Internet audio, Ogg Vorbis, RTP, RTSP

Acknowledgements

This Master's thesis has been done for the department of Research, Standardization, and Technology in Nokia Networks. I want to thank my patient boss Yrjö Raivio for the opportunity to work in a field of high-quality audio and acoustics. The NSR team shared guidance and knowledge openly and gave me a lot of encouraging comments how to improve the thesis. Inspiring lessons about the current Internet multimedia services were also kept. Those were a great help, thanks team!

The use of previous work done by Jari Selin was an essential part of high-quality audio streaming. Mikko Vainikainen was also a valuable person for the signaling protocol implementation.

I would also like to thank Martti Rahkila for being my instructor. His visions and knowledge of Internet audio ensured that the proper content was included in this thesis. My gratitude also goes to Professor Matti Karjalainen how took this kind of unusual work under his supervision.

Finally, I would like to thank my family and especially my dad who bravely came to see the presentation.

Otaniemi, August 20, 2002

Jussi Mutanen

Contents

Abbreviations

AAC	Advanced Audio Coding
ACM	Association for Computer Machinery
ADIF	Audio Data Interchange Format
ADSL	Asymmetric Digital Subscriber Line
ADTS	Audio Data Transport Stream
ARQ	Automatic Repeat Request
ASCII	American Standard Code for Information Interchange
ASF	Advanced Streaming Format
AVI	Audio Video Interlaced
AVT	Audio/Video Transport
CD	Compact Disc
CSRC	Contributing source
DAB	Digital Audio Broadcast
DPCM	Differential PCM
DRM	Digital Rights Management
DVD	Digital Versatile Disc
EMD	Electronic Music Distribution
FEC	Forward Error Correction
FIFO	First In First Out
GRPS	General Packet Radio Service
GPS	Global Position System
HTTP	Hypertext Transfer Protocol
IESG	Internet Engineering Steering Group
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPR	Intellectual Property Rights
ISMA	Internet Streaming Media Alliance

ISO	International Organization for Standardization
ITU	International Telecommunication Union
LAN	Local Area Network
MDCT	Modified Discrete Cosine Transform
MIDI	Musical Instrument Digital Interface
MIME	Multipurpose Internet Mail Extension
MMUSIC	Multiparty Multimedia Session Control
MPEG	Moving Pictures Experts Group
MS	Media Server
MSS	Media Services Streaming
MTU	Maximum Transmission Unit
OMG	Object Management Group
OS	Operating System
PC	Personal Computer
PCM	Pulse Code Modulation
PCMA	A-law PCM
PCMU	μ-law PCM
PDA	Personal Digital Assistant
PT	Payload Type
RFC	Request for Comments
RIAA	Recording Industry Association of America
RL	Run Length
RSVP	Resource Reservation Protocol
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
RTT	Round Trip Time
SAP	Session Anouncement Protocol
SDMI	Secure Digital Music Initiative
SDP	Session Description Protocol
SIP	Session Initation Protocol
SIPPING	Session Initation Protocol Proposal Investigation
SMIL	Synchronized Multimedia Integration Language
SMTP	Simple Mail Transfer Protocol
SNR	Signal to Noise Ratio
SSRC	Synchronization source

TCP	Transmission Control Protocol
THD	Total Harmonic Distortion
TNS	Temporal Noise Shaping
UA	User Agent
UDP	User Datagram Protocol
UEP	Un-Equal Protection
UI	User Interface
UML	Unified Modeling Language
URI	Uniform Resource Indentifier
URL	Uniform Resource Locator
VoIP	Voice over IP
VQ	Vector Quantization
WLAN	Wireless LAN
WG	Working Group
WWW	World Wide Web

List of Figures

List of Tables

Chapter 1

Introduction

New services are created rapidly in the Internet. This happens because an “Internet year” is said to be equivalent to approximately 3 months in the real world. A possibility to download music to one’s home was a new thing a couple of years ago but not anymore. Audio and video took only 2 % of the whole Internet traffic in 2000, but is expected to be around 6 % in 2003. [51] [53]

There are many reasons for this strong increase of audio and video traffic. A faster Internet access has become common, radio is entering to digital age where stations start operating in the web, soundcards and speakers are nowadays a basic feature of a new PC, and high-quality compression technologies have boosted the usage of encoded audio in the Internet. [29]

Multimedia is a combination of two or more types of communication media. The communication media may contain information in a certain format, e.g. pictures, text, music etc. However, widely accepted standards for the public audio and video format have not been created. Several dominant and commercial closed ’standards’ are available, e.g. Microsoft Windows Media Audio and RealNetworks G2. These commercial ’standards’ are often closed by several patents and are therefore entitled to royalties.

A conventional compact disc (CD) is typically sampled at 44,1 kHz using pulse code modulation (PCM) with a 16 bit sample resolution. For stereo music this results in a 1,41 Mbit/s of bandwidth. However, this bandwidth requirement is too much for the today’s Internet. Thus, audio stream has to be compressed. It can be compressed down to 100 kbit/s without any subjective degradation in the perceived audio quality [51]. MP3 is the most well known compressed audio format for hiqh-quality music and it has been spread rapidly all over the Internet. In this thesis high-quality audio refers to CD-quality audio or better.

A system for real-time high-quality audio transmission between two parties in the In-

ternet is presented in this thesis. Live audio streams can be established and controlled in real-time. This kind of system could be used in fast local networks requiring accurate management of music. In a centralized server architecture music management is easier than in a distributed system.

1.1 Music as a media component

Sound and voice are fundamental parts of human rituals and normal living. Music has always been one way to communicate across linguistic and cultural barriers. Sound, voice, and music are the core elements of understanding people. If language is unknown, speech may not be understood but music is always approachable. Music has emotional dimensions and music is intellectually stimulating. These are the key elements why music could be significant application in the Internet. [4]

Computer technology has developed very fast over the past five years. Today's processors have enough computing power for encoding and decoding music in real-time. The MPEG standard specifications are available for everyone and a low bit rate high-quality audio coding can be performed at low costs [51]. Faster Internet access has made the distribution of digital audio possible. Audio CD-writers have became very common and highlighted the digital audio as a core component in the multimedia Internet. [7]

Two types of media delivery/playback exist, static and dynamic [51]. The content can be media parameters or data. The static one downloads the content from the server to the client first and starts playing the stream after the last bit has arrived. The dynamic, i.e. streaming one, starts playing the music after the first few seconds when the stream has arrived, while the rest of the content is still being downloaded. The media content is transferred on-line as a flow of packets. Each packet contains a timestamp when the packet was actually sampled. The packets are sent in chronological order. In the Internet, packets can, however, arrive to the destination in any order. The receiver gathers all the received packets and rearranges them according the timestamps before the reproduction and playing of the audio stream, e.g systems using RTP media stream.

1.2 Internet audio problems

Subtle shifts in temporal music are very easy to recognize. Musicians are extremely sensitive to these shifts and this kind of accuracy would be desirable in the Internet audio aswell. Audio is the most time-critical element of the multimedia components. Audio streaming differs substantially from the real-time voice over IP (VoIP) phone call. Streaming is usually uni-directional, while in the VoIP phone call up- and downlink streams are reserved.

Latency and bandwidth Time delays are caused by several factors. Audio material is usually stored in advance to a hard drive in an encoded format to keep the latencies only in few milliseconds. The high-quality audio encoding is the most time consuming process. In the MPEG standard, the minimum encoding delay for MP3 is quoted as 59 ms, but in practise it can be 3-4 times more depending on sample resolution and bit rate [12]. Thus, the real-time encoding should be avoided. In the audio playback or record process, operating system (OS) latencies should be taken into account. Routing and switching latencies are major issues when congestion occurs in the network. Transmission latencies depend on the underlying network properties.

The media transport protocol has to ensure the delivery of data to the receiver with low latency, low error and loss rate, and without breaks in continuity. Broadband systems, which are capable of transmitting rates greater than the needed one, are a vital condition for high-quality audio streaming. The lack of bandwidth increases the latencies and reduces the sound quality. [4]

Multimedia content creation Creating a multimedia presentation is a huge task when compared to typing text. Therefore, to produce content for multimedia with a good or even moderate quality is very demanding. Ready made multimedia content would be perfect for streaming, but the question is not so easy due to digital rights management issues. Copyright protection is a major interest in todays business world. People are used to treating the Internet as a free zone, but the Intellectual Property Rights (IPR) legislation is still valid and used.

Multimedia annotation and management Standardization of multimedia session control and description protocols are still ongoing, but transport protocols are very well specified. Standards must be ready before multimedia streaming becomes economically viable.

Nowadays a text search is very powerful, but just some multimedia content is searchable. Synchronized Multimedia Integration Language (SMIL) [52] is designed for multimedia authoring. Multimedia Content Description Interface (MPEG-7) [1] promises a generalized annotation schema when it arrives.

In 2001 Association for Computer Machinery (ACM) had a panel discussion with the topic: *Is Streaming Media becoming Mainstream?* [8]. It was noticed that in order to make multimedia annotation part of our real life there is a need for a global positioning system (GPS) information in the digital cameras. Advanced recognition and filtering methods are also needed for the separation of multimedia content. Personal pictures, news, or ads should be quickly separable to corresponding categories. Business models are needed to speed up the growth of multimedia annotation.

All these issues will make the search, filtering and authoring capabilities of the multimedia content even better. The query-string-based text search could be replaced with an evolution search based on auditory and musical content [4]. Also short summaries of musical pieces, i.e. “audio thumbnails”, would help the search [2].

1.3 Current streaming services

Current streaming services are based on the ready made content such as television or radio programs. It is not yet economically resonable to design streaming media content for the Internet use only. Some of the Internet Radio service providers are **Icecast** [1], **Live365** [2], and **RealAudio** [3], but also TV news can be watched e.g with RealNetworks RealPlayer. A streaming video consumes even more network bandwidth than the radio stream, so video content is only suitable for very broadband networks.

Distant learning with multimedia content and interactive exercises will be the next phase in educational teaching techniques. The Internet provides an opportunity to make cost effective distance learning methods. Lecture material has been distributed in an electrical format through the web pages, but nowadays also the lectures can be viewed in a streaming format. [15] [34] [3] [43]

1.4 Future

Handheld devices will become more and more popular, and the mobile world will merge with the Internet some day. Wireless Local Area Networks (WLANs) arrive to homes and enable high-quality audio streaming. A streaming video will be available a few years later. Available Internet bandwidth increases and broadband services will get new users. Because of the use of wireless services low bandwidth services won’t vanish. Preconditions for successful broadcasting are a flat charging rate per month or some other form of cheap Internet access. Improved audio and video coding schemes for low bit rates are needed as well as pre-installed multimedia players or viewers to every new device sold. However, quality of service (QoS) technologies should be used in the Internet before the high-quality streaming becomes economically viable. [21] [51]

[1] http://yp.icecast.org
[2] http://www.live365.com/home/index.live
[3] http://realguide.real.com/tuner

Chapter 2

Problem Statement and Evaluation Criteria

This chapter formulates the problem for this thesis and defines criteria for the evaluation of the solution. In addition to the evaluation criteria also a structure of this thesis is presented. After the first chapter the reader should have enough knowledge to understand the problem, the field where it belongs to, and maybe have some own opinions of the problem.

2.1 Formulation of the problem

An object for this thesis is audio streaming with the Ogg Vorbis media format. Figure 2.1 illustrates the problem that has to be solved before the Ogg Vorbis audio streaming is possible. The problem is the reliable distribution of the Ogg Vorbis codebooks in unicast environment.

A set of codebooks is required to decode a Vorbis stream. These codebooks are allowed to change for each logical Ogg Vorbis bit stream of each physical stream, for example, for each song encoded in a radio stream. The codebooks must be completely intact and the stream cannot be decoded with an incomplete or corrupted set of codebooks. The codebooks packet is around 4 kilobytes in size. Real-time Transport Protocol (RTP) is a standardized way to transport the media content, but because of its unreliable nature, RTP cannot be used for the transmission of the codebooks.

2.2 Criteria for evaluation

Streaming system solution is evaluated according to the following criteria.

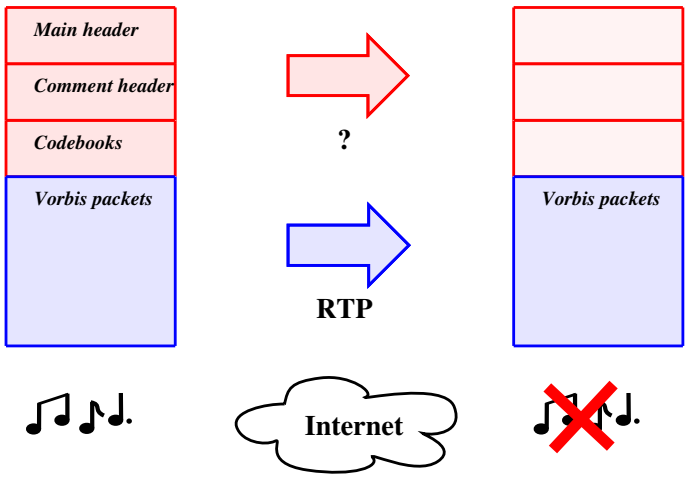

Figure 2.1: The problem in the thesis is the reliable distribution of the Ogg Vorbis codebooks.

Architecture: Modularity. Architecture should be specified in a modular way so that it can be extended in the future. The specified protocols and use of clear interfaces are main issues for achieving a good result in modularity.

Implementation: Open interfaces. The implementation should be done using standardized IETF protocols. The use of proprietary protocols or codecs should be avoided.

Quality: CD-quality audio. A system should be able to transfer a high-quality audio without any significant perceived quality loss. This means that the music should be able to be listened with the same quality as in the home stereo system. No breaks in music playback are accepted.

2.3 Structure of the thesis

The structure of this thesis is presented in figure 2.2. The thesis is divided into four phases.

The first phase is an introduction, which is split into two chapters. Chapter 1 presents briefly an overview of the field of Internet audio. In the second chapter an object for this thesis is given and key criteria for the evaluation of the solution are defined. Also a clear structure of the thesis is presented.

In the second phase Internet audio is expanded on technical aspect. An overview of the field of IETF multimedia architecture and a real-time audio transmission over the Internet are discussed in chapter 3. Chapter 4 presents properties of perceptual audio coding, content

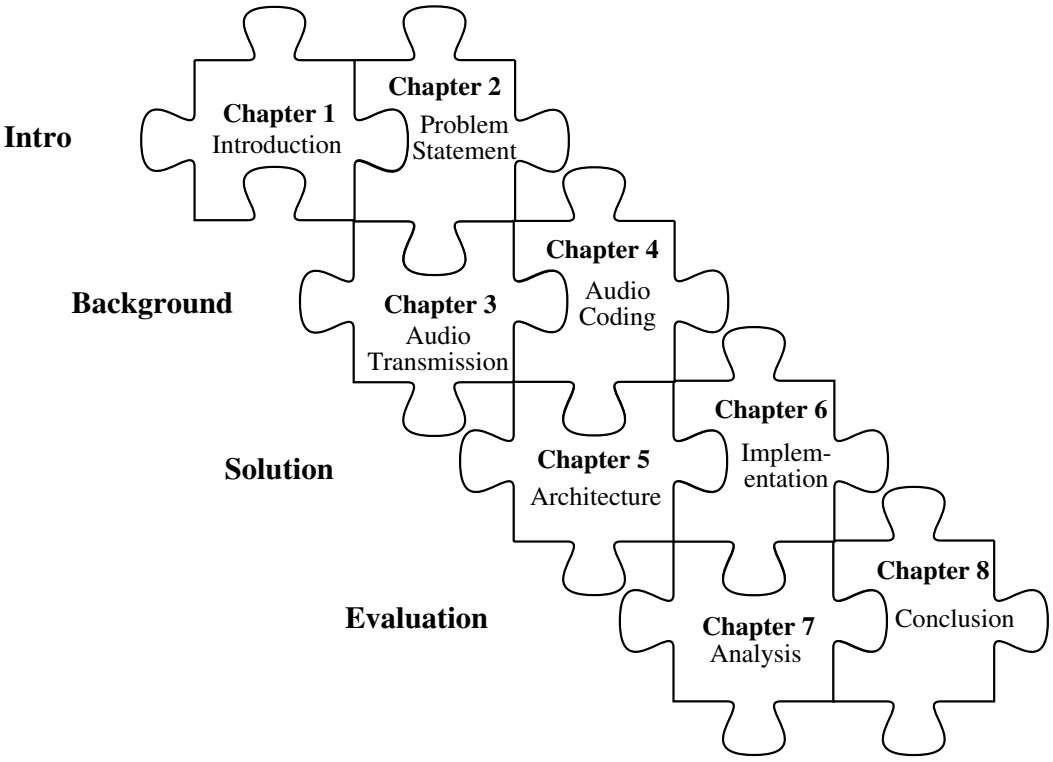

Figure 2.2: Structure of this thesis.

media types, and different audio coding standards.

The solution phase contains two chapters. Chapter 5 discusses previous work and existing software and presents a solution for the streaming system. No programming skills are required to understand the architecture of the solution. An implementation specific view of the architecture is described in chapter 6. This chapter assumes that the reader has some basic programming knowledge and skills to understand the Unified Modeling Language (UML) flow charts and diagrams.

In the evaluation phase the quality of the solution is analyzed in chapter 7. The last chapter 8 presents the future work and concludes this thesis.

Chapter 3

Audio Transmission System

This chapter presents the Internet Engineering Task Force (IETF) multimedia architecture and general properties of real-time media transport protocols. It contains an introduction to the IETF standardization field and explains the IETF protocols used in this thesis. The real-time media transport is a tricky issue, where packet loss can't be totally avoided and therefore some repair mechanism should be implemented to achieve better sound quality. This chapter concentrates on the media transport and especially on the properties of Real-time Transport Protocol (RTP). Audio coding and different audio standards are discussed in the next chapter.

3.1 Internet Engineering Task Force (IETF)

Internet standardization is done by the Internet Engineering Task Force (IETF) [1]. It is a community of experts where, basically, even an individual can have influence. Standardization is controlled by the Internet Engineering Steering Group (IESG). The IETF is divided into areas and further to Working Groups (WG). Meaningful WGs for this thesis are all from the transport area. Audio/Video Transport (AVT) WG is responsible of standardization of RTP, Multiparty Multimedia Session Control (MMUSIC) WG is envolved in specification of Session Description Protocol (SDP), Session Initation Protocol (SIP) WG continues the development of SIP, and Session Initation Protocol Proposal Investigation (SIPPING) WG develops requirements for SIP extensions.

The IETF multimedia data and control architecture [47] is presented in figure 3.1. It has protocols for real-time media data transport and quality of service (QoS) feedback, controlling the delivery of streaming media, and multimedia session description and initiation. Hyper Text Transfer Protocol (HTTP) is used in WWW and Simple Mail Transfer Protocol

[1] http://www.ietf.org

(SMTP) is used in e-mail. Both HTTP and SMTP traffic is carried with Transmission Control Protocol (TCP) so the content delivery is guaranteed. SIP is used in Internet telephony applications and Real-Time Streaming Protocol (RTSP) in streaming services. These two protocols can work on top of TCP or Used Datagram Protocol (UDP). UDP is meant for real-time and low latency applications, e.g Session Announcement Protocol (SAP) enables multicast session announcement, Resouce reServation Protocol (RSVP) reserves network resources and provides QoS, and RTP and Real-time Transport Control Protocol (RTCP) handles real-time media transport.

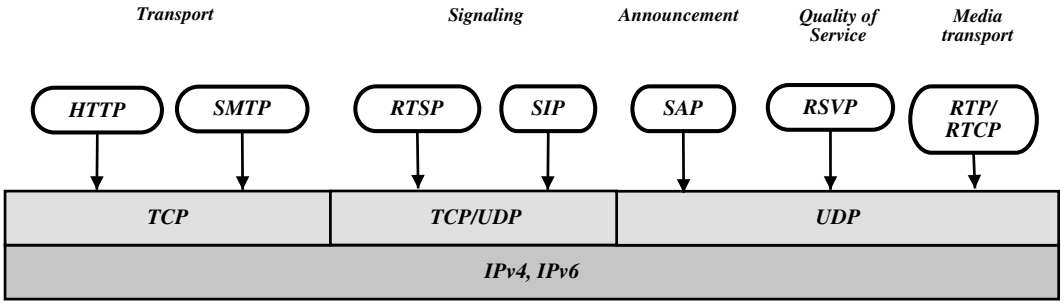

Figure 3.1: IETF multimedia protocol stack.

Transferred media content is specified with Multipurpose Internet Mail Extension (MIME) [18]. MIME was specified for the transmission of non-ASCII data through e-mail. A MIME message contains information of the media type and the encoding format [19]. Binary data can be converted to 7-bit ASCII representation using *base64* encoding. An example of MIME message is presented in figure 3.2.

```
From: Jussi.Mutanen@nokia.com
To: Kehveli.Ketale@foo.bar
MIME-Version: 1.0
Content-Type: image/gif
Content-Transfer-Encoding: base64

   ...data for image...
```

Figure 3.2: An example of MIME message.

3.1.1 Transmission Control Protocol and User Datagram Protocol (TCP, UDP)

Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are the dominant network protocols in the Internet. TCP [38] is connection oriented, reliable, and full-duplex

protocol with acknowledgment and retransmission capabilities. UDP [37] is connectionless, unreliable protocol with no flow control mechanism. [10].

3.1.2 HyperText Transfer Protocol (HTTP)

HyperText Transfer Protocol (HTTP) [13] is an application level, text-based protocol, which uses TCP as a transport protocol. It is a stateless request/response protocol and is used in web servers and browsers. In the Internet each web page is identified by Uniform Resource Locator (URL) [5] [2]. A HTTP URL has the following form:

http: // hostname [: port]/path[;parameters][? query],

where brackets denote an optional item. An example of the HTTP URL could be:
`http://music.nokia.com:8000/bob_james/joy_ride.sdp`.
It specifies the used protocol *http*, the host name *music.nokia.com*, the port number *8000*, and finally the target file */bob_james/joy_ride.sdp*.

HTTP is a request/response protocol where a client starts a transaction by sending a HTTP request (HTTP methods listed in table 3.1) to the server. Possible methods are e.g GET, POST, PUT, OPTIONS, etc. The server acknowledges the request by sending back a response. Possible data is transported in the payload of the request or the response. The response class is defined by the first bit of a response status code. Response classes are listed in table 3.2.

An example of HTTP transaction is presented in the first part of the figure 3.3. A client sends a HTTP GET request to the server. The GET request contains request URL identifying the resource to be fetched. The server accepts the request by sending back 200 OK response containing the requested data in the payload.

Table 3.1: HTTP methods.

Method	Description
CONNECT	Used in proxy for tunneling
DELETE	Deletes the information under the supplied Request-URI
GET	Retrieves the information
HEAD	Obtains the meta information about the entity
OPTIONS	Queries the available methods
POST	Post the information
PUT	Stores the information under the supplied Request-URI
TRACE	Invokes a remote, application-layer loop back

[2] A URL is a specific type of general Uniform Resource Identifier (URI).

Table 3.2: HTTP and RTSP status code classes.

Status code	Class	Explanation
1xx:	Informational	Request received, continuing process
2xx:	Success	The action was successfully received, understood, and accepted
3xx:	Redirection	Further action must be taken in order to complete the request
4xx:	Client Error	The request contains bad syntax or cannot be fulfilled
5xx:	Server Error	The server failed to fulfill an apparently valid request

3.1.3 Session Description Protocol (SDP)

Multimedia sessions are described with the Session Description Protocol (SDP) [22]. It is used for session announcement, session invitation, and for session initiation. A session description of SDP describes the capabilities of a system and possibly provide a choice between number of alternatives. Every session description has one or more media descriptions. The format of SDP is described in table 3.3.

SDP is a text based protocol and all the information in it should be understood by the end user. SDP does not specify a transport protocol so it is intended to be used with different transport protocols like SAP, SIP, RTSP, with e-mail using MIME extensions, or with HTTP. With a session description the client announces the session capabilities that it supports, e.g. codecs, transport protocols, control protocols etc. An example of the SDP usage in a basic SIP phone call is described in section A.1.

SDP next generation (SDPng) [28] is now in specification phase and it will offer a solution to the offer-answer model session description negotiation. In a offer-answer model both the supported codecs and the used codecs have to be defined. With SDP only supported codecs can be announced, which in a negotiation of SIP phone call raises troubles. If more than one supported codecs exist after the negotiation there are no proper way to decide which one should be used.

3.1.4 Real-Time Streaming Protocol (RTSP)

Real-Time Streaming Protocol (RTSP) [46] establishes and controls time-synchronized streams of continuous media. RTSP can be used on top of TCP or UDP, but actual data is not transferred with RTSP. The RTSP specification does not specify the media transport protocol, but RTP is a common choice.

RTSP has a similar syntax and operation like HTTP. The biggest differences are that RTSP has different methods (RTSP methods are listed in table 3.4) and a different protocol identifier is used. E.g. RTSP URL could be:
`rtsp://music.nokia.com/bob_james/joy_ride/take_me_there.ogg`.

Table 3.3: SDP session description format, optional parameters are in gray.

Session description	
v=	protocol version
o=	owner/creator and session identifier
s=	session name
i=	session information
u=	URI of description
e=	email address
p=	phone number
c=	connection information
b=	bandwidth information
z=	time zone adjustments
k=	encryption key
a=	zero or more session attribute lines

Time description	
t=	time the session is active
r=	zero or more repeat times

Media description	
m=	media name and transport address
i=	media title
c=	connection information
b=	bandwidth information
k=	encryption key
a=	zero or more media attribute lines

In this URL protocol is *rtsp*, host name *music.nokia.com*, the standard RTSP port number *554* is used, and the wanted target file is */bob_james/joy_ride/take_me_there.ogg*.

Response status codes and classes are almost the same as in HTTP (listed in table 3.2). A RTSP server keeps a record of the state of the RTSP session in almost all cases as opposed to the stateless nature of HTTP. In RTSP both the server and the client can issue requests. In HTTP data is carried in a payload, but in RTSP out-of-band by a different protocol.

RTSP does not specify the methods that have to be implemented in a RTSP server. Thus, all the RTSP servers don't support the same functionality.

An example of RTSP streaming is presented in figure 3.3. Before a client can establish the streaming session it somehow has to get the session description. In this example the client gets the session description from the web server using HTTP protocol. According to the information in the session description the client sends a RTSP SETUP request to the

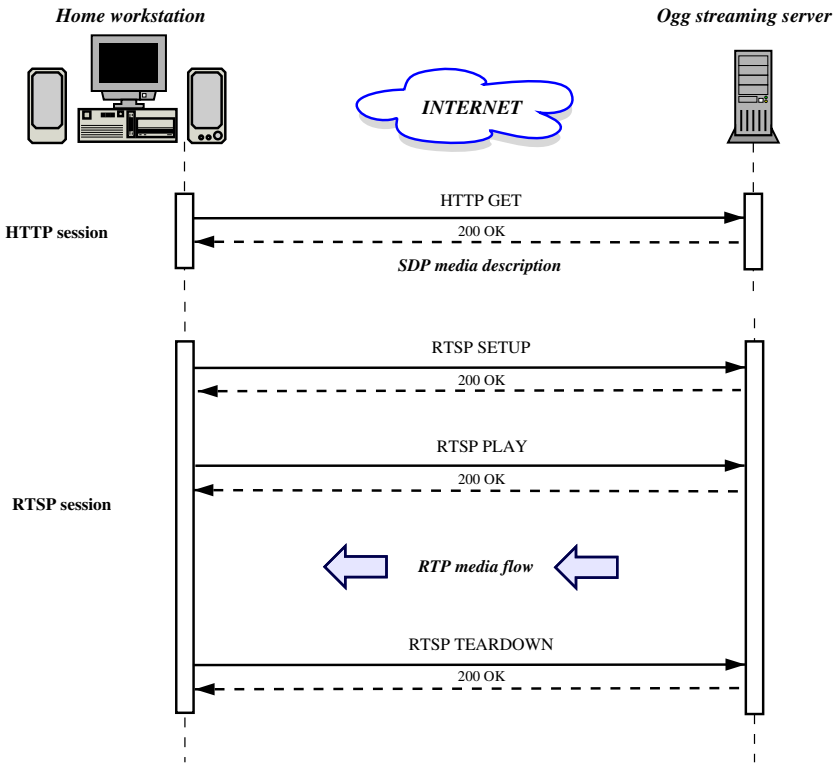

Figure 3.3: RTSP streaming.

streaming server. The server informs the client with 200 OK response to indicate that the stream has been prepared succesfully. The client starts the streaming with a RTSP PLAY request and ends the streaming session with a RTSP TEARDOWN request.

Table 3.4: RTSP methods.

Method	Description
DESCRIBE	Retrieves the description of a presentation
ANNOUNCE	Posts the description of a presentation
GET_PARAMETER	Retrieves the value of a parameter
OPTIONS	Queries the available methods
PAUSE	Stream delivery is halted temporarily
PLAY	Starts sending data
RECORD	Starts receiving data
REDIRECT	Informs to connect another server location
SETUP	Specifies the transport mechanism
SET_PARAMETER	Requests to set the value of a parameter
TEARDOWN	Stops the stream delivery and frees the resources

3.1.5 Session Initiation Protocol (SIP)

Session Initiation Protocol (SIP) [23] is for creating and managing sessions, where data is sent between two or more participants. These sessions include Internet multimedia conferences, Internet telephone calls, and multimedia distribution. By the time of writing this thesis the standardization of SIP is still ongoing. SIP is not the main topic of this thesis so only a brief introduction of SIP is given.

SIP is an application-layer signaling protocol. SIP is not a whole communications system, but it is a part of IETF multimedia data and control architecture (see figure 3.1). SIP is used together with other protocols like SDP and RTP. RTP will be discussed in next section. SIP handles the session establishement and control, SDP defines the session parameters, and RTP provides the media delivery.

SIP has similarities to HTTP. Just like HTTP, SIP is a text-based protocol with a client-server architecture. A client sends requests to the server and the server answers with suitable responses. SIP transaction can consist of more than one requests and responses. SIP methods and response codes are shown in tables 3.5 and 3.6. Some of the SIP methods are still on the Internet Draft (I-D) phase and haven't yet been accepted to the standard. SIP transactions can also be established through persistent TCP connections.

Table 3.5: SIP methods.

Method	Description	Status
INVITE	Starts a new session	RFC [23]
MESSAGE	Sends an instant message	I-D [9]
OPTIONS	Queries the available methods	RFC [23]
REGISTER	Registers the contact information	RFC [23]
ACK	Acknowledgment of responses	RFC [23]
PRACK	Acknowledgment of provisional responses	I-D [41]
CANCEL	Cancels the pending request	RFC [23]
BYE	Terminates the session	RFC [23]
INFO	Information exchange	RFC [11]
REFER	Call transfer	I-D [50]
SUBSCRIBE	Subscription to an event	I-D [39]
NOTIFY	Event notification	I-D [39]

Every SIP user or service has a SIP identity, which is identified by a SIP URL. The SIP URL scheme is just like a normal e-mail URL except of the different protocol identifier: `sip:jussi.mutanen@nokia.com`.

SIP network There are two kind of network elements in a SIP system, user agents and servers. SIP user agent provides a session initialization and control functionality to the end

users. SIP network servers offer location, call routing and other supplementary services to the SIP user agents. A basic SIP call flow example is described in section A.1.

Table 3.6: SIP status code classes.

Status code	Class	Explanation
1xx:	Informational	Request received, continuing process the request
2xx:	Success	The action was successfully received, understood, and accepted
3xx:	Redirection	Further action must be taken in order to complete the request
4xx:	Client Error	The request contains bad syntax or cannot be fulfilled at this server
5xx:	Server Error	The server failed to fulfill an apparently valid request
6xx:	Global-Failure	The request cannot be fulfilled at any server

3.1.6 Real-time Transport Protocol and Real-time Transport Control Protocol (RTP/RTCP)

Real-time Transport Protocol (RTP) [45] provides a real-time data transport for applications transmitting audio, video or other real-time media. RTP is independent of the underlying transport protocol, but usually UDP is chosen because of its efficiency and best effort nature.

An RTP session is established between the end users and the session is identified by the pair of participants destination addresses. In a media transport the media itself is encoded, fragmented into small RTP packets, and sent through the network. Every RTP packet has a RTP header, which has all needed information to reconstruct the media stream in the receiver end. In the RTP header (see figure 3.4) only the gray parts are present in every RTP packet. After the RTP header comes the RTP payload, which contains the encoded media content.

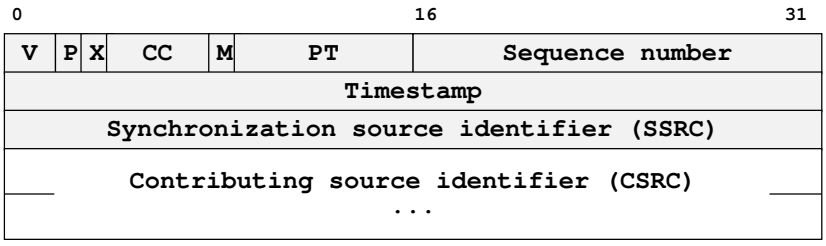

Figure 3.4: RTP header.

Version (V) Version number, specified to be two by RFC 1889.

Padding (P) If set, the last byte of the payload indicates the number of bytes to be ignored from the end of the payload.

Extension (X) If set, fixed header extension will follow.

CSRC count (CC) Number of contributing source identifiers that follows the fixed header.

Marker (M) With this bit frame boundaries can be marked in a packet stream. For voice a marker bit is used to mark the beginning of a talk spurt but the function of the marker bit is defined by the used RTP profile.

Payload Type (PT) Identifies the encoding format for the payload. Payload types can be statically defined in the RTP profile or dynamically using SDP.

Sequence number The sequence number is increased every time by one when the RTP packet is sent. The receiver can detect the RTP packet loss from the sequence numbers.

Timestamp Sampling instant of the first byte of the payload data.

SSRC A synchronization source identifier (SSRC) is selected randomly. Different RTP session can't have same SSRC identifiers.

CSRC List of contribution source identifiers (CSRC) is used in RTP sessions when RTP mixers are in the middle of the connection. Number of CSRCs are identified by the CC field.

Real-time Transport Control Protocol (RTCP) [45] is an accompanying protocol for exchanging control messages of the RTP stream. The primary function of RTCP is to provide feedback of the quality of the data transmission. RTCP provides flow and congestion functions to RTP. Minimal session control information can also be distributed with RTCP among participants. Different RTCP packet types are described in table 3.7.

Table 3.7: RTCP packet types.

Type	Description
SR	Sender report for transmission and reception statistics from active senders
RR	Receiver report for reception statistics that are not active senders
SDES	Source description items
BYE	End of participation
APP	Application specific functions

RTP profile The RTP profile describes how audio and video data may be carried within RTP. RFC 1890 [44] contains profiles for the common audio and video codecs, e.g. PCMA and PCMU. For specific codecs, e.g. Ogg Vorbis, a sepate document describes the media transport over RTP.

3.2 Bandwidth requirements: unicast vs multicast

A lack of bandwidth is the main problem of the Internet audio, especially with high-quality audio broadcasting [51]. In broadcast services, such as TV or radio programs, the same data is sent to many receivers simultaneously. The multicast backbone (MBone) [3] was created, because all the backbone routers in the Internet don't support it. MBone tunnels the multicast packets in ordinary IP datagrams. In this way multicast can be used in heterogeneous networks, where multicast and unicast routers can co-exist. On-demand streaming services can also be implemented in a multicast environment by using an Un-Equal Protection (UEP) [54] coding scheme.

3.3 Solutions for the repair of streaming media

In the Internet packet loss can occur during the real-time media transport. Packet recovery is a fundamental problem of Internet audio and is usually implemented as an "optional" protocol. Options for repairing streaming media are discussed in RFC 2354 [36]. TCP is unsuitable because of it's high latency, which is caused by TCP's network-friendly congestion control algorithms. For greedy, high-bandwidth real-time applications, UDP is thus the only choice for retransmission. [53]

In general, a tradeoff between audio quality and latency has to be made. For low-latency applications it is difficult to implement a perfect packet loss recovery protocol. There are four candidates for the sender side recovery technique: redundant transmission, retransmission, interleaving and forward error correction (FEC). Recovery techniques in the receiver side are much simpler. Lost packets can be replaced by noise, silence or by the previous packets.

Knowledge of the loss characteristics has to be known before implementing a packet recovery protocol. Measurements in the Mbone environment proved that in a large conference some receivers will experience packet loss. The experienced packet loss was in the range of 2-5%. There hasn't been a clear answer to the question whether a single packet loss or a burst losses occurs more often. Mbone measurements showed that the vast majority of the losses are of single packets. However, burst losses occured more often than it would have been expected from a purely random process. [36]

In radio or television broadcast stream latency is not so important as the received quality. Interleaving, retransmission, or FEC are the appropriate choices for the recovery techniques in this kind of non-interactive and uni-directional transmission. In interactive applications – session is defined as an interactive if end-to-end delay is less than 250ms [4] – the use of

[3] http://www.savetz.com/mbone/

interleaving or retransmission is out of question because of the high latency. Thus FEC is the best choice for the packet recovery.

3.3.1 Forward error correction

In forward error correction (FEC) [36] repair data is added to the stream so that the packet loss can be recovered without any notification to the sender. Two types of FEC exists. In a media independent FEC redundant data is transmitted in separate packets but in media dependent FEC knowledge of the compression scheme can be employed and more efficient repair can be achieved.

FEC redundant data calculation follows the procedure where a block code or a parity code is applied to a set of k packets to generate a set of $n > k$ packets. Receiver only needs to receive any k-packet subset of the resulting n-packet block in order to recover perfectly the k original packets. FEC can also be used for the most significant bits of data rather than applying it over the entire packet. [53]

Advantages of FEC are that the sender doesn't need to know anything about the packet losses and no retransmission is needed. FEC is suitable for low latency interactive applications, where large end-to-end delays cannot be tolerated. Negative aspects of FEC are that bandwidth usage is increased due to the redundant data and FEC cannot recover from large bursts of packet loss.

3.3.2 Automatic repeat request

Automatic repeat request (ARQ) is based on a retransmission of the lost packet. Retransmission can be handled with some other protocol or included into the ongoing transmission. Retransmission introduces a delay of at least three one-way trip times. In addition to the high latency there is a potentially large bandwidth overhead due to multiple data units and also the additional control traffic. ARQ methods are not suitable for real-time applications but have demonstrated the effectiveness of repair in high-bandwidth uni-directional music streaming. [53]

3.3.3 Interleaving

Interleaving could be used when the unit size is smaller than the packet size and the end-to-end delay is not so important. Interleaving means that the units are resequenced before the transmission. The first packet could contain units 1, 5, 9, 13; the second packet would then contain units 2, 6, 10, 14; and so on. In this way a loss of a single packet will result in multiple small gaps in the reconstructed stream.

One of the advantages of the interleaving is that the bandwidth requirements of the stream is not increased. Interleaving is also media and codec independent. However, increased latency limits the use of interleaving only to non-interactive applications. [36]

3.3.4 Congestion control

Congestion control is needed if a retransmission recovery scheme is used. If the packet loss continues during the retransmission, the congestion becomes even more severe, leading to a further increase of packet loss. TCP implements an effective congestion avoidance algorithm. It will back off and decrease the transmission rate whenever congestion is encountered.

In real-time applications the transmission interval can't be modified. However, the packet size could be decreased on the fly if a layered coding technique has been used. Idea of the layered congestion control system is to reduce the sampling resolution when congestion is encountered. In this way bandwidth usage of the media stream is decreased but the frame rate is still kept constant.

The layered coding congestion control is better than nothing. A continuous playback stream can be achieved, but with a reduced sampling resolution. There has been a study for the threshold: For every 4% increase in received packet loss reduce the sampling resolution by 1 bit [53].

3.4 Quality of service

Definition of the quality of service (QoS) [51]:

> *Network capability that protects the integrity, end-to-end predictability, and bandwidth utilization of data transmission.*

QoS becomes much more important when transmitting real-time audio and/or video multimedia content than just plain text or still pictures. QoS is needed in the future mobile networks but it is not widely used in the Internet yet. QoS can be divided into two parts, Integrated Services (IntServ) and Differentiated Services (DiffServ).

IntServ reserves the needed bandwidth from the whole end-to-end network connection. IETF has specified Resource Reservation Protocol (RSVP) [25] for integrated services, which provides admission and policy control. QoS for each packet is determined by the packet classifier. So far RSVP is not used in backbone routers because of denial-of-service threats and the additional state in the backbone routers, but RSVP could be used in local and secure networks.

DiffServ [6] is based on a packet classification. Packets are measured and compared to the traffic class. If the packet doesn't fit in to this traffic shape, the router will use some algorithm for queuing or dropping the packet.

3.5 Legal aspects of digital audio

Piracy has always been a problem in the music industry. In a digital world it is easy to produce an exact copy and distribute it over high-speed connections. Recording Industry Association of America (RIAA) is one of the companies in Secure Digital Music Initiative (SDMI) [48] forum which is developing and standardizing techniques for protected digital music distribution. However, Digital Rights Management (DRM) issues are out of the scope of this thesis. [42]

3.6 Summary

The IETF multimedia architecture was presented with a deeper look to the signaling and media transport protocols. Several options for repairing streaming media was also discussed. The reader should now have a clear picture of the IETF multimedia architecture and shoud be able to identify problems and difficulties in real-time media transport.

Chapter 4

Audio Coding

Audio coding is its own branch of science. A huge amount of coding technologies exists, from speech coding to parametric synthetic audio. A complete codec review is not the main object of this thesis so only the main properties of perceptual audio coders are briefly described. A much more complete discussion of different audio codecs is presented in Ted Painter's paper *Perceptual Audio Coding* [35].

4.1 Perceptual coding of digital audio

Perceptual audio coding is a lossy compression system where both perceptual irrelevancies and statistical redundancies are removed. Lossy compression technique means that decoded file is not a bit-exact copy of the original. Figure 4.1 presents a generic model of a perceptual encoder. The objective is to extract time-frequency parameters from the input audio which can be then quantized with a proper distortion metric. A lossy distortion metric is achieved in form of masking thresholds from the psychoacoustic analysis. Statistical redundancies can be removed with classical lossless techniques such as differential pulse code modulation (DPCM) or adaptive DPCM (ADPCM). Vector quantization (VQ) could be used when the quantization is a probability density function. Once a compact quantization parameter set is formed remaining redundancies can be encoded with lossless run-length (RL) and entropy (e.g. Huffman) coding techniques. [35].

4.2 Audio quality

In the Internet audio the overall quality depends on the content encoding, the transport protocols, and the network conditions. During the media transport packet losses can occur and the overall latency may vary. Lost or late arrived data packets can easily destroy the

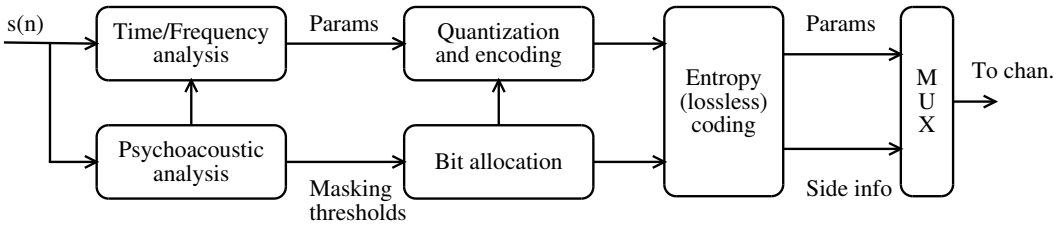

Figure 4.1: Generic perceptual audio encoder.

feeling of continuity of the audio signal. Video streams are however less sensitive to short breaks in content continuity [4]. A high-quality audio stream is better to mute for several seconds rather than play a corrupted segment due to missing packets [53].

4.2.1 Bit rate versus quality

A basic relationship between stream bit rate and audio quality is that lower bit rates lead to higher compression factors and to lower quality of the compressed audio. Bit rate means a rate at which binary digits pass a given point. With higher bit rates probability of signals with any audible artifacts decreases. Encoders have their “sweet spots” where they are designed to work best. The “sweet spots” depend on the codec characteristics, e.g. Huffman codebooks. [7]

There is also a basic tradeoff where to spend the bits available for encoding. Frequency response can be improved, but then no clear sound can be produced at lower frequencies. The best choice is to introduce a fixed bandwidth limitation if a clean reproduction of a full bandwidth signal is not possible.

An acceptable quality for stereo music can be achieved with bit rates as low as 30 kbit/s [51]. With these low bit rates (64 kbit/s for stereo and lower) bandwidth versus cleanness is, however, a question of taste [7]. A good speech quality can be achieved even with bit rates below 10 kbit/s. A normal cable or Asymmetric Digital Subscriber Line (ADSL) modems have bit rates approximately 200 kbit/s and in the Internet2 backbone routers bit rates are as high as 622 Mbit/s.

4.2.2 Common type of artifacts

If the bit rate is too low or wrong parameters have been chosen perceptual audio coders might sound strange. Differences in music might sound a little distorted, but not like harmonic distortion. Music can also sound noisy, but the noise occurs only in a certain frequency range. Roughness is characteristic to the perceptual audio encoder and the roughness is changing characteristics about every 20ms. [7]

Roughness Roughness is especially heard at low bit rates and with low sampling frequencies. Roughness is also called *double-speak*, because of its nature. MEPG-2 AAC tries to avoid this by temporal noise shaping (TNS), which provides noise shaping in time domain and thus enhances the time resolution of the filter bank. [7]

Pre-echoes A pre-echo is a very common artifact for a high-frequency resolution perceptual audio coding system. In the pre-echo a quantization noise is spread out over an entire coding frame. The pre-echo comes audible if the transient sound occurs in the end of the frame. Pre-echoes can be reduced with variable bit rate coding or with the use of short blocks. Pre-echoes are, however, very difficult to avoid. [7]

4.2.3 Quality measures of perceptual audio coding

The quality of a perceptual audio coder is difficult to measure. Measurements are needed to evaluate if emerging techniques are in some sense better than alternative methods. Most often evaluation is done in terms of bit rate, complexity, delay, robustness, and output quality [35]. Measurement of the output quality has turned out to be a significant challenge. Perceptual coders improve the subjective quality by shaping the quantization noise and a signal to noise ratio (SNR) can then be lower than without the noise shaping [7]. Classical objective measures such as SNR or total harmonic distortion (THD) are therefore completely inadequate.

Subjective listening tests are also needed for comparing the performance of different coding algorithms and encoders. It is a time-consuming process and small coding artifacts will became audible only after an extensive training. [7]

ITU-R has standardized a listening test for perceptual high-quality audio codecs. A group of experts are listening A-B-C triple stimulus, where A contains always the reference (uncoded) signal, and B or C a hidden reference and the impaired (coded) signal. After listening to all these three signals a hidden reference should be identified from B or C. The impaired signal is then graded with five-category scale from the very annoying to the imperceptible. [35]

4.3 Audio coding standards

Coding system standards have become less important, since decoders can be implemented as plug-ins and do not require a lot of processing power [51]. Most of these codecs are proprietary and media can be decoded for free, but the use of encoder has to be paid.

People want music to be available, transferable and easy to manipulate [33]. MPEG is recognized as an international standard [29] for audio and video coding. Most widely used

used MPEG standards are MPEG-1/ Audio Layers 2 and 3 and Dolby AC-3. Especially in MPEG-4 extreme low bit rates are enabled for speech coding. [51]

Much of experience and knowledge are needed to implement a good quality MPEG encoder. In these perceptual audio coders a dynamic range is well beyond the equivalent of a 24 bit D/A resolution. With this resolution dynamic range of every known audio source is perfectly retained. Frequency responses are also perfectly even and no coloration in the signal occur due to coding artifacts. [7]

MPEG-1 Layer III

The international audio coding standard for the CD-quality audio Moving Pictures Experts Group (MPEG) of the Internation Standardization Organization (ISO) was defined in 1992. The MPEG-1 standard uses hybrid coding methods including subband decomposition, filter bank analysis, transform coding, entropy coding, dynamic bit allocation, non-uniform quantization, adaptive segmentation, and psychoacoustic analysis. The audio encoding used in popular MP3 files is base on MPEG-1 Layer III standard and mainly patented by the Frauenhofer Institute [1]. MP3 is the de facto standard for storage of compressed audio in WWW. MP3 supports sampling rates of 32, 44.1 and 48 kHz and offers separate modes for mono, stereo, and joint stereo coding. Frequency responses are limited to 16 kHz in MP3 encoders [51]. Available bit rates are 32 - 192 kbit/s for mono and 64 - 384 kbit/s for stereo coding. The target bit rate for the MP3 codec is 1.33 bit/sample (i.e. 128 kbit/s for a stereo signal at 48 kHz). [35] [7]

MPEG-2 AAC

MPEG-2 is a multichannel (3+2) version of MPEG-1 standard and it is backwards compatible to MPEG-1. MPEG-2 Advanced Audio Coding (AAC) is a successor for the MPEG-2 audio. It has the same basic coding paradigm as MPEG-1 Layer III, but has improved quality at low bit rates. One of the technical improvements is the use of a standard switched modified discrete cosine transform (MDCT) filter bank rather than the hybrid filter bank. Other improvements are spectral coefficient prediction, temporal noise shaping (TNS), and bandwidth and bit rate scalable operation. MPEG-2 AAC is not backwards compatible to its predecessors. [35] [7]

MPEG-2 AAC reaches on average the same quality as MPEG-1 Layer III at about 70% of the bit rate. MPEG-2 AAC supports sample rates of 8 - 96 kHz. Target bit rate for the MPEG-2 AAC is 1 bit/sample (i.e. 96 kbit/s for a stereo signal at 48 kHz). MPEG-2 AAC supports two different media content formats. Audio Data Interchange Format (ADIF) is

[1] http://www.iis.fhg.de/amm/

designed for content storage and therefore the format has only one header in front of a file. Audio Data Transport Stream (ADTS) format is meant for streaming purposes so every frame has a header. The header has the needed information for the file or stream decoding. MPEG-2 AAC uses an Electronic Music Distribution (EMD) copyright protection system specified by SDMI. [7]

MPEG standards leave the implementation of an encoder completely open, but a decoder implementation is always done in a way that there can not be audible differences between compliant decoders. A study of different MPEG audio decoders compliance to the standard has also been made [2].

MPEG-4

MPEG-4 standard was developed in parallel to MPEG-2 and the first version of the audio standard was finished in 1999. MPEG-4 is not a compression method like MPEG-1/2, but it has a general audio coder based on MPEG-2 AAC, a family of speech coders, parametric coders for speech and music, and a scheme for representing arbitrary synthetic sounds. Also MPEG-4 has a scalable coding scheme which enables the transmission and decoding of a audio stream through varying network conditions. MPEG-4 is aimed at achieving good full-bandwidth audio quality with very low bit rates (i.e 24 kbit/s per audio channel). [24]

Ogg Vorbis

Ogg Vorbis [3] is an open source, non-proprietary, and patent free compressed audio format for high-quality audio and music. Ogg Vorbis is founded by Xiph.Org and no license fees are charged even in commercial use. Source code is also available for free.

Ogg Vorbis is divided into two parts, i.e. to the wrapper format Ogg and to the expandable audio codec Vorbis. An Ogg file can contain several audio and video streams. In the time of writing this thesis a release candidate three was available of the Vorbis audio codec and discussion about the Tarkin video codec was started.

Vorbis doesn't have subbanding like MP3, but uses MDCT and vector quantization. Sample rates of 44.1 to 48.0kHz are supported with sample resolution of 16+ bit. Vorbis is a polyphonic codec with bit rates of 30 - 190 kbit/s per channel. The content format is designed in a way that the encoded content is very easy to manipulate. One of the advantages is a bit rate peeling, which means that bit rate of a stream can be lowered on the fly without re-encoding. Sample granularity on every Vorbis packet helps in seeking and decoding an Ogg file. In the future low bit-rates (24 - 64 kbit/s) are vital for streaming. [33]

[2] http://www.mars.org/home/rob/proj/mpeg/compliance/
[3] http://www.xiph.org

4.3.1 Proprietary audio codecs

Microsoft has its own proprietary audio codec Windows Media Audio 8 [4]. According to the Microsoft a near CD-quality sound could be achieved with bit rate as low as 48 kbit/s and a CD-quality sound with bit rate 64 kbit/s with using Windows Media Audio 8 codec. RealNetworks has also its own music codec G2 [5]. For stereo music G2 supports bit rates from 16 kbit/s to 352 kbit/s. New features in Apple's QuickTime 6 [6] are MPEG-4 AAC audio and MPEG-4 video coding, but also AVI, MIDI, and MPEG-1 formats are supported.

4.4 Summary

Techniques used in perceptual audio coding were presented and a brief overview of standard codecs were given. Also a few proprietary codecs were presented. Existence of unwanted artifacts in perceptual audio coders were briefly explained and quality measurement methods were presented.

[4] http://www.microsoft.com/windows/windowsmedia/WM8/default.asp
[5] http://www.realnetworks.com/resources/howto/audio_video/audio.html
[6] http://www.apple.com/quicktime/specifications.html

Chapter 5

Architecture

In this chapter an architecture for a streaming system is presented and reasons for the solution are discussed. First, a brief review of previous and existing technology is presented to the reader. Building blocks for the streaming system are presented and how these pieces interwork with each other. Finally an Ogg Vorbis file streaming signaling flow is illustrated.

5.1 Requirements

Requirements for the system are that the system should be designed with growth and scalability. The system architecture has to provide clear and understandable interfaces for the inner and outer parts of the system. Different kinds of services could then be implemented quickly and easily without a negative impact of a clumsy architecture. E.g. a SIP user agent should have only SIP functionality and a streaming server only RTSP support, but if needed, these two could be integrated together without headache. The service applications should also be very easy to use.

All the used protocols have to be standardized by IETF. No proprietary protocols or codecs should be used. Finally, the system has to be capable of transporting and handling a real-time CD-quality audio without any perceived loss in the sound quality.

5.2 Existing streaming software

There are several server and client implementations available in the Internet for free. The commercial companies usually offer a player for free and make a profit with expensive server software. The commercial companies mostly use proprietary signaling protocols and patented codecs for the media transport and encoding.

Apple Apple's Darwin Streaming Server [1] is an open source project of Apple QuickTime Streaming Server. It has support for the industry standard Internet protocols like RTSP and RTP. It can handle both unicast or multicast streams and it is designed for native Hinted MPEG-4 streaming. MPEG-4 Hinted files are defined by Internet Streaming Media Alliance (ISMA) [2]. MP3/Icecast streaming and QuickTime is also supported. Quality of service functions include congestion control, retransmission, and server algorithms for reliable UDP streams.

GStreamer GStreamer [3] streaming media framework is based on common set of plugins for MP3 decoding, audio I/O, or anything what streams on real-time. It doesn't have ready made server or user agent software but it gives a framework where to build services like media players, video editors, streaming media broadcasters and so on. The development of the framework is still ongoing.

SHOUTcast and Icecast SHOUTcast [4] is Nullsoft's free-of-charge audio distribution system. Icecast [5] is an open source streaming server from Xiph.org. Both of these servers are based on MPEG audio technology and allows to broadcast an audio stream to as many people as their bandwidth can support. Icecast and SHOUTcast use HTTP as a stream transport protocol.

Microsoft Windows Media Player is a part of Microsoft Windows operating system distribution and therefore very popular player among Windows users. Windows Media Player has everything from digital versatile disc (DVD) video playback to real-time audio streaming. Microsoft has its own media formats for audio and video storage and transport. Microsoft Advanced Streaming Format (ASF) can be streamed with Windows Media Server. It supports multicast and unicast transport with Microsoft proprietary protocol Media Services Streaming (MSS). MSS adapts to the network conditions and lowers automatically video or audio quality if needed. Windows Digital Rights Management enables the secure distribution, promotion, and sale of digital media in the Internet. [31]

RealNetworks RealNetworks [6] introduced Internet's first widely used streaming audio format RealAudio in 1995. RealNetworks has a RealOne player and a RealSystem Server,

[1] http://www.apple.com/quicktime/products/qtss
[2] http://ism-alliance.tv/index.html
[3] http://www.gstreamer.net
[4] http://www.shoutcast.com
[5] http://www.icecast.org
[6] http://www.realnetworks.com

a streaming server family. A free version of the RealOne player with basic features can be downloaded even for the mobile phones and for the personal digital assistants (PDAs) with Symbian OS. RealOne uses RTSP and SMIL protocols for signaling and presentation. RealSystem server has over 45 media types including streaming MP3, Flash 4 and Sony ATRAC3 formats. FEC is used for the packet loss recovery. RealSystem has also an advertising, an authentication, and a stream encryption extensions.

There are also available pure audio and video players without a server functionality. The most well known players for Unix are XMMS [7] and FreeAMP [8]. E.g. FreeAMP can play MPEG-1, MPEG-2, MPEG-2,5 and Ogg Vorbis files. It can play music over Internet with HTTP unicast or RTP multicast streams.

WinAMP [9] is a very popular player for the Windows. It supports e.g. MP3 audio, SHOUTcast Radio and Windows Media formats.

SoundJam MP [10] is a free MP3 player for the Machintosh. A commercial version of SoundJam supports encoding to MP3, MP, AIFF, and WAV formats. SoundJam has streaming capabilities and can receive MP3, IceCast, SHOUTcast, FlyCast and QuickTime streams.

5.3 Streaming system

If there are so many different software on the market why to develope a new one? One of the requirements for the streaming system was that the system should use standardized IETF protocols and non-proprietary codecs. Therefore Microsoft and RealNetworks are immediately out of the question. SHOUTcast and Icecast would be good choices for the server software, but they don't support RTP. RTP is a standardized way to transport real-time media content. GStreamer is just a streaming media framework and it doesn't have signalling protocols like SIP and RTSP. Apple's Darwin streaming server would be the best choice for MediaServer implementation if it had an Ogg Vorbis support. At the moment, all the players lack the SIP functionality, which is a good property to have in the future services.

[7] http://www.xmms.org/
[8] http://www.freeamp.org
[9] http://www.winamp.com
[10] http://www.soundjam.com/

5.3.1 Architecture

An overview of the streaming system architecture is presented in figure 5.1. The UserAgent is based on a SIP phone with RTSP support. SIP provides session establishment and control, but lacks the real-time media control functionality. Thus, RTSP protocol is chosen for the signaling protocol between the MediaServer and the UserAgent. The MediaServer is located in the network and provides a storage for the Ogg files and enables RTSP file streaming. RTP is a clear choice for the real-time media transport between all parties.

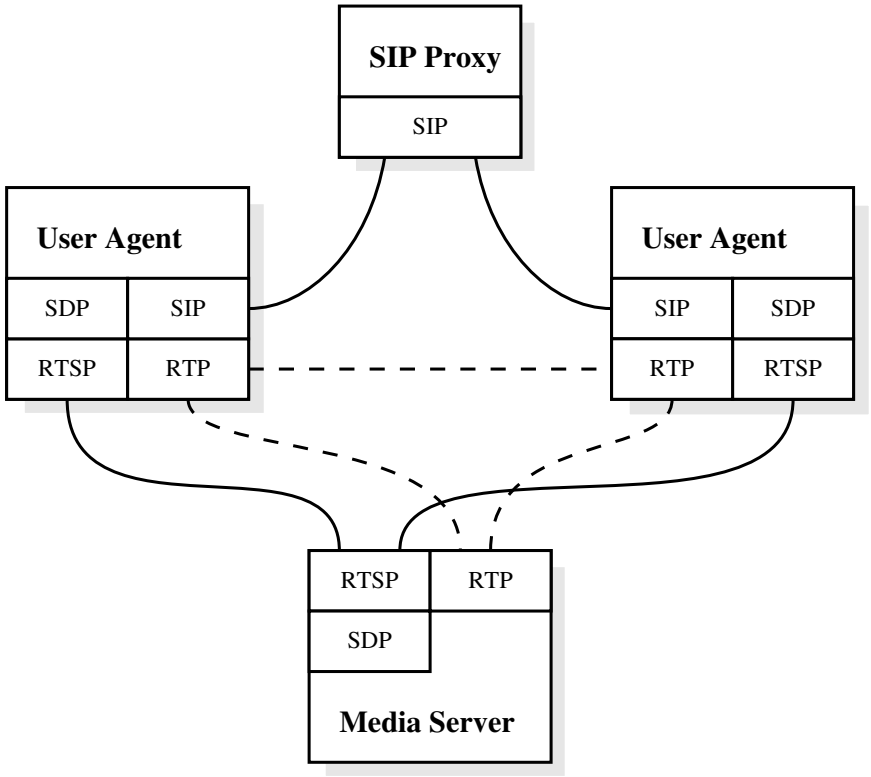

Figure 5.1: Overview of the streaming system architecture.

Figure 5.2 presents the main building blocks for the streaming system. The streaming system is divided into three parts: user applications, signaling protocols, and media subsystem.

Applications contain a UserAgent and a MediaServer. These two applications use SIP, RTSP, and SDP to establish and control streaming sessions. With SDP the codecs used, the RTP addresses and port numbers are negotiated and announced to the MediaSubSystem. The SDP is transported between the UserAgent and the MediaServer inside a SIP or RTSP payload. The session control functions e.g. session establishment and teardown are handled with SIP or RTSP. The MediaSubSystem handles the audio playback, encoding and

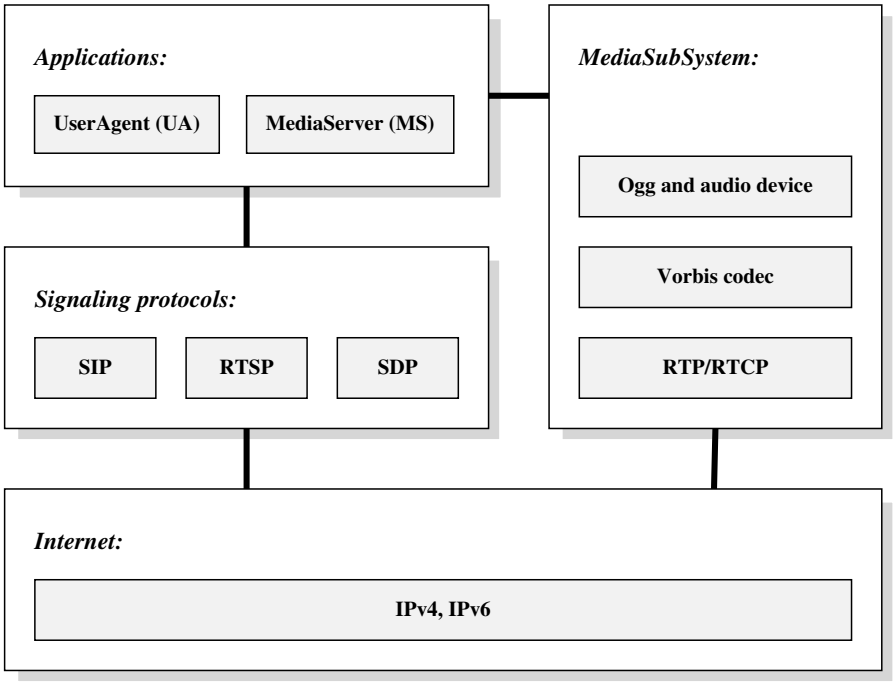

Figure 5.2: The streaming system building blocks.

decoding, and the RTP media transport over Internet.

To meet the high-quality audio requirements the Ogg Vorbis audio codec is chosen for the compressed audio media type. The main reason was that Ogg Vorbis is open source and free. Easy manipulation of encoded Ogg files was a minor reason. On the other hand, one day Ogg Vorbis might be the de facto high-quality audio standard in the Internet.

An overview of the end-to-end media transmission is presented in figure 5.3 [49]. In the MediaServer side MediaSubSystem reads an Ogg file and packs Vorbis packets to RTP packets. RTP packets are given to the network device for the transport. RTP packets are received by the UserAgent. A RTP jitter buffer is used to re-arrange the received RTP packets to the timestamp order. Vorbis packets are extracted from the RTP packets, and decoded PCM audio is given to the audio device for playback.

5.3.2 Streaming signaling flow

The objective of this thesis is the streaming of an Ogg Vorbis file. The Ogg Vorbis codebooks distribution is a problem and this section shows how to do the codebooks distribution with RTSP. An RTSP signaling flow example is presented in figure 5.4. When streaming of an Ogg Vorbis file RTSP is used for the signaling, SDP for the session desription, and RTP for the real-time content transport.

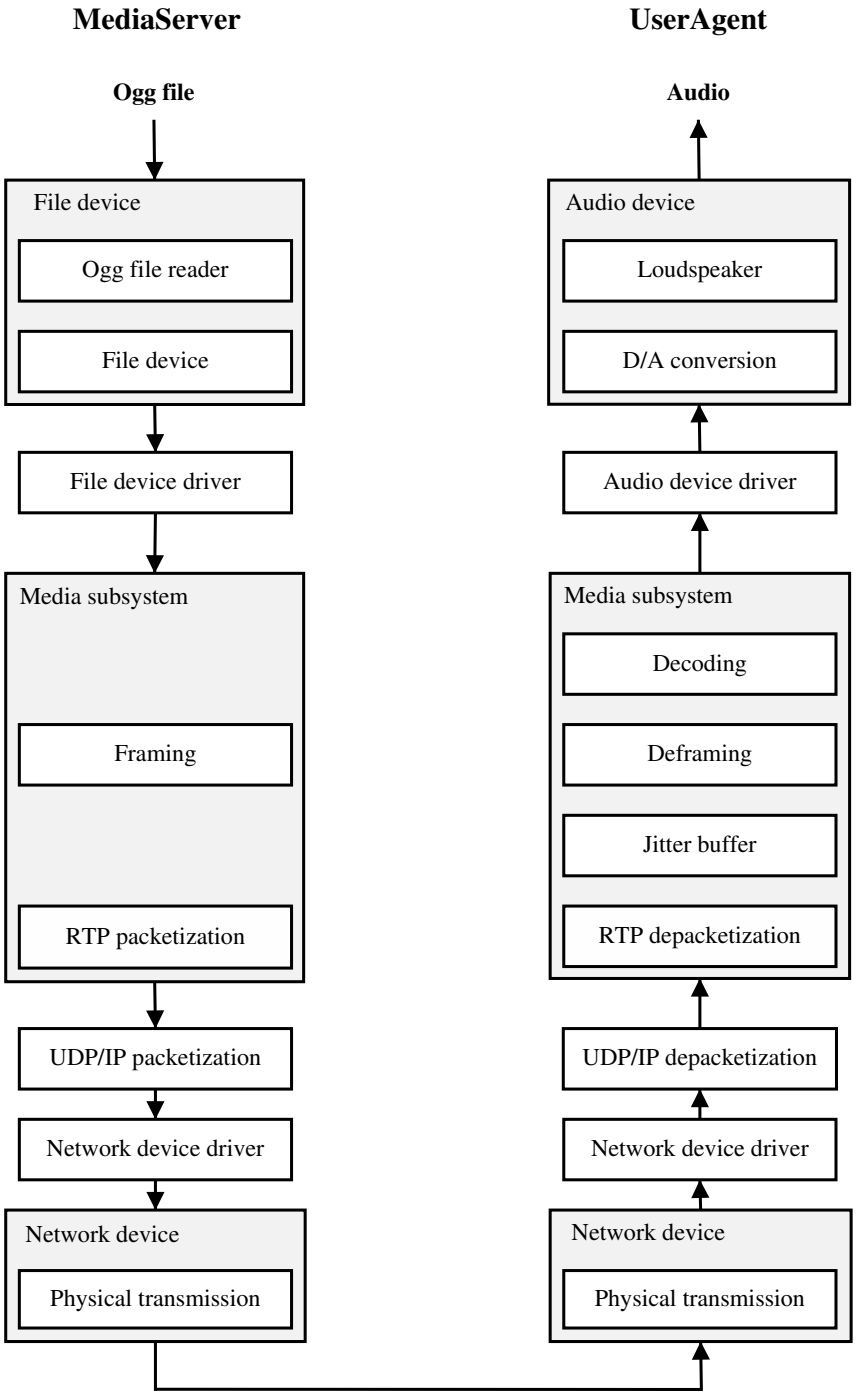

Figure 5.3: Overview of the end-to-end media processing.

The UserAgent contacts to the MediaServer and queries the available tracks of *Bob James* album *Joy Ride* [27].

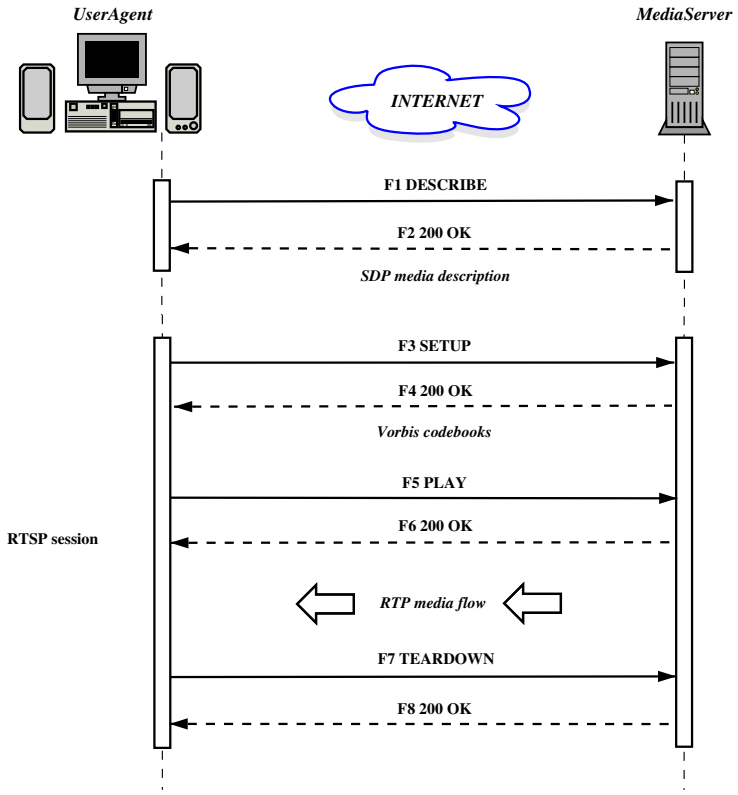

Figure 5.4: RTSP file streaming.

```
F1 DESCRIBE UserAgent -> MediaServer

DESCRIBE rtsp://music.nokia.com/bob_james/joy_ride RTSP/1.0
CSeq: 1
```

The MediaServer has the album in its collection so the MediaServer can send a 200 OK response. Available tracks are listed in a session description, which is attached to the payload of the response.

```
F2 200 OK MediaServer -> UserAgent

RTSP/1.0 200 OK
CSeq: 1
Content-Type: application/sdp
Content-Length: 164

v=0
o=media_server 2890844256 2890842807 IN IP6 fec0::202:b3ff:fe14:1073
s=Bob James
i=Joy Ride
a=recvonly
t=0 0
m=audio 0 RTP/AVP 108
a=rtpmap:108 VORBIS/44100
a=rtpmap:108 channels=2
a=rtpmap:108 quality=10.00
a=control:rtsp://music.nokia.com/bob_james/joy_ride/take_me_there.ogg
a=control:rtsp://music.nokia.com/bob_james/joy_ride/raise_the_roof.ogg
a=control:rtsp://music.nokia.com/bob_james/joy_ride/it's_all_right.ogg
a=control:rtsp://music.nokia.com/bob_james/joy_ride/swingset.ogg
a=control:rtsp://music.nokia.com/bob_james/joy_ride/joy_ride.ogg
a=control:rtsp://music.nokia.com/bob_james/joy_ride/what's_up.ogg
a=control:rtsp://music.nokia.com/bob_james/joy_ride/fly_by.ogg
a=control:rtsp://music.nokia.com/bob_james/joy_ride/the_first_time.ogg
a=control:rtsp://music.nokia.com/bob_james/joy_ride/strollin'.ogg
a=control:rtsp://music.nokia.com/bob_james/joy_ride/sweet_talk_me_now.ogg
a=control:rtsp://music.nokia.com/bob_james/joy_ride/trade_winds.ogg
a=control:rtsp://music.nokia.com/bob_james/joy_ride/bisso_baba.ogg
```

The UserAgent wants to play the first track *Take Me There*. The UA picks up the first RTSP URL (URL pointing to the *take_me_there.ogg* file) from the session description and sends a RTSP SETUP message to this URL. In the transport header UserAgent announces the RTP port numbers to the server.

```
F3 SETUP UserAgent -> MediaServer

SETUP rtsp://music.nokia.com/bob_james/joy_ride/take_me_there.ogg RTSP/1.0
CSeq: 2
Transport: RTP/AVP,unicast;client_port=8000-8001
```

The MediaServer receives the RTSP SETUP request and initializes the session for streaming. The MediaServer starts reading the *take_me_there.ogg* file. Before the actual Vorbis data there are three header packets in a front of a Vorbis stream. The first packet is a main header, which is very short and contains general information about the stream, e.g. number of channels, sample rate, and average bit rate. The second packet is a comment header. The comment header [16] contains information like title, version, album, track number, and artist. The third packet in the Vorbis stream contains the essential codebooks. Each of the first three packets, the main header, the comment header, and the codebooks are *base64* encoded and attached to the payload of the 200 OK response as a MIME multiparty format.

```
F4 200 OK MediaServer -> UserAgent

RTSP/1.0 200 OK
CSeq: 2
Transport: RTP/AVP;unicast;client_port=8000-8001
Session: 12345678
Content-Type: multipart/mixed; boundary="vorbis"
Content-Length: 3164

--vorbis
Content-Type: audio/vorbis
Content-transfer-encoding: base64

... Vorbis main header ...

--vorbis
Content-Type: audio/vorbis
Content-transfer-encoding: base64

... Vorbis comment header ...

--vorbis
Content-Type: audio/vorbis
Content-transfer-encoding: base64

... Vorbis Codebooks ...

--vorbis--
```

The UserAgent receives the first three headers of the stream and can now initialize the Vorbis decoder. When the Vorbis decoder initialization has succeeded the UserAgent can send a RTSP PLAY request to the MediaServer and the streaming can begin.

```
F5 PLAY UserAgent -> MediaServer

PLAY rtsp://music.nokia.com/bob_james/joy_ride/take_me_there.ogg RTSP/1.0
CSeq: 3
Range: npt=0-
Session: 12345678
```

An uni-directional RTP connection is established and the rest of the Vorbis packets in the Ogg file are transported through RTP.

```
F6 200 OK MediaServer -> UserAgent

RTSP/1.0 200 OK
CSeq: 3
Session: 12345678
```

After listening the stream the UserAgent hangs up the stream by sending RTSP TEARDOWN request to the MediaServer.

```
F7 TEARDOWN UserAgent -> MediaServer

TEARDOWN rtsp://music.nokia.com/bob_james/joy_ride/take_me_there.ogg RTSP/1.0
CSeq: 4
Session: 12345678
```

The MediaServer responds and the RTSP streaming session ends.

```
F8 200 OK MediaServer -> UserAgent

RTSP/1.0 200 OK
CSeq: 4
```

5.4 Summary

Requirements for the streaming system were defined and existing streaming applications were presented. The streaming system includes a MediaServer and a UserAgent. The UserAgent has support for SIP, SDP, RTSP and RTP protocols, but the MediaServer lacks the SIP functionality. A RTSP signaling flow between the UserAgent and the MediaServer in an Ogg Vorbis file streaming is also descibed. In the Ogg Vorbis file streaming RTSP is used for the signaling and for the Ogg Vorbis codebooks distribution, SDP is used for the session description, and RTP is used for the real-time media content transport over Internet.

Chapter 6

Implementation

This chapter discusses the implementation issues of the streaming system described in chapter 5. The UserAgent and the MediaServer are implemented and UML diagrams presented. RTSP and SDP implementation is also discussed and an *RTP payload format for Vorbis encoded audio* [32] is implemented to the MediaSubSystem. In addition a simulator for measuring properties of the Vorbis stream was also made.

6.1 Overview of the implementation

Implementation was done in small pieces while writing this thesis. Implementation was done in Nokia Networks as a part of a project studying SIP services in 2002. The system required a lot of straightforward coding and one of the challenges was how to integrate previous code and existing modules to the new system. A ready made SIP parser was used in a RTSP implementation. The real-time audio transport system (MediaSubSystem) was taken from the previous work done by Jari Selin [49]. However, a lot of modifications were made and new components were implemented.

6.2 Tools

The choice for the programming language was clear. The protocols used were implemented in C++, so C++ was a clear choice for SDP, RTSP, UA and MS implementation. C++ is not as efficient as C because of the object-oriented nature, but it has powerful tools for the implementation of clear and modular interfaces. The MediaSubSystem was implemented in C so the Ogg Vorbis codec components had to be written in C. The whole implementation work was done on a Linux RedHat 7.2 platform and measurements were made with Pentium III processor running at 500MHz.

6.3 Applications: UserAgent and MediaServer

The UserAgent (UA) and the MediaServer (MS) are the main applications of the streaming system presented in chapter 5. While running the UserAgent or the MediaServer several procedures runs in parallel. Underlying protocols are waiting for the incoming requests and the user might also activate something from the console. If the same resource is used by two processes at the same time something unpredictable can happen. For example the value of a variable cannot be trusted if two processes have altered it at the same time. If pointers are used usually concurrency problems occur in a form of a segmentation fault, which halts the system. Therefore some kind of locking mechanism is needed, e.g. semaphores or locks, but the concurrency problems can be avoided by running the application with one thread. A structure for this kind of successive running procedure is presented in figure 6.1.

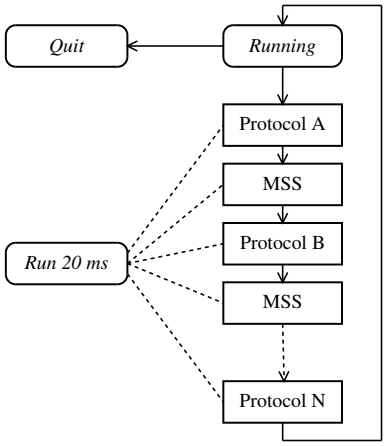

Figure 6.1: Scheduler for running UserAgent and MediaServer.

Every registered protocol is been run only $20ms$ in consecutive order. New protocols can be afterwards added easily by registering them to the scheduling engine. Problems in this kind of structure occur when some protocols need to be processed more often than others. Therefore MediaSubSystem (MSS) is run after every registered protocol, to allow enough time to read or write data to the hardware audio buffers.

The UserAgent extends to two abstract classes: RtspUA and SipUA. These two classes implement the SIP and RTSP protocols, which is discussed in section 6.4. UserAgent (UA) type classes means that in order to support the wanted functionality in the main program these methods have to be overloaded with proper implementations. UserInterface (UI) type classes define all the methods that can be called from the main program. Unified Modeling Language (UML) [17] class hierarchy diagrams of the MediaServer and the UserAgent are presented in figures 6.3 and 6.2. Version 1.2 of the OMG (Object Management Group)

UML standard is used.

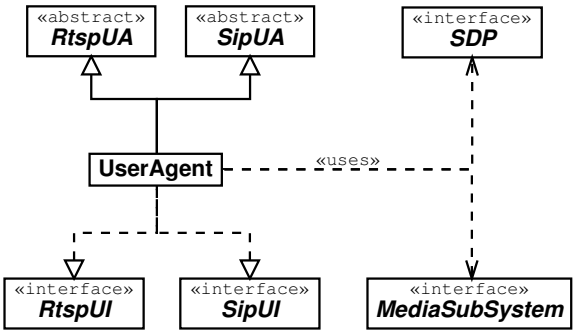

Figure 6.2: UserAgent UML class hierarchy.

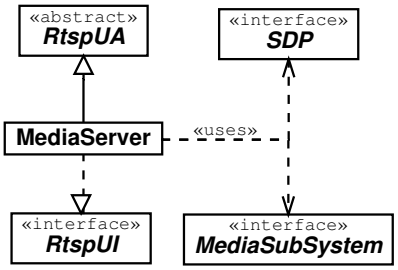

Figure 6.3: MediaServer UML class hierarchy.

The UserAgent and the MediaServer use SDP for session description and negotiation. MediaSubSystem is used in both applications for the real-time RTP media transport without RTCP functionality. The MediaServer has all the same functionality as UserAgent except the SIP protocol.

6.4 Signaling protocols: SIP, RTSP, and SDP

SIP

The SIP protocol implementation was done in a previous project in Nokia. However, the project didn't produce suitable SIP interfaces for the user agent creation so the SipUA and the SipUI classes are implemented on top of the ready made SIP stack and parser. UML diagrams of the SipUA and SipUI classes are shown in figures 6.4 and 6.5.

RTSP

The RTSP protocol is very similar to SIP so the RTSP implementation was done using the SIP stack. Only the SIP protocol identifier and the SIP URL are changed to mach ones

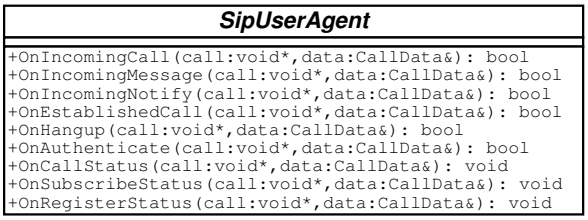

Figure 6.4: SIP UserAgent.

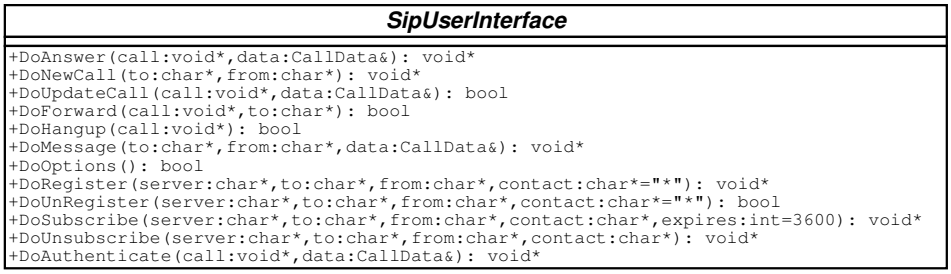

Figure 6.5: SIP UserInterface.

defined in RTSP specification. Also a couple of new headers are attached to the RTSP implementation including public, range, rtp-info, and transport headers. Figures 6.6 and 6.7 presents the UML diagrams of the RtspUA and RtspUI classes.

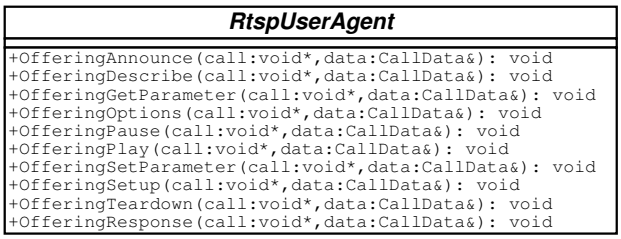

Figure 6.6: RTSP UserAgent.

SDP

The SDP implementation is divided into seven classes as shown in figure 6.8. SessionDecription is the main class which instantiates and uses the rest of the classes. Every class has Parse and ToString methods for parsing and creating incoming and outgoing SDP messages.

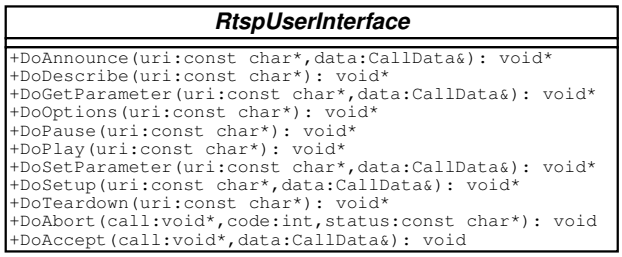

Figure 6.7: RTSP UserInterface.

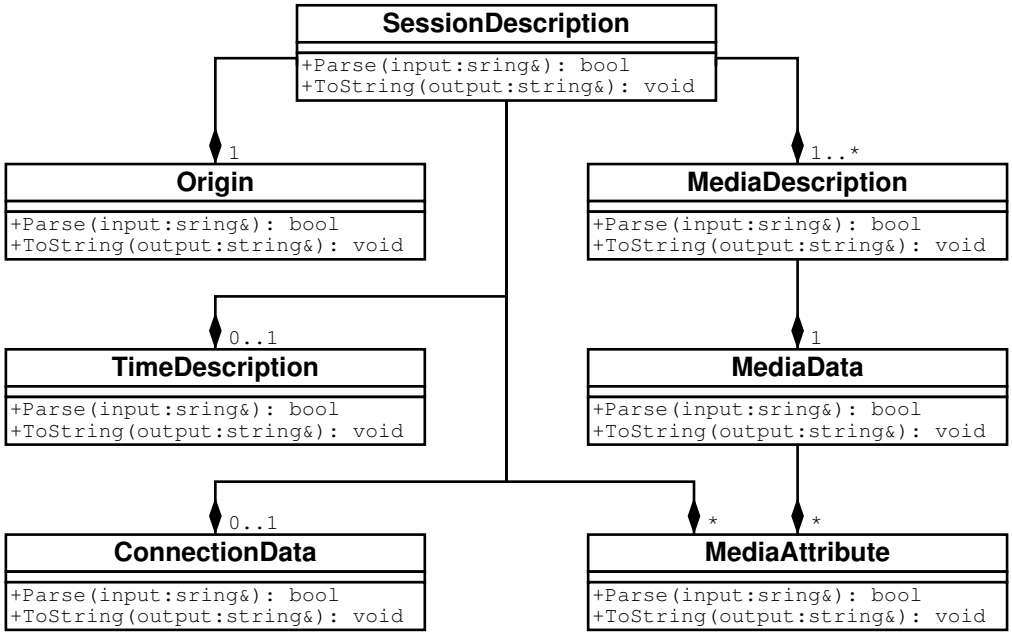

Figure 6.8: SDP UML diagram.

6.5 MediaSubSystem

The MediaSubSystem is controlled through RTSP like interface with help of SDP. The MediaSubSystem (see figure 5.2) is divided into three modules, to codecs, to audio devices, and to the RTP implementation. The codec module contains all the available codecs and is responsible of audio encoding, decoding, framing and packing RTP packets. The RTP module is responsible of the media transport. Different types of audio devices can be attached to the MediaSubSystem through the device interface. For this thesis a device for the audio playback and a device for the reading of Ogg files was needed.

6.5.1 Audio device

The audio device has several parameters which can be altered. A $buffer\ size$ determines the size of the PCM buffer in the audio device. A $periodtime$ sets the time interval at which hardware will interrupt the host system to notify it that space/data is available. Usually the $period\ size$ is same as a codec frame size and the $buffer\ size$ is n times $period\ size$, where n is the number of periods. Latency of the audio drivers is controlled by n. For low latency applications n is small, but for high-quality streaming applications, where latency is not so crucial n can be quite large.

6.5.2 Ogg file device

Music is usually stored on a hard drive in an encoded format. Music encoded with Vorbis codec is stored in an Ogg file and in order to stream these files an Ogg file device has to be implemented for the MediaSubSystem.

An Ogg file consists of Ogg pages. An Ogg page contains one or more Vorbis packets, and the Vorbis packets variable size in length contain encoded audio. A Vorbis packet can span over several Ogg pages. Every Vorbis packet has a field called $granulepos$, which determines the packet's exact location in the stream in samples. A $granulepos$ with value -1 means that the packet is not the last Vorbis packet in this Ogg page. Ogg page number 5 in a figure 6.9 is a normal situation. Every Vorbis packet in a page has a $granulepos$ value -1 except the last one, which has a proper value 3006.

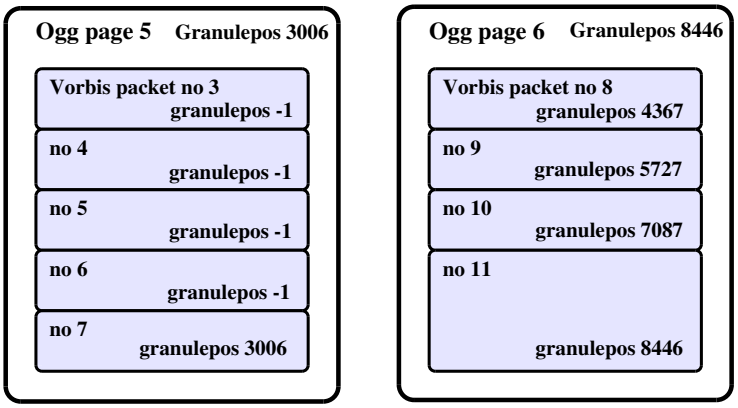

Figure 6.9: A normal and a modifed Ogg page

In the Ogg Vorbis streaming the Ogg layer is ripped off and only the Vorbis packets are sent through the network. If RTP is used as a transport protocol the Vorbis packet $granulepos$ value is used as a RTP packet timestamp value. Therefore, a Vorbis packet $granulepos$ value -1 has to be changed. In figure 6.9 the Ogg page number 6 has a modified

Vorbis packet $granulepos$ values.

The modified $granulepos$ value in the Vorbis packet is derived from equation 6.2.

$$duration = \frac{granulepos_{prev.page} - granulepos_{page}}{n} \tag{6.1}$$

$$granulepos_i = i \cdot duration + granulepos_{prev.page}, \tag{6.2}$$

where $granulepos_i$ is the next odd number, $granulepos_{page}$ is the granulepos of the Ogg page, $granulepos_{prev.page}$ is the granulepos of the previous Ogg page, n is the number of Vorbis packets in this Ogg page, and $i = 1...n-1$ is the i:th Vorbis packet on page.

In this way an odd granulepos value means in the receiver side that the exact Vorbis packet granulepos is unknown and should be -1. On the other hand, with an even granulepos value an exact sample position is known.

6.5.3 RTP payload format for Vorbis encoded audio

Before the actual Ogg Vorbis streaming could be established the *RTP payload for Vorbis encoded audio* [32] has to be implemented. New modifications and enhancements to the draft are presented and analyzed in the next chapter.

6.5.4 Packing Vorbis packets to RTP packet

In every RTP packet there is a Vorbis header (see figure 6.10), which takes the first byte of the RTP payload data.

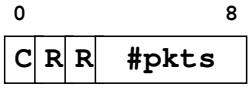

Figure 6.10: Vorbis header is present in every RTP packet.

Continuous (C) Set to one if Vorbis packet is fragmented over several RTP packets.

Reserved (R) Reserved for the future use. Must be zero.

Number of packets (pkts) Number of Vorbis packets in one RTP packet. If the $continuous$ bit is set, this field should be zero.

Constructing RTP packets from Vorbis packets is described in figure 6.11. If the Vorbis packet size is larger than 256 bytes it is placed in the RTP packet itself. If the Vorbis packet size is larger than the path maximum transmission unit (MTU) then the Vorbis packet is fragmented into several RTP packets. Every fragmented Vorbis packet has a value of zero

in the $pkts$ field. A $continuous$ bit is set to one after the first fragment. The last fragment has its RTP $marker$ bit set to one.

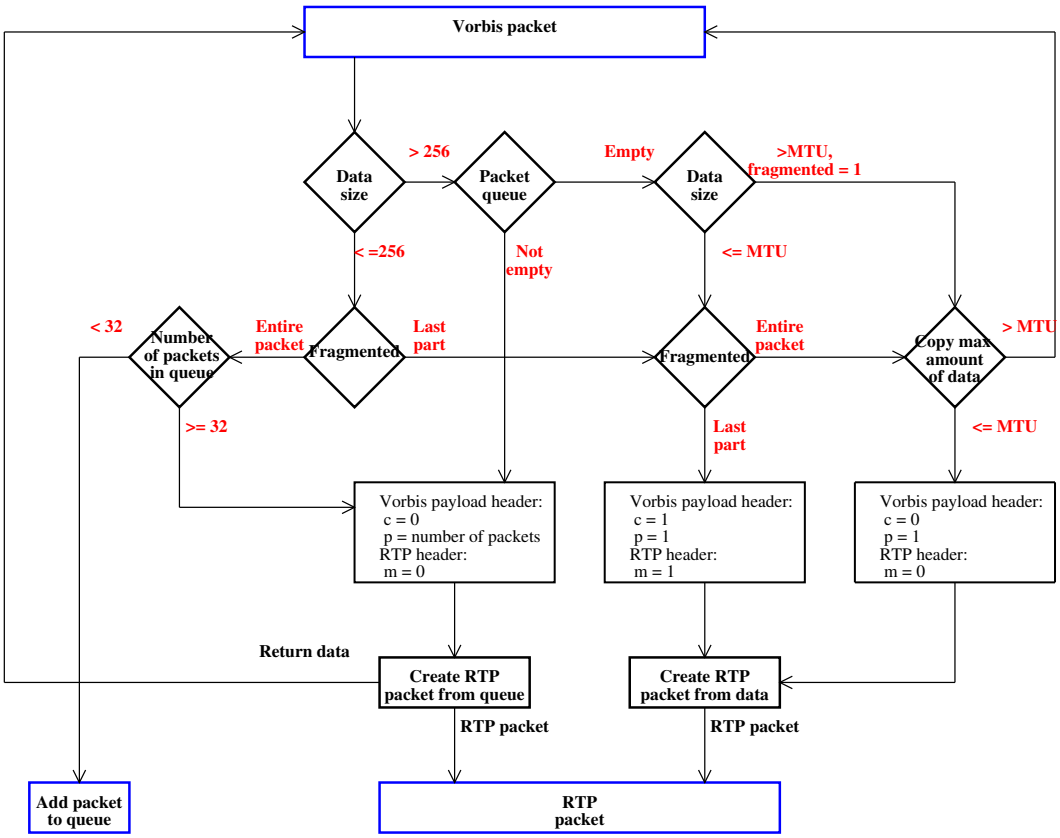

Figure 6.11: A flowchart for constructing RTP packets from Vorbis packets.

Consecutive packets smaller than 256 bytes should be bundled into the same RTP packet. A maximum number of Vorbis packets in one RTP packet is limited to 32. When multiple Vorbis packets are bundled into one RTP packet, a byte is added to represent the length of the data before the actual payload. Figure 6.12 shows an example of a RTP payload data with two Vorbis packets inside.

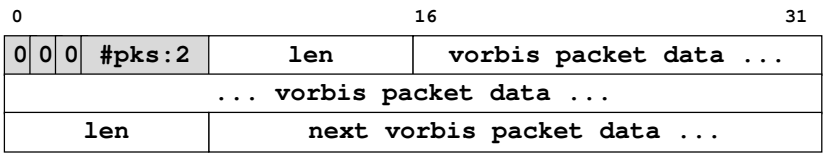

Figure 6.12: Example of the RTP packet.

6.5.5 Extracting Vorbis packets from RTP packet

On the receiver side Vorbis packets are extracted from the RTP packets. The Vorbis header in every RTP packet has the needed information so that all the Vorbis packets can be extracted even if there are multiple Vorbis packets in one RTP packet, or a Vorbis packet is fragmented over several RTP packets.

A decoder keeps the sum of the decoded samples. If a RTP packet is lost during the transport the decoder knows from the Vorbis packet $granulepos$ value that how many samples of silence must be generated.

6.5.6 Mapping to SDP Parameters

SDP is used for describing the session parameters. With Vorbis the sampling rate, the number of channels, and the quality or bit rate could be specified as an optional fmtmap parameters. Table 6.1 presents the SDP fmtmap attributes and the range of the possible values. An example of the Vorbis attributes usage in SDP is illustrated in figure 6.13.

Table 6.1: Vorbis SDP parameters

Attribute	Range
channels	1+
max bit rate	16 - 390
nominal bit rate	16 - 390
min bit rate	16 - 390
quality	0.00 - 10.00

```
v=0
o=media_server 2890844256 2890842807 IN IP6 fec0::202:b3ff:fe14:1073
s=Bob James - Joy ride
i=Take Me There
t=0 0
m=audio 0 RTP/AVP 108
a=rtpmap:108 VORBIS/44100
a=fmtmap:108 channels=2
a=fmtmap:108 quality=10.00
a=control:rtsp://music.nokia.com/bob_james/joy_ride/take_me_there.ogg
```

Figure 6.13: Mapping to sdp parameters

6.6 Ogg Vorbis stream measurements

Nokia has a software for testing MediaSubSystem codecs. The software was modified so that the Ogg Vorbis stream properties could be measured. This simulator had a graphical

user interface for controlling stream parameters, e.g. stream quality, packet drop, and estimated network bandwidth and delay. RTP packets were generated but never sent to the network. There is a first-in, first-out (FIFO) buffer, where the sender added and the receiver removed RTP packets in a timestamp order. The simulator enables also to log Vorbis packet sizes, RTP packet sizes, timestamps, and RTP packet processing, encoding, and decoding delays to an external file. The measured data is analyzed in the next chapter.

6.7 Summary

The streaming system architecture was discussed in an implementation specific level. UML class diagrams of the MediaServer and of the UserAgent were presented. Also the protocols used in session description, establishement, and control was implemented. To enable Ogg Vorbis audio streaming the *RTP payload for Vorbis encoded audio* was implemented to the MediaSubSystem. Also measurements of the Ogg Vorbis audio stream were made.

Chapter 7

Analysis

In this chapter the solution is evaluated by referring to the previously defined criteria. Different methods for the Vorbis codebooks packet distribution are compared to the one given in solution. The quality of the Vorbis packets packaging to the RTP packets are measured and overhead transmission simulations are made. Also a network congestion situation is briefly presented.

7.1 Criteria for analysis

Criteria for the evaluation of the solution were defined previously in chapter 2.

7.1.1 Architecture: Modularity

The modularity of the system was achieved by dividing all the used protocols to own modules. Signaling protocol modules are attached straight to the application without a common interface. This kind of structure was very easy to implement. A negative aspect of this kind of solution is the amount of different protocol interfaces between the application and the signaling protocols. Therefore, RTSP and SIP protocols had separate interfaces for the application, which made things complex.

UserAgent and MediaServer have been implemented in C++. Thus multiple inheritance was allowed, which made the addition of new protocols to the existing service possible. New services and applications could be implemented quickly, because only the needed methods have to be overloaded with proper implementations.

7.1.2 Implementation: Open interfaces

The streaming system was implemented using IETF protocols. The signaling was provided with RTSP and the real-time media transport with RTP. The high-quality audio codec Ogg Vorbis was also an open source and free. No proprietary protocols or codecs were used. Thus the streaming system was made fully with open standards. Interworking with other streaming systems using the same standards should be possible.

7.1.3 Quality: CD-quality audio

The last evaluation criterium was that the system should be capable of transporting CD-quality audio through the Internet. Buffers are important factors in audio transport, because they enable data storage and processing in advance. The CD-quality Ogg Vorbis audio codec was chosen for the encoded audio content format. The Vorbis RTP payload format defines how Vorbis encoded audio is transported through the Internet.

Breaks in music destroy the feeling of continuation. Breaks will occur if there is not enough processing time to play the scheduled frames in time or the frame buffer is empty. In some extent breaks in music can be eliminated with the use of buffering.

Buffering

Buffering is an essential part of real-time audio delivery. There are buffers in both sending and receiving ends. Buffer sizes depend on the used codec, sampling rate, and the amount of latency. In low latency interactive applications buffer sizes are small but in applications used in streaming the size of the receiving end buffer could be measured in seconds.

In RTP audio transport over UDP, RTP packets arrive in a random order. Receiver end buffers are needed to equalize the stream. An RTP buffer is used to organize the received RTP packets to the timestamp order. This is the order where RTP packets are also decoded. The perceptual audio decoder produces different amount of PCM samples from every encoded packet. Sometimes it doesn't produce anything and sometimes it produces a bunch of PCM samples so an additional PCM buffer is also needed. An example of the RTP and PCM buffer variations are presented in figure 7.1.

This figure reveals that the RTP buffer variation is quite small but the PCM buffer fluctuates quite heavily. PCM buffer variation is due to the properties of the Ogg Vorbis decoder and how it produces PCM samples. If low latency services are needed the buffer variation minimization is necessary. However, in streaming the latency is not so critical, but the variation of buffers have to be taken into account when specifying the buffer sizes.

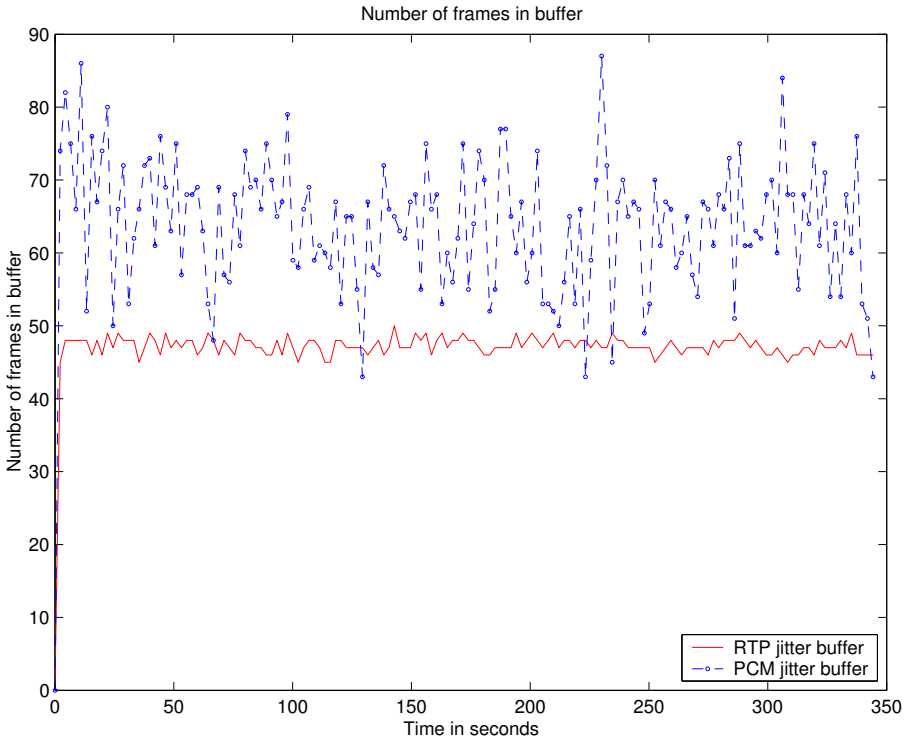

Figure 7.1: RTP jitter and PCM buffer size variation over time with Ogg Vorbis stream encoded with quality 10.

RTP payload format

The RTP packet size is one of the most important factors, when specifying a RTP payload format. Because the additional protocol headers (20/32 bytes for the IPv4/IPv6 header, 8 bytes for the UDP header, and 12 bytes for the RTP header) take total of 40/52 bytes the size of the data to be packed has to be known so that the RTP packets wont get fragmented. On avarage RTP packet sizes should be quite small, because RTP packets can get lost. Of course, the amount of lost content is proportional to the lost RTP packet size. *RTP payload format for Vorbis encoded audio* defines how the Vorbis packets are transported through the Internet.

7.2 RTP payload format for Vorbis encoded audio

The implementation of the *RTP payload format for Vorbis encoded audio* [32] is now analyzed. In the time of writing this thesis a Release Candidate 3 (1.0rc3) was available of the Ogg Vorbis codec.

7.2.1 Vorbis codebooks distribution

This section evaluates different methods for the Vorbis codebooks distribution and tries to find a best solution for the unicast streaming.

HTTP HTTP is ment for reliable data transport. The Vorbis codebooks packet can be *base64* encoded and attached to the payload of the HTTP response. The server creates a HTTP link pointing to the Vorbis codebooks packet. The HTTP link could be carried inside SDP. However, server and client implementations will become more complex because of the additional HTTP protocol. The HTTP server is also responsible of creating and maintaining the Vorbis codebooks links.

SDP Distributing the Vorbis codebooks packet using SDP is not recommended, because SDP can be transported with an unreliable protocol. Therefore error detection and correction algorithms are needed. And first of all, all the information attached to the SDP should be understood by the end users. However, SDP could contain a HTTP link to the Vorbis codebooks packet as discussed in a previous paragraph.

SIP In the future SIP will be widely used in multimedia session establisment. With SIP Ogg Vorbis audio streams could be started and ended easily, but the real-time session control would be missing. Real-time media control could be done with SIP, but the implementation would then be quite tricky. SIP is a versatile protocol but everything shouldn't be handled with it. With SIP Ogg Vorbis streams could be advertised and established, but the final session control should be done with RTSP.

RTSP RTSP is designed to establish and control real-time sessions. The actual data should be transported with some other protocol, e.g RTP. For the essential binary data the RTSP specification proposes a method called binary interleaving. In a binary interleaving the important RTP packets are sent through the persistent RTSP TCP connection and not within RTP. The property was specified for tunneling UDP traffic through firewalls in a TCP connection. However, this should be avoided since it complicates client and server operation.

An easier way to do the distribution is to treat the Vorbis codebooks packet as signaling information and send the packet within a RTSP response. This technique provides an easy and clear way to handle Vorbis codebooks packet distribution. Client and server implementations are also explicit and don't contain any tricky hacks. However, both of these RTSP distribution techniques are valid only if RTSP is carried over a TCP connection.

The signaling flow example in figure 5.4 shows how the Ogg Vorbis codebooks distribution can be handled in a single physical unicast Vorbis stream. However, a physical Vorbis stream can contain several logical Vorbis streams. Each logical Vorbis stream can be encoded with different sets of codebooks. The logical Vorbis stream codebooks can also be distributed in a middle of a physical stream within RTSP. RTSP has a property that the server can also establish requests to the client. In a beginnig of a logical Vorbis stream the server can send all the headers to the client with a RTSP SET_PARAMETER message. However, this is useful only in a unicast environment.

RTP/RTCP RTP and RTCP are ment for real-time media transport. RTCP is an optional protocol which can be implemented if needed. Thus Vorbis codebooks packet could be interleaved to the RTP media stream and be sent in small blocks inside the actual stream. This produces an additional overhead, because the same blocks are sent multiple times during the stream. Interleaving the Vorbis codebooks packet to the RTP media stream could be a worth while to study further for multicast streaming. It has to be kept in mind that packet delivery can't be guaranteed so error detection and protection algorithms have to be used.

SAP Multicast streams can be advertised with SAP, but it doesn't offer a good solution for the distribution of the Vorbis codebooks packet. SAP is carried over UDP so it has the same disadvantages as RTP. SAP announcements are done periodically in a maximum rate of 4000 bits per seconds so it is ment for quite small packets, e.g. SAP announcements could contain only a SDP.

As a conclusion it can be noticed that the suggested RTSP method for the distribution of the Vorbis codebooks packet is the best choice for the streaming purposes.

7.2.2 Vorbis packet sizes

Currently Vorbis packets are unbound in length, but maybe in the future there is a practical limit for the packet length. Typical Vorbis packet sizes are from very small (20 bytes) to quite large (8-12 kilobytes). Vorbis can encode audio in different quality levels ranging from 0.00 (the worst) to 10.00 (the best). Figure 7.2 shows the Vorbis packet size distribution over time with Ogg Vorbis example stream encoded in eleven different quality levels.

It shows that the Vorbis packet size depends on the encoding quality and in a Vorbis stream there are always "big" and "small" Vorbis packets. The figure also proves the broad range of Vorbis packet sizes. Statistical data of the Vorbis packets are listed in table 7.1, where also average bit rates are calculated. It can be noticed that the amount of the Vorbis

packets are almost the same, but the size of the packets alternates. With quality levels 0.00 and 1.00 almost all the Vorbis packets, and with quality 2.00 half of the Vorbis packets are shorter than 256 bytes. With quality levels 3.00 or better the number of small Vorbis packets is constant. Avarage stream bit rates and size of the Vorbis packets with only the worst and the best encoding quality levels are presented in more detail in section B.1.

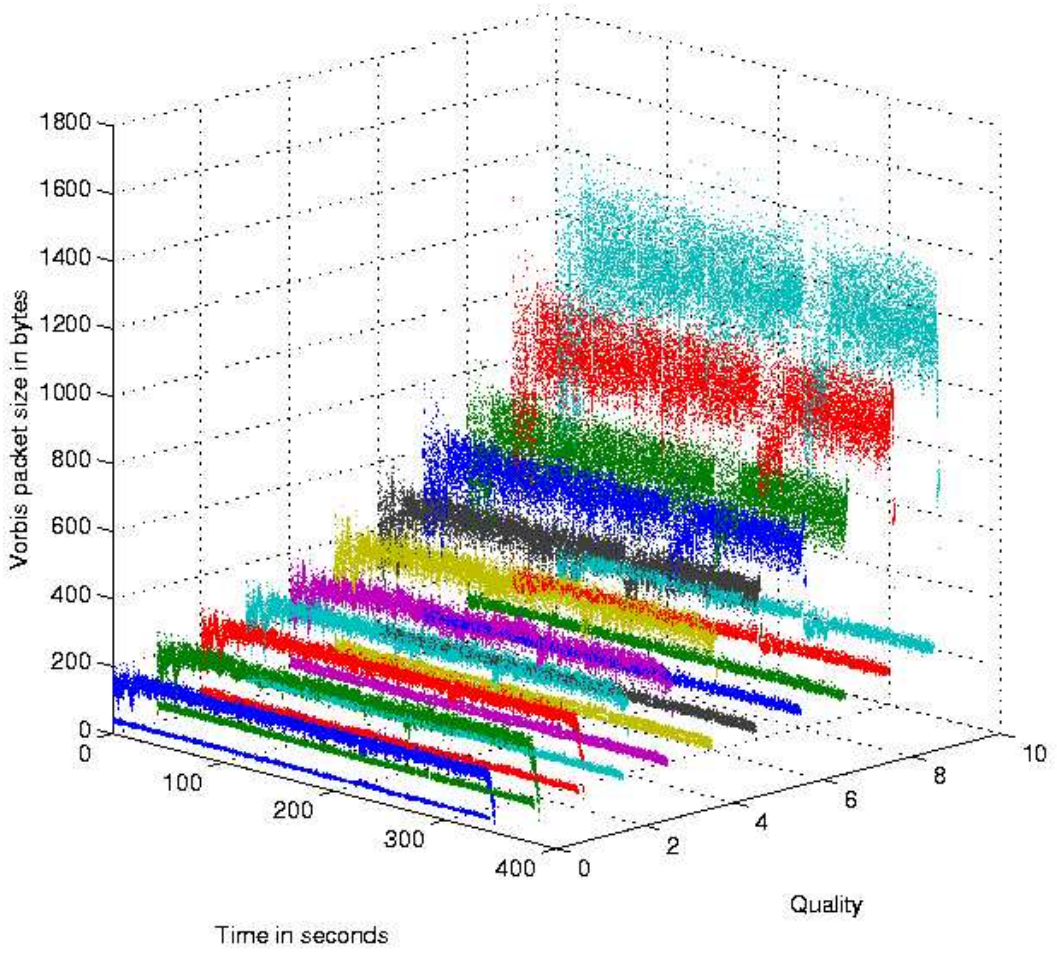

Figure 7.2: Vorbis packet size distribution of the example stream with different quality levels.

7.2.3 RTP packet sizes

Now it is time to analyze how the implementation packs the Vorbis packets into the RTP packets. Figure 7.3 shows the RTP packet size distributions of the previous Vorbis streams. Almost all the small Vorbis packets (under 256 bytes) have disappeared and been bundeled into the bigger RTP packets. Statistical data from the RTP packets are gathered to table 7.2.

Table 7.1: Vorbis packet size statistics of the example stream, where n is the number of Vorbis packets. Vorbis packet sizes are in bytes.

Quality	Bit rate	n	< 256 bytes	Min size	Max size	Average size
0.00	60,5	19632	19632	20	228	131
1.00	79,4	21621	21547	1	272	156
2.00	94,5	21621	12171	28	323	186
3.00	109	21810	8490	50	365	212
4.00	119	21810	8300	55	429	232
5.00	149,1	21810	7995	68	528	291
6.00	178,8	21810	7965	81	634	349
7.00	214,0	21810	7956	93	799	418
8.00	232,4	21719	7956	102	925	454
9.00	310,9	21810	7956	115	1282	607
10.00	390,4	21810	7953	122	1625	763

Figure 7.3 also reveals that with quality levels 2.00 or better the implementation works quite nicely. However, with quality levels 0.00 and 1.00 RTP packets are spread from few bytes to almost up to 2 kbytes in size. In the real network streaming situation the maximum size of the RTP packet would be limited to the size of path MTU. However, in this simulation the path MTU was not taken into acount. A reason for the this kind of dispersion in RTP packet sizes is that all the Vorbis packets are under 256 bytes in length. Thus up to 32 Vorbis packets can be bundled into one RTP packet. RTP packet sizes with only the worst and the best encoding quality levels are presented in more detail in section B.1.

Table 7.2: RTP packet statistics of the example stream, where n is the number of RTP packets. RTP packet sizes are in bytes.

Quality	Bit rate	n	Min size	Max size	Average size
0.00	60,5	2453	225	1605	1056
1.00	79,4	2786	56	1987	1220
2.00	94,5	12304	55	2027	328
3.00	109	15000	61	1977	310
4.00	119	15140	71	1975	335
5.00	149,1	15390	72	1473	414
6.00	178,8	15412	91	1000	495
7.00	214,0	15419	103	1020	592
8.00	232,4	15328	108	1118	643
9.00	310,9	15419	122	1408	861
10.00	390,4	15422	128	1788	1081

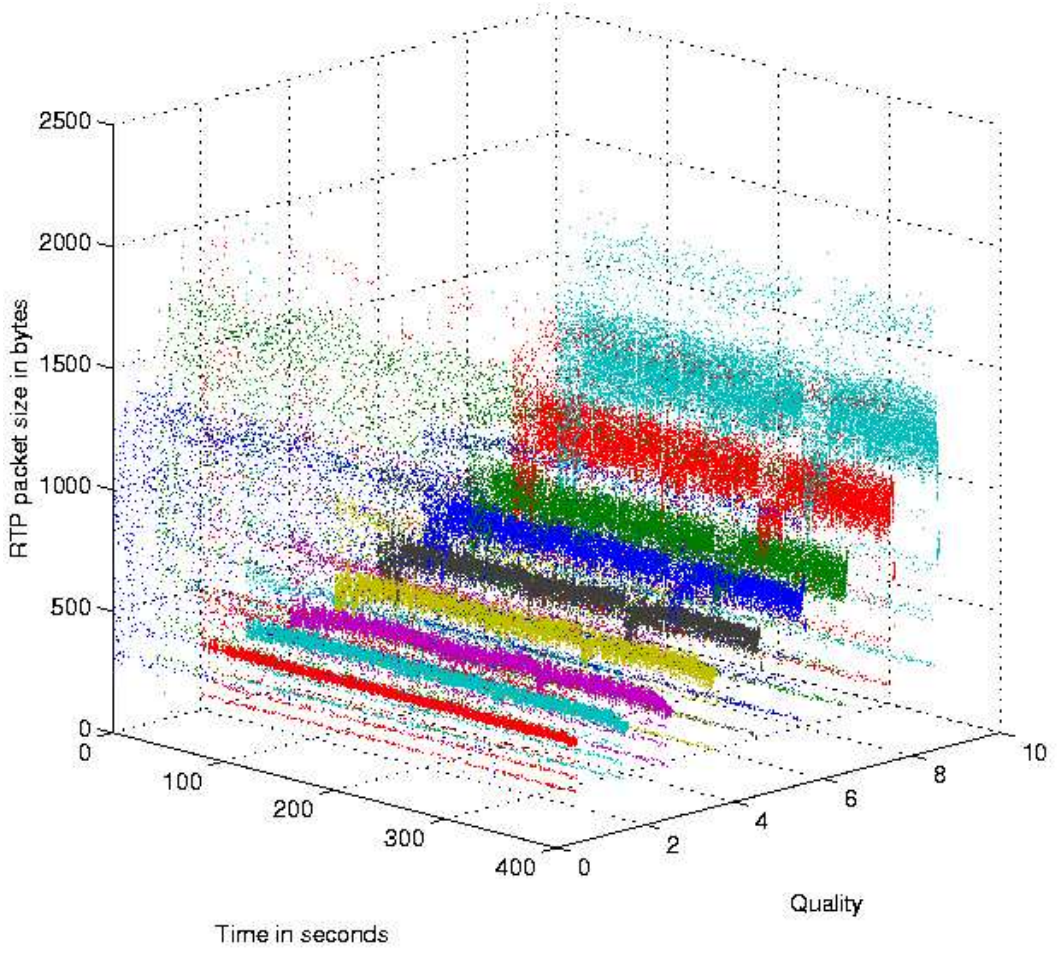

Figure 7.3: RTP packet size distribution of the example stream with different quality levels.

7.2.4 Comparison of Vorbis and RTP packets

Figure 7.4 shows a short period of the Vorbis stream encoded with quality 1.00. The same time period in a RTP stream is presented in figure 7.5. These two figures prove that the implementation really produces occasionally very big RTP packets.

The time period of the Vorbis stream in figure 7.4 has packets in size around 250 bytes and 50 bytes. The number of RTP packets are reduced, but the size of the RTP packets are increased up to 1900 bytes (see figure 7.5). RTP packets should be quite small and constant in size. The huge variation in RTP packet sizes increses the overall latency and the buffer utilization.

Of course, if the RTP packaging takes too much time in the sender side RTP packets might get dropped in the receiver end. Each RTP packet contains a timestamp of the first Vorbis packet inside the RTP packet. If the timestamp is too old the whole RTP packet is

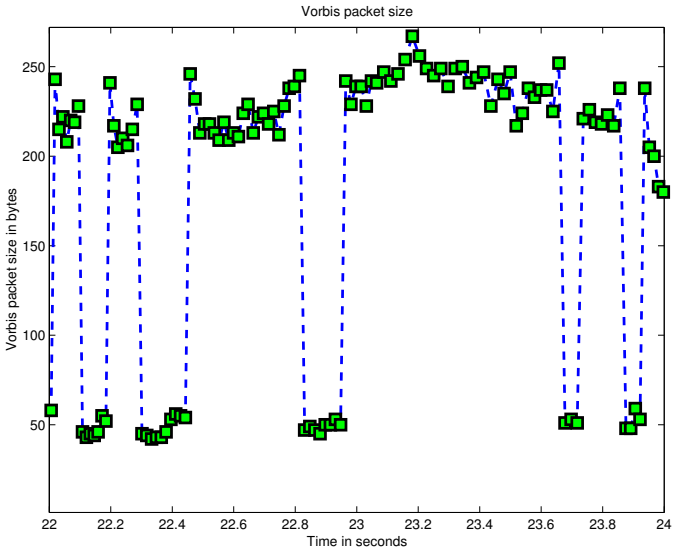

Figure 7.4: Vorbis packet sizes with stream quality 1.00.

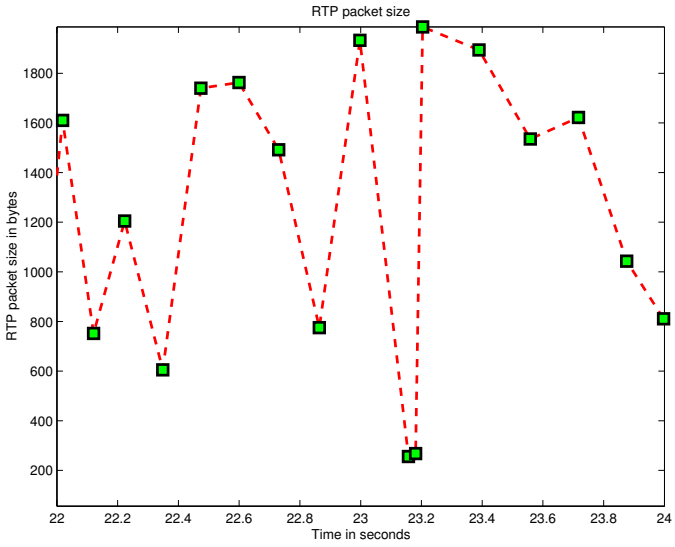

Figure 7.5: RTP packet sizes with stream quality 1.00.

treated as late arrived RTP packet and dropped. This could happen probably more often with Vorbis stream encoded with quality 3.00 or lower. The packet drop depends also on the size of receiving end buffers. Thus in the RTP payload draft the Vorbis packet size limit should be proportional to the encoding quality (see figure 7.6). In this figure “small” Vorbis packet size distribution is presented with different quality levels. A line has been added to separate these “small” Vorbis packets from the “big” ones. Thus the suggested function for

the Vorbis packet size maximum limit is:

$$s = 18 \cdot quality + 70, \tag{7.1}$$

where the $quality$ is the used encoding quality.

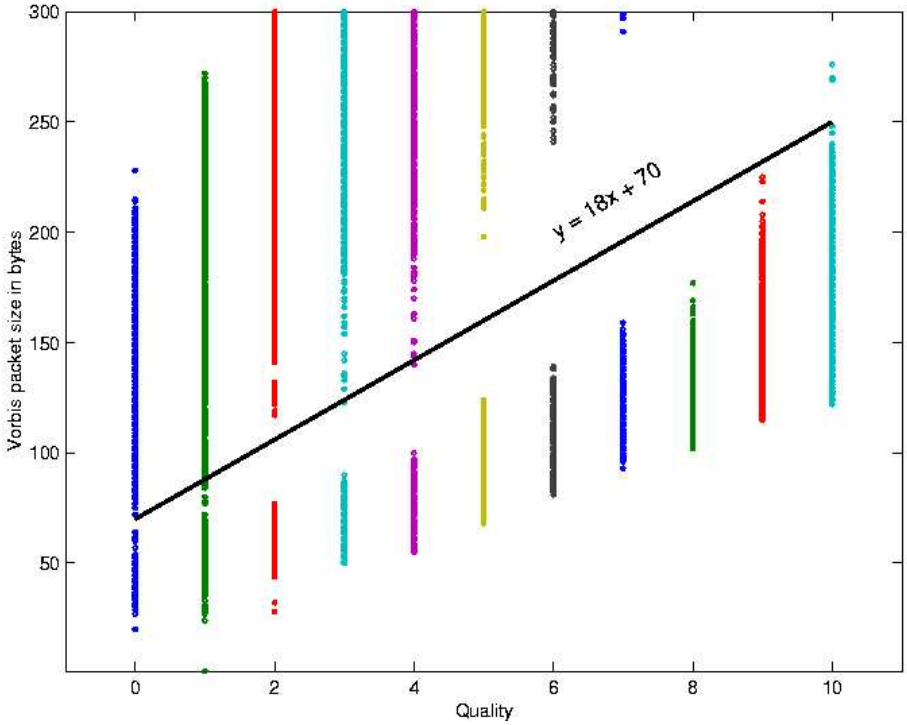

Figure 7.6: "Small" Vorbis packets size distribution with different quality levels.

7.2.5 Transmission overhead

In every RTP packet there is a total of 41 or 53 bytes of header information (20/32 bytes for IPv4/IPv6 header, 8 bytes for UDP header, 12 bytes for RTP header, and a one byte for Vorbis header). If one Vorbis packet is always packed into the RTP packet the largest overhead is produced.

$$D = \frac{header\ size}{payload\ size}. \tag{7.2}$$

Figure 7.7 shows the transmission overhead calculated from the equation 7.2. If n_p is greater than 10 transmission overhead is not reduced at all. Otherwise transmission overhead is reduced only little if n_p is increased. Thus, the amount of the transmission overhead is important only with low bit rates.

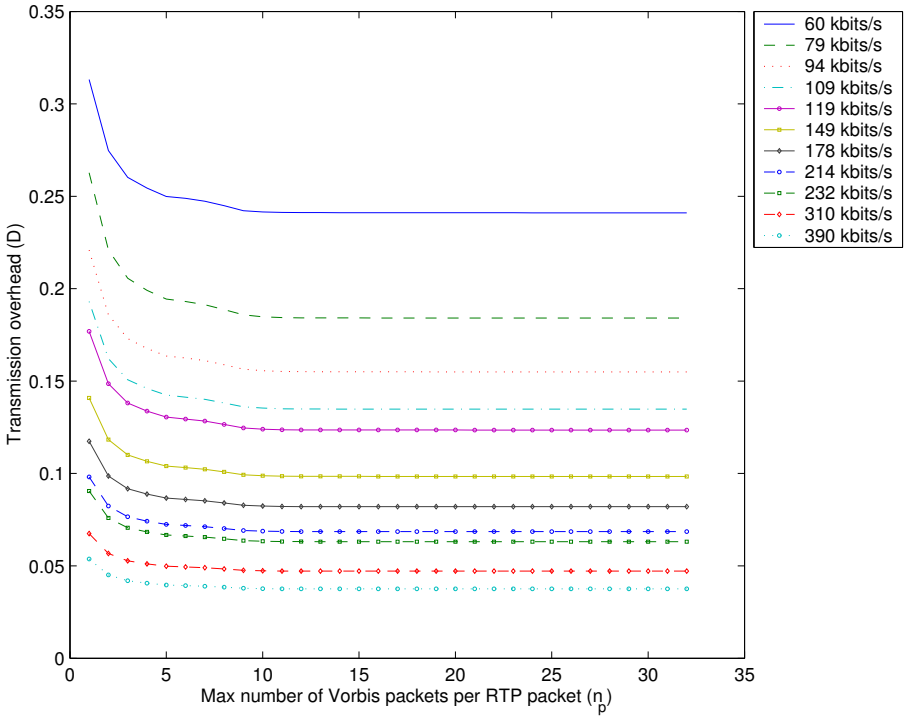

Figure 7.7: Transmission overhead in proportion to the maximum number of Vorbis packets inside each RTP packet with different encoding quality levels.

7.2.6 Interleaving

In MPEG-4 streaming with ISMA standard [20] it was noticed that if interleaving was used the perceived quality was improved considerably. However, the listening tests were out of the scope of this thesis, but it would be interesting to test different interleaving methods with the Ogg Vorbis media stream. The RTP payload format for interleaved encoded audio is described in [26].

7.2.7 Congestion control

Congestion control is needed in real-time streaming applications. Network conditions may change during the real-time music streaming and packet loss could occur. Especially breaks in music are irritating. In a study of a real-time audio streaming method for time-varying network loads [30] an additional protocol was used for the bandwidth estimation. The audio stream was then adapted to the estimated network bandwidth. This kind of adaptive streaming model is suitable for the Internet and could be used in modified format with Ogg Vorbis too.

The Ogg Vorbis codec supports bit rate peeling, at least in the time of writing this thesis,

which enables one to alter the bit rate on the fly without re-encoding the content. In congestion the sender could automatically lower the bit rate and smooth over the packet loss. And first of all, no additional protocol for bandwidth estimation need to be used. RTCP informs the received packet loss rate and an estimation of the network bandwidth could be derived from this packet loss rate. This could be specified as an optional feature and used only with RTCP.

The Vorbis packets could be truncated in size by dropping off the last bits according to equation 7.3.

$$\hat{s} = s \cdot quality, \tag{7.3}$$

where s is original Vorbis packet size, $\hat{s}$ is the modified packet size, and $0 \leq quality \leq 1$ is the quality value of the Vorbis packet. An initial value of the $quality$ is 1. $quality$ is modified by equations 7.5 and 7.4.

When the packet drop rate p is zero, $quality = quality + quality_{inc}$, but otherwise $quality = quality - quality_{dec}$. $quality_{inc}$ and $quality_{dec}$ are defined by equations 7.4 and 7.5.

$$quality_{inc} = k_{inc} \cdot RTT, \tag{7.4}$$

where k_{inc} is a constant ranging from 0 to 1 and RTT is the round-trip delay. A suggested value for k_{inc} is 0.03.

$$quality_{dec} = k_{dec} \cdot p, \tag{7.5}$$

where k_{dec} is a constant ranging from 0 to 1 and p is the observed packet drop rate. A suggested value for k_{dec} is 0.4.

Figure 7.8 shows how the congestion control algorithm reduces the stream bit rate when the packet drop occurs. If the average stream bit rate exceeds the simulated network bandwidth RTP packets are dropped. It can be noticed that if only a few packets are dropped the stream bit rate is reduced only a little, but in a bursty packet loss the stream bit rate is reduced in large steps. The congestion control algorithm cannot recover the lost packets, but when the packet loss occurs the algorithm prevents packet loss from continuing.

7.2.8 Packet loss recovery

When the packets are lost either some breaks or distortion will occur in music. To prevent these disturbances a separate packet recovery protocol should be implemented, but implementation of such protocol was out of the scope of this thesis. However, FEC is a good choice for recovering single packet losses in real-time applications. FEC can be used with

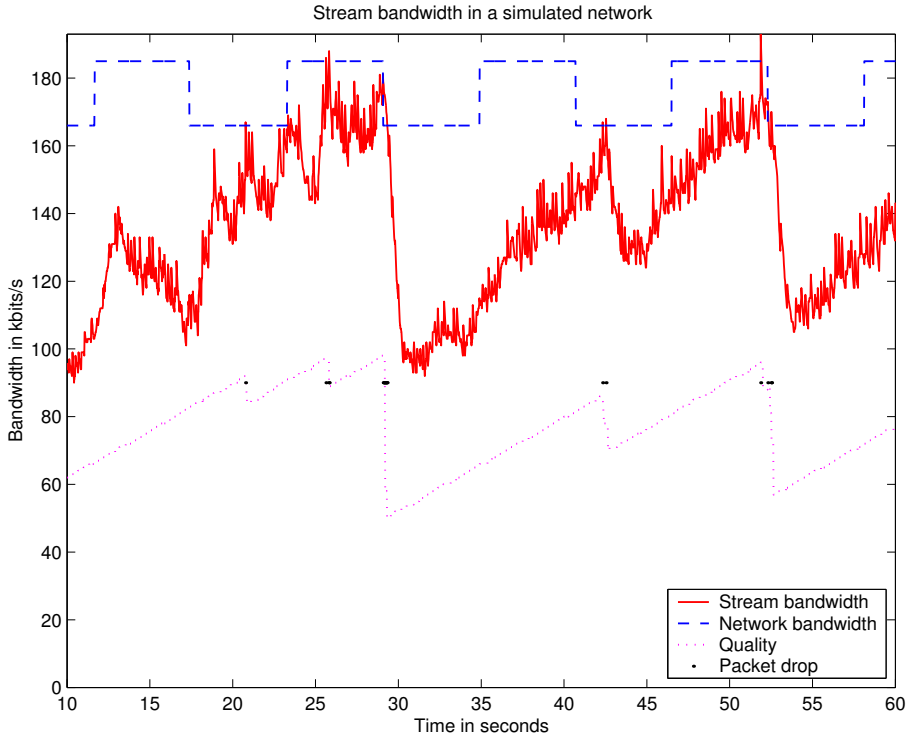

Figure 7.8: A simulated packet loss in a RTP Vorbis stream. Quality scale is between 0 and 100 %.

RTP and it has been described in RFC2733 [40]. For high-quality streaming purposes ARQ would be the best choice [53]. In the Vorbis RTP payload implementation silence is added in place of lost packets.

7.3 Summary

The streaming system was evaluated in regarding to the previously defined criteria. Modularity was achieved and new protocols could be added easily. The streaming system used only IETF specified protocols for signaling and media transport. When using the Ogg Vorbis audio codec as encoded media content format the whole streaming system was based on non-proprietary protocols and codecs. Thus, compatibility with IETF multimedia standards was achieved.

The proposed Vorbis codebooks packet distribution method within RTSP response was compared to the other possibilities. This RTSP method proved to be the best choice for this kind of streaming system in unicast environment. When using RTSP no other additional protocols are needed for the Vorbis codebooks distribution. For multicast streaming some

other, a more scalable distribution method, has to be defined. However, one solution for the Vorbis codebooks distribution in multicast streaming is to use all the time the same, static codebooks.

The evaluation of the system capabilites to transport CD-quality audio in the Internet led to inspection of properties of the *RTP payload format for Vorbis encoded audio*. CD-quality audio was achieved if no packet loss occured during the media transport in the Internet. In the case of packet loss, the lost packets were not recovered and only a silence was added in a place of dropped packets. If congestion occured in the network a burst loss of packets were experienced. To cope with varying network bandwidth situations an algorithm for reducing Vorbis stream bit rate was proposed. However, this is valid only if RTCP is used and if the final version of Ogg Vorbis audio codec supports the bit rate peeling.

RTP packaging process was also analyzed and some enhancements were made. However, these enhancements did not have affect to the perceived audio quality but made the transport of the Vorbis encoded audio more efficient. Vorbis RTP packets vary from few bytes to few kilobytes in size. This kind of RTP packets size variation can't be avoided with a simple RTP payload format. The existing RTP payload format is good enough for high bit rate Vorbis streams but with low bit rate streams some enhancements should be made.

In the example Vorbis stream the transmission overhead was reduced a little if up to 10 "small" Vorbis packets were bundeled into same RTP packets. However, the transmission overhead was in every case so small that there is no point of trying to reduce it even more.

In low bit rate Vorbis streams multiple Vorbis packets are always bundeled into the same RTP packet, because majority of the Vorbis packets are shorter than 256 bytes in size. In a case of one lost RTP packet, many Vorbis packets inside are missed and a long gap of silence is produced to the reconstructed stream. This could be avoided with limiting the maximum size of Vorbis packet, which can be bundeled to the RTP packet, with equation 7.1. The equation will always bundle the "small" Vorbis packets, but create separate RTP packets for every "big" Vorbis packet. In this way characteristics of low and high bit rate Vorbis streams are equal.

Chapter 8

Conclusions and Future Work

Audio streaming in the Internet is not a new invention and the first streaming services and applications were made many years ago. There are several commercial and non-commercial streaming server and client software on the market. Sometimes proprietary protocols and codecs are used, which makes these systems incompatible with the existing multimedia standards.

The objective of this thesis was to specify and implement a streaming system capable for high-quality audio streaming based on open and non-proprietary standards. The signaling between the server and the client was done with RTSP and the real-time Ogg Vorbis encoded audio was transported within RTP.

Different Vorbis codebooks packet distribution methods were studied. The use of RTSP as a transport protocol proved to be the best choice in unicast environment, but it won't scale to multicast systems.

The reference implementation proved that the Ogg Vorbis media content could be streamed in fast local networks without a perceived quality loss even with high bit rates. If packet loss occured gaps of silence were produced to the reconstructed stream. The gaps and experienced distortion will break the continuity of music, which is in high-quality music very annoying and unacceptable. In a network congestion burst losses of packets occur and even more disturbance is generated to decoded output. A congestion control method was proposed to cope with these time varying network loads. Feedback of the network congestion was given to the sender in a form of RTP packet drop rate. In a network congestion situation the sender was able to reduce the stream bit rate, but some RTP packets have to get lost first. Therefore, a packet recovery protocol has to be used to achieve a pleasant enjoyment of music.

Transmission overhead was also simulated and analyzed. The results showed that Ogg Vorbis streams don't produce major amounts of additional header overhead with any en-

coding quality. The transmission overhead is an important factor only with low bit rates.

Efficiency problems occured in RTP packet generation with low bit rate streams. In low bit rate Vorbis streams multiple Vorbis packets are always bundeled into the same RTP packet, because the majority of the Vorbis packets are shorter than 256 bytes in size. An equation was proposed to limit the maximum size of the Vorbis packet, which can be bundeled to the same RTP packet, proportional to the stream bit rate. The tests with the reference implementation indicated that the low bit rate RTP streams are then smoother and latency is slightly reduced.

8.1 Future work

The work with Ogg Vorbis should be continued and an updated version of the expired Internet-Draft *RTP payload format for Vorbis encoded audio* should be done according to the analysis of this thesis. However, Vorbis codebooks packet distribution in multicast streaming is still open.

A packet recovery protocol is an essential and needed property to fullfill the high-quality audio requirements. In the study of packet recovery protocols a decision to use FEC, ARQ or both should be made.

There has also been a study of MP3 wireless streaming over the Bluetooth Wireless Digital protocol [14], but an intention is to study Ogg Vorbis audio streaming in heterogenous Wireless Local Area Networks (WLAN) and GRPS networks. Especially in WLAN networks the packet recovery protocol would be needed.

Bibliography

[1] ISO/IEC JTC1/SC29 WG 11. Multimedia Content Description Interface (MPEG-7), December 2001. http://mpeg.telecomitalialab.com/standards/mpeg-7/mpeg-7.htm.

[2] J-J. Aucouturier and M. Sandler. Finding Repeating Patterns in Acoustic Musical Signals : Applications for Audio Thumbnailing. *22nd AES International Conference on Virtual, Synthetic and Entertainment Audio*, July 2002.

[3] C. Baltes. The E-Learning Balancing Act: Training and Education with Multimedia. *IEEE Multimedia*, 8(4):16–19, December 2001.

[4] R. Bargar, S. Church, A. Fukuda, J. Grunke, D. Keislar, B. Moses, and B. Novak. Networking Audio and Music Using Internet2 and Next-Generation Internet Capabilities. *J. of the Audio Engineering Society*, 47(4):300–310, April 1999.

[5] T. Berners-Lee, R. Fielding, and L. Masinter. RFC 2396: Uniform Resource Identifiers (URI): Generic syntax, August 1998.

[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. RFC 2475: An architecture for differentiated services, December 1998.

[7] K. Brandenburg. MP3 and AAC Explained. *17th AES International Conference on High-Quality Audio Coding*, September 1999.

[8] D. Bulterman, E. Delp, A. Eleftheriadis, P. Fernicola, R. Lanphier, and S-M Tan. Is Streaming Media Becoming Mainstream? *Association for Computing Machinery*, October 2001.

[9] B. Campbell and J. Rosenberg. Session initiation protocol extension for instant messaging. Internet Draft, Internet Engineering Task Force, http://www.ietf.org/internet-drafts/draft-ietf-sip-message-05.txt, June 2002. Work in progress, [Referenced 30.06.2002].

[10] Douglas E. Comer. *Internetworking with TCP/IP*, volume 1. Prentice Hall, 4 edition, 2000.

[11] S. Donovan. RFC 2976: The SIP INFO method, October 2000.

[12] M. Feerick. How Much Delay is Too Much Delay? *112th AES Convention*, May 2002.

[13] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616: Hypertext transfer protocol – HTTP/1.1, June 1999.

[14] A. Floros, M. Koutroubas, N-A. Tatlas, and J. Mourjopoulos. A Study of Wireless Compressed Digital-audio Transmission. *112th AES Convention*, May 2002.

[15] A.C.M. Fong and S.C. Hui. Low-Bandwidth Internet Streaming of Multimedia Lectures. *IEEE Engineering Science and Education Journal*, pages 212–218, December 2001.

[16] Xiph.Org Foundation. Homepage of Ogg Vorbis, February 2001. http://www.xiph.org/ogg/vorbis.

[17] M. Fowler and K. Scott. *UML Distilled*. Addison-Wesley, 1997.

[18] N. Freed and N. Borenstein. RFC 2045: Multipurpose Internet Mail Extensions (MIME) part one: Format of Internet message bodies, November 1996.

[19] N. Freed and N. Borenstein. RFC 2046: Multipurpose Internet Mail Extensions (MIME) part two: Media types, November 1996.

[20] B. Grill, T. Hahn, D. Homm, K. Krauss, G. Ohler, W. Sörgel, and C. Spitzner. Characteristics of Audio streaming over IP networks within the ISMA Standard. *112th AES Convention*, May 2002.

[21] Linley Gwennap. Linley on Linux: Linux Drives Digital Audio Revolution. *Linux journal*, 78:186, October 2000.

[22] M. Handley and V. Jacobson. RFC 2327: SDP: Session description protocol, April 1998.

[23] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. RFC 2543: SIP: session initiation protocol, March 1999.

[24] J. Herre. Audio Coding - An All-Round Entertainment Technology. *22nd AES International Conference on Virtual, Synthetic and Entertainment Audio*, July 2002.

[25] S. Herzog. RFC 2750: RSVP extensions for policy control, January 2000.

[26] O. Hodson. RTP payload for interleaved audio. Internet Draft, Internet Engineering Task Force, http://www.ietf.org/internet-drafts/draft-ietf-avt-rtp-interleave-00.txt, May 2001. Work in progress, [Referenced 30.06.2002].

[27] B. James. CD album Joy Ride, August 1999.

[28] D. Kutscher, J. Ott, and C. Bormann. Session description and capability negotiation. Internet Draft, Internet Engineering Task Force, http://www.ietf.org/internet-drafts/draft-ietf-mmusic-sdpng-04.txt, March 2002. Work in progress, [Referenced 30.06.2002].

[29] Frank LaMonica. Streaming Media. *Linux journal*, 81:108, 110, 112–114, January 2001.

[30] S-J. Lee, S-W. Kim, and E. Oh. A Real-Time Audio Streaming Method for Time-Varying Network Loads. *112th AES Convention*, May 2002.

[31] Microsoft. Homepage of Windows Media Techonologies, May 2002. http://www.microsoft.com/windows/windowsmedia/default.asp.

[32] J. Moffitt. RTP payload format for vorbis encoded audio. Internet Draft, Internet Engineering Task Force, http://www.xiph.org/ogg/vorbis/doc/draft-moffitt-vorbis-rtp-00.txt, February 2001. Work in progress, [Referenced 30.06.2002].

[33] Jack Moffitt. Ogg Vorbis — Open, Free Audio — Set Your Media Free. *Linux journal*, 81:146, 148–150, January 2001.

[34] S. Naegele-Jackson, N.L.M. Eschbaum, and P. Holleczek. Distributed Television Production for Distance Education with a Customizable Internet Platform. *Proceedings of the IEEE 12th International Workshop*, pages 838–842, September 2001.

[35] T. Painter and A. Spanias. Perceptual Coding of Digital Audio. *Proceedings of the IEEE*, 88(4):451–513, April 2000.

[36] C. Perkins and O. Hodson. RFC 2354: Options for repair of streaming media, June 1998.

[37] J. Postel. RFC 768: User datagram protocol, August 1980.

[38] J. Postel. RFC 793: Transmission control protocol, September 1981.

[39] A. Roach. SIP-specific event notification. Internet Draft, Internet Engineering Task Force, http://www.ietf.org/internet-drafts/draft-ietf-sip-events-05.txt, February 2002. Work in progress, [Referenced 30.06.2002].

[40] J. Rosenberg and H. Schulzrinne. RFC 2733: An RTP payload format for generic forward error correction, December 1999.

[41] J. Rosenberg and H. Schulzrinne. Reliability of provisional responses in SIP. Internet Draft, Internet Engineering Task Force, http://www.ietf.org/internet-drafts/draft-ietf-sip-100rel-06.txt, February 2002. Work in progress, [Referenced 30.06.2002].

[42] N. Rump, J. Herre, K. Brandenburg, and J. Koller. Legal Distribution of Music Through the Internet. *Proc. 106th Audio Engineering Society Convention*, May 1999. preprint 4988.

[43] A. Schapira, K. De Vries, and C. Pedregal-Martin. MANIC: An Open-Source System to Create and Deliver Courses over the Internet. *Proceedings of the IEEE 2001 Symposium*, pages 21–26, January 2001.

[44] H. Schulzrinne. RFC 1890: RTP profile for audio and video conferences with minimal control, January 1996.

[45] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 1889: RTP: A transport protocol for real-time applications, January 1996.

[46] H. Schulzrinne, A. Rao, and R. Lanphier. RFC 2326: Real time streaming protocol (RTSP), April 1998.

[47] H. Schulzrinne and J. Rosenberg. Internet Telephony: Architecture and Protocols, an IETF perspective, July 1998.

[48] SDMI. Homepage of Secure Digital Music Initiative (SDMI), June 2002. http://www.sdmi.org/.

[49] J. Selin. Media Management in IP Telephony System. Master's thesis, Helsinki University of Technology, February 2001.

[50] R. Sparks. SIP call control : Transfer. Internet Draft, Internet Engineering Task Force, http://www.ietf.org/internet-drafts/draft-sparks-sip-cc-transfer-05.txt, July 2002. Work in progress, [Referenced 30.06.2002].

[51] G. Stoll. Streaming-Audio@Internet: Perspectives for the Broadcasters. *17th AES International Conference*, September 1999.

[52] World Wide Web Consortium (w3c). Homepage of Synchronized Multimedia Integration Language (SMIL), June 2002. http://www.w3.org/AudioVideo/.

[53] A. Xu, W. Woszczyk, Z. Settel, B. Pennycook, R. Rowe, P. Galanter, and J. Bary. Real-Time Streaming of Multichannel Audio Data over the Internet. *J. of the Audio Engineering Society*, 48(7/8):627–641, August 2000.

[54] L. Xu. Efficient and Scalable On-Demand Data Streaming Using UEP Codes. *ACM Press Series Proceeding Session Article*, pages 70–78, 2001.

Appendix A

Signal Flow Examples

A.1 SIP phone call

Figure A.1 presents a typical SIP call signaling flow.

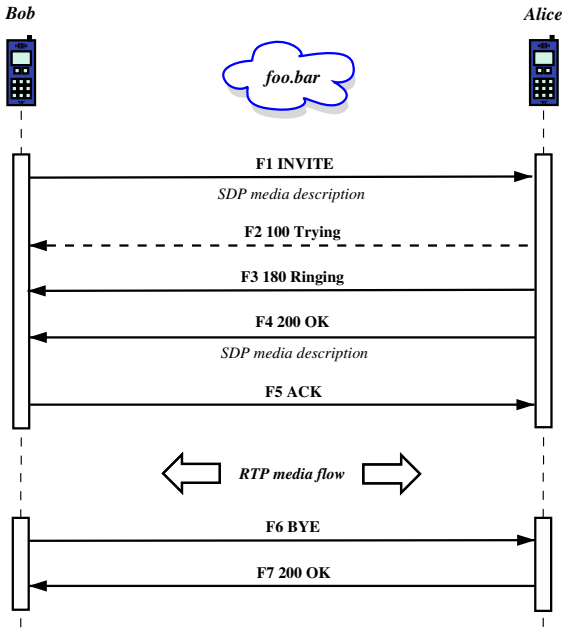

Figure A.1: A typical SIP call

Bob wants to call to Alice. Bob sends a SIP INVITE to Alice with a proper SDP in the payload. In the SDP Bob tells that he uses RTP for the content transport and supports PCMA or PCMU encoded audio. He is waiting for RTP packets in port 48 000.

```
F1 INVITE Bob -> Alice

INVITE sip:alice@foo.bar SIP/2.0
Via: SIP/2.0/UDP foo.bar:5060
From: Bob <sip:bob@foo.bar>
To: Alice <sip:alice@foo.nar>
Call-ID: 12345601@foo.bar
CSeq: 1 INVITE
Contact: <sip:bob@foo.bar>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844256 2890842807 IN IP6 fec0::202:b3ff:fe14:1073
s=Hello Alice
i=Phone call
t=0 0
c=IN IP6 fec0::202:b3ff:fe14:1073
m=audio 48000 RTP/AVP 0 8
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
```

When Alice receives Bob's invitation she automatically sends a 100 Trying response back to Bob.

```
F2 (100 Trying) Alice -> Bob

SIP/2.0 100 Trying
Via: SIP/2.0/UDP foo.bar:5060
From: Alice <sip:alice@foo.bar>
To: Bob <sip:bob@foo.bar>
Call-ID: 12345601@foo.bar
CSeq: 1 INVITE
Content-Length: 0
```

Alice's phone rings. A 180 Ringing response is sent to Bob.

```
F3 180 Ringing Alice -> Bob

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP foo.bar:5060
From: Alice <sip:alice@foo.bar>
To: Bob <sip:bob@foo.bar>; tag=8321234356
Call-ID: 12345601@foo.bar
CSeq: 1 INVITE
Content-Length: 0
```

Alice answers to the incoming call by sending back a 200 OK response. Alice's own SDP is attached to the payload. In the SDP only the information what Alice can receive is described. It shows in the SDP that Alice can receive RTP media flows, she supports PCMA or PCMU codecs, and she is waiting incoming RTP packets in port 58 000.

```
F4 200 OK Alice -> Bob

SIP/2.0 200 OK
Via: SIP/2.0/UDP foo.bar:5060
From: Alice <sip:alice@foo.bar>
To: Bob <sip:bob@foo.bar>
Call-ID: 12345601@foo.bar
CSeq: 1 INVITE
Contact: <sip:bob@foo.bar>
Content-Type: application/sdp
Content-Length: 147

v=0
o=alice 2890844256 2890842807 IN IP6 fec0::202:b3ff:fe14:811
s=Hello Bob
i=Phone call
t=0 0
c=IN IP6 fec0::202:b3ff:fe14:811
m=audio 58000 RTP/AVP 0 8
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
```

Bob receives Alice's session description and accepts the parameters by sending an ACK response. At the moment RTP media flow is established between the callers.

```
F5 ACK Bob -> Alice

ACK sip:alice@foo.bar SIP/2.0
Via: SIP/2.0/UDP foo.bar:5060
From: Bob <sip:bob@foo.bar>
To: Alice <sip:alice@foo.nar>
Call-ID: 12345601@foo.bar
CSeq: 1 ACK
Content-Length: 0
```

The SIP phone call can be ended by one of the caller parties by sending a BYE request. Bob wants to hangup the phone call and he sends a BYE request to Alice.

```
F6 BYE Alice -> Bob

BYE sip:alice@foo.bar SIP/2.0
Via: SIP/2.0/UDP foo.bar:5060
From: Alice <sip:alice@foo.bar>
To: Bob <sip:bob@foo.bar>
Call-ID: 12345601@foo.bar
CSeq: 2 BYE
Content-Length: 0
```

Alice respondes to the hangup request by sending back a 200 OK response.

```
F7 200 OK Bob -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/UDP foo.bar:5060
From: Bob <sip:bob@foo.bar>
To: Alice <sip:alice@foo.nar>
Call-ID: 12345601@foo.bar
CSeq: 2 BYE
Content-Length: 0
```

The SIP phone call is over. The SIP call leg is destroyed and the RTP media flow is ended.

Appendix B

An Example of Audio Stream

The *Bob James - Take Me There* [27] CD audio track was used in every example in this thesis, but sometime with different encoding quality settings. The audio stream from the album *Joy Ride* made by *Bob James* in 1999.

B.1 Vorbis packet sizes

The audio track encoded with the best quality is presented in figure B.1. The figure shows that there are quite many big Vorbis packets in size around 1.2 kbytes, but also a lot of smaller packets in size of around 200 bytes. The same track encoded with the lowest possible quality is presented in a figure B.2. It can be seen that these big Vorbis packet sizes are now around 170 bytes and small ones under 50 bytes. Bit rates of these streams are also presented in figures B.3 and B.4.

How these Vorbis streams are packed to the RTP packets are presented in figures B.6 and B.5. With the highest encoding quality RTP packets are nicely spread in around 1100 bytes. However, with the lowest encoding quality all the RTP packets are spread in range from few bytes up to 1400 bytes.

B.2 Stream information

```
title=Take Me There
artist=Bob James
album=Joy Ride
Bitstream is 2 channel, 44100Hz
Decoded length: 15203916 samples
Encoded by: Xiphophorus libVorbis I 20011231
```

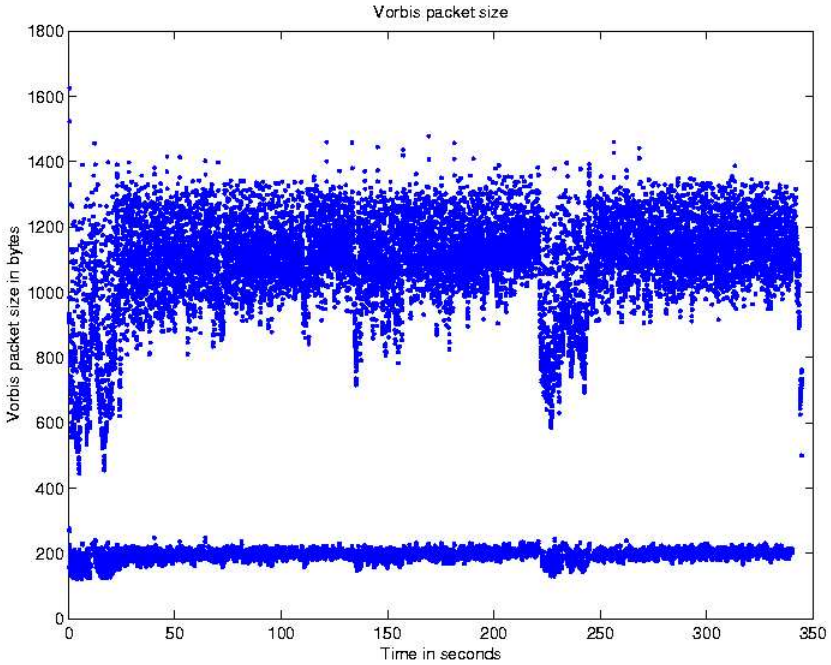

Figure B.1: Vorbis packet size distribution with the highest encoding quality 10. The stream average bitrate was 390 kbits/s.

```
Input bitstream contained 1 logical bitstream section(s)
Total bitstream playing time: 344.76 seconds
Logical bitstream information fromsection 1
Rate 44100Hz channels 2  bitrate 343kbps serialnumber 151540532
Header length: 3924 bytes
Compressed length: 14827307 bytes
Play time: 344.76s
```

B.3 Statistics of the stream encoded with quality 10.00

```
Vorbis packets:
        number of vorbis packets        21810
        under 256 bytes                 7953
        min Vorbis size                 122
        max Vorbis size                 1625
        average Vorbis size             763
        total bytes of Vorbis data      16640496
Sent RTP packets:
```

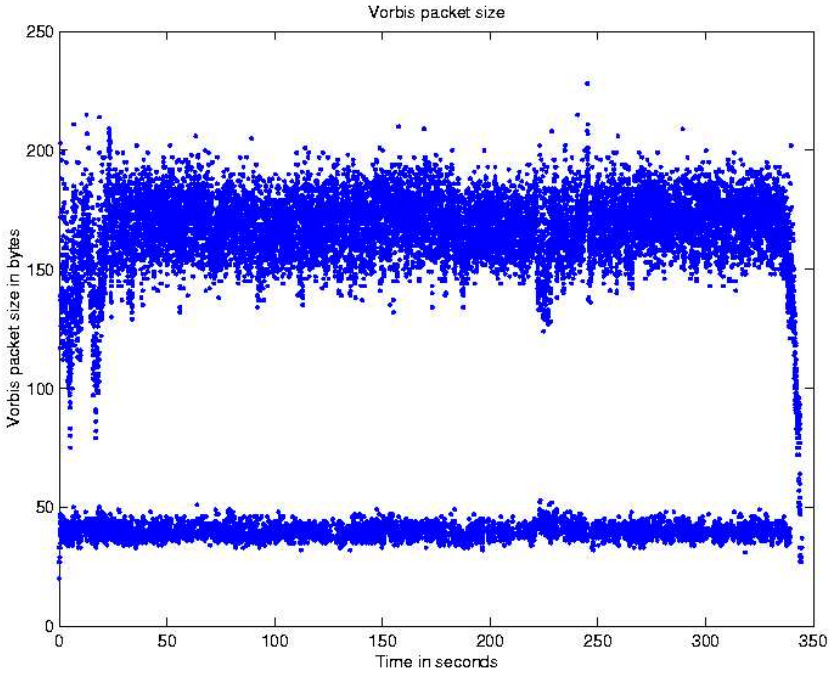

Figure B.2: Vorbis packet size distribution with the lowest encoding quality 0. The stream average bitrate was 60 kbits/s.

```
        number of RTP packets           15422
        min RTP size                    128
        max RTP size                    1788
        average RTP size                1081
RTP packets dropped:
        number of dropped RTP packets   0
Received RTP packets:
        number of RTP packets           15422
        min RTP size                    128
        max RTP size                    1788
        average RTP size                1081
RTP jitter buffer:
        min rtp jitter size             1
        max rtp jitter size             55
        average rtp jitter size         47
        rtp jitter standard deviation   2
        rtp jitter variation            3
```

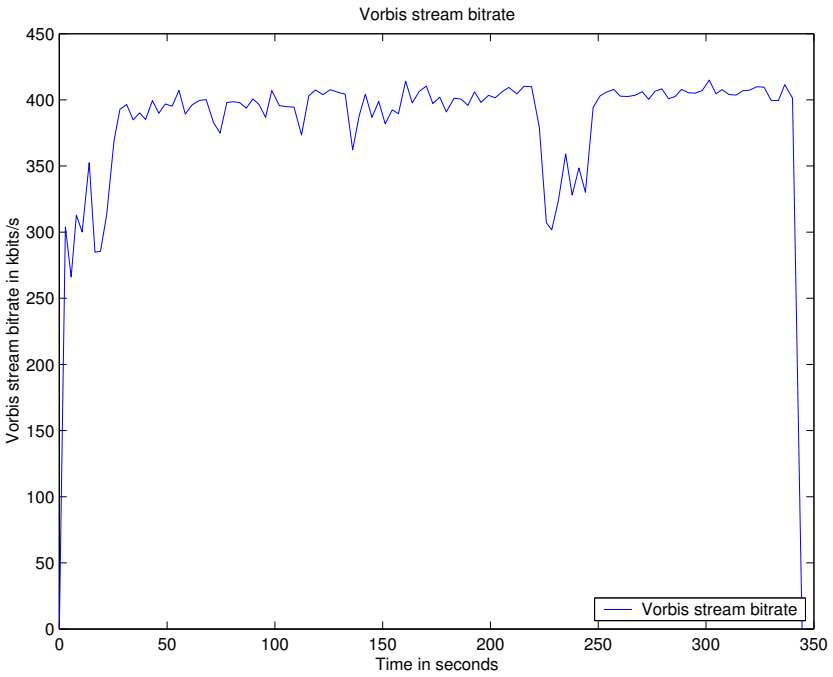

Figure B.3: An example of Vorbis stream bit rate encoded with quality 10.00.

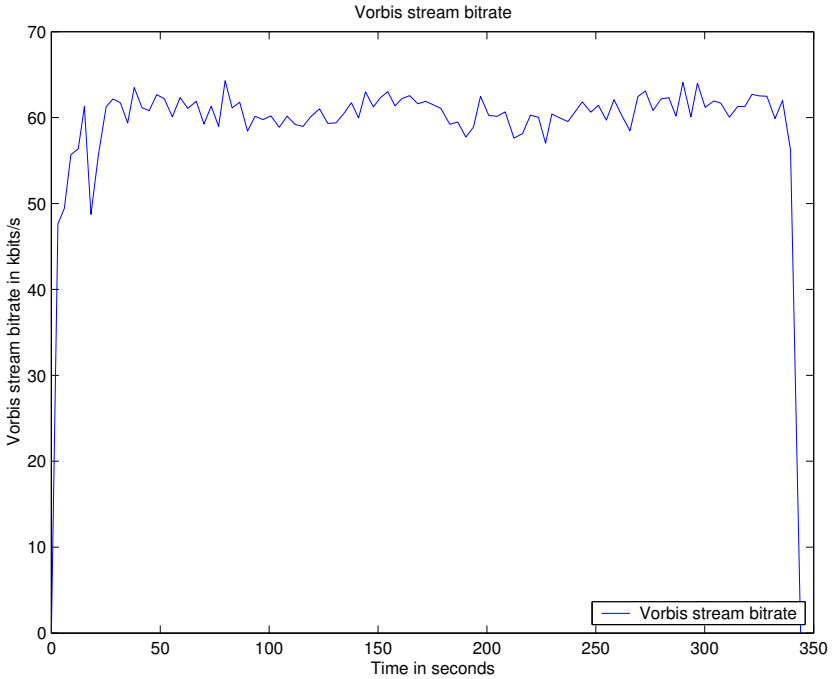

Figure B.4: An example of Vorbis stream bit rate encoded with quality 0.00.

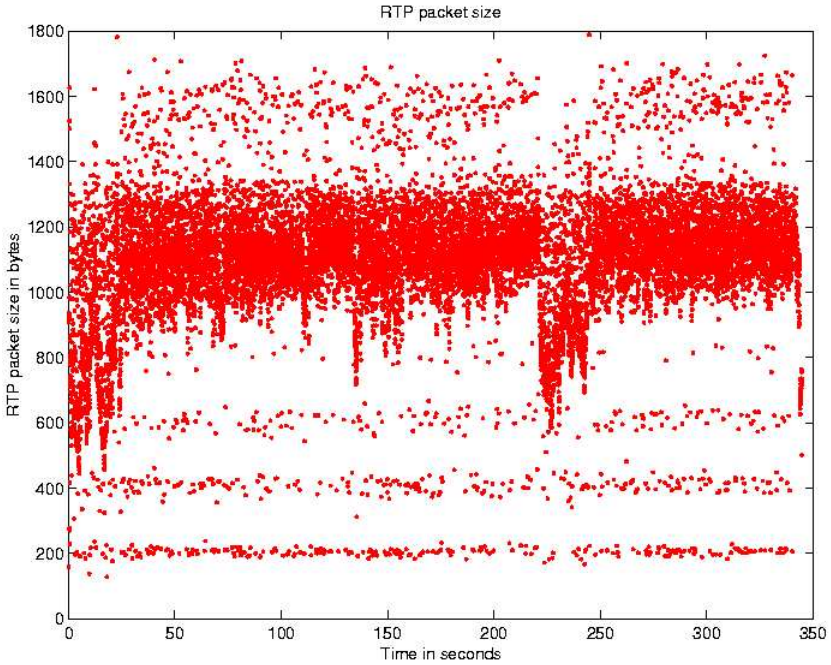

Figure B.5: RTP packet size distribution with the highest encoding quality 10. The stream average bitrate was 390 kbits/s.

```
PCM jitter buffer:
        min pcm jitter size             0
        max pcm jitter size             106
        average pcm jitter size         64
        pcm jitter standard deviation   10
        pcm jitter variation            102
Encoding delay:
        min delay                       0.00ms
        max delay                       0.30ms
        average delay                   0.01ms
Build payload delay:
        min delay                       0.09ms
        max delay                       8.98ms
        average delay                   0.11ms
Strip payload delay:
        min delay                       0.01ms
        max delay                       0.14ms
        average delay                   0.02ms
```

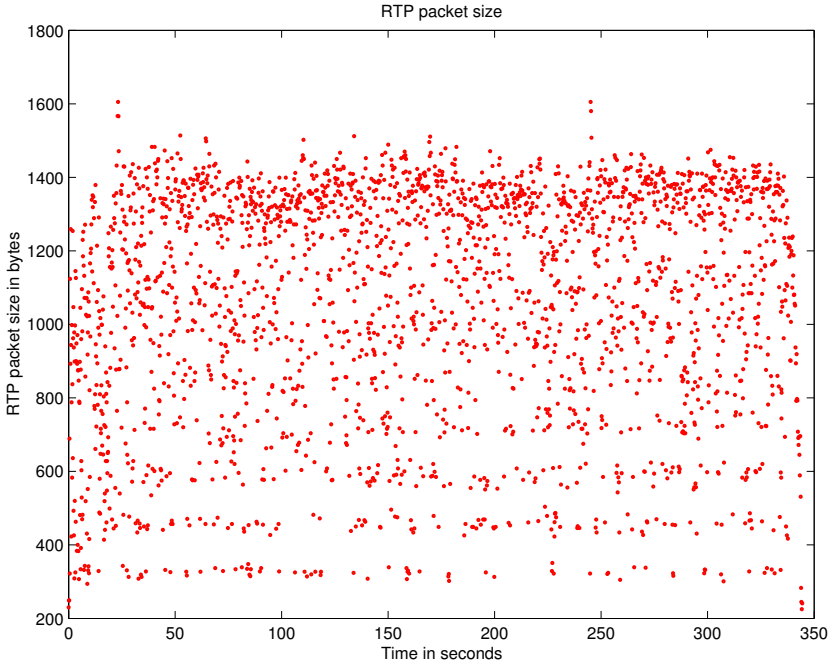

Figure B.6: RTP packet size distribution with the lowest encoding quality 0. The stream average bitrate was 60 kbits/s.

```
Decoding delay:
        min delay                       0.55ms
        max delay                       47.01ms
        average delay                   1.86ms
```