

Saarland University  
Faculty of Natural Sciences and Technology I  
Department of Computer Science  
Master's Program in Computer Science

Master's Thesis

**Design and Development of a Quality of  
Service Framework for the  
Network-Integrated Multimedia  
Middleware (NMM)**

submitted by

**Martin Thielen**

on May 30, 2007

Supervisor

Prof. Dr. Ing. Philipp Slusallek

Advisor

Dipl. Inform. Michael Repplinger

Reviewers

Prof. Dr.-Ing. Philipp Slusallek

Prof. Dr.-Ing. Thorsten Herfet



### **Statement**

Hereby I confirm that this thesis is my own work and that I have documented all sources used.

Saarbrücken, May 30, 2007

### **Declaration of Consent**

Herewith I agree that my thesis will be made available through the library of the Computer Science Department.

Saarbrücken, May 30, 2007



# Acknowledgments

I would like to thank ...

Professor Slusallek for allowing this Master's Thesis.

Michael Repplinger for the excellent supervision.

all members of the NMM-group and the system administrators of the Computer Graphics Chair for their support.

my family and friends for their encouragement during difficult situations.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Objectives . . . . .	3
1.3	Structure . . . . .	3
<b>2</b>	<b>Fundamentals of Multimedia and Networking</b>	<b>5</b>
2.1	Multimedia Basics . . . . .	5
2.1.1	Digitalization . . . . .	6
2.1.2	Audio Compression . . . . .	9
2.1.3	Introduction to Digital Video . . . . .	11
2.2	Network Protocols . . . . .	13
2.2.1	Transmission Control Protocol (TCP) . . . . .	14
2.2.2	User Datagram Protocol (UDP) . . . . .	15
2.2.3	Real-Time Transport Protocol (RTP) . . . . .	17
2.2.4	Real-Time Transport Control Protocol (RTCP) . . . . .	19
<b>3</b>	<b>Related Work</b>	<b>23</b>
3.1	Quality of Service (QoS) . . . . .	23
3.1.1	Abstract Definition and Motivation . . . . .	23
3.1.2	Approaches . . . . .	24
3.1.3	Conclusion . . . . .	32
3.2	Tools for Evaluation of Adaptation Techniques . . . . .	33
<b>4</b>	<b>Environment for Development and Evaluation</b>	<b>35</b>
4.1	Setup . . . . .	35
4.2	Installation of NIST Net . . . . .	37
4.2.1	Prerequisites . . . . .	38
4.2.2	Installation . . . . .	38
4.3	Usage of NIST Net . . . . .	39
4.3.1	Controlling NIST Net . . . . .	39
4.3.2	Emulating Certain Network Conditions . . . . .	40
4.3.3	Statistics, Settings and Debug Messages . . . . .	42
4.3.4	Miscellaneous . . . . .	42

<b>5</b>	<b>NMM - General Overview</b>	<b>45</b>
5.1	Multimedia Flow Graph . . . . .	46
5.2	Nodes . . . . .	46
5.2.1	Formats for Interconnection . . . . .	47
5.2.2	Jacks . . . . .	47
5.2.3	Types of Nodes . . . . .	48
5.2.4	State of a Node . . . . .	49
5.3	Distributed Communication . . . . .	50
5.3.1	Types of Messages . . . . .	50
5.3.2	Transport-Strategies . . . . .	51
5.3.3	RTP in NMM . . . . .	51
5.3.4	Serialization . . . . .	52
5.3.5	Transparent Network Access . . . . .	52
5.4	Middleware Services and Applications . . . . .	54
5.4.1	Device Management . . . . .	54
5.4.2	Middleware Services . . . . .	54
<b>6</b>	<b>Integration of AMR into NMM</b>	<b>57</b>
6.1	Encoding and Decoding of AMR Codecs . . . . .	57
6.1.1	Plug-ins for Encoding and Decoding . . . . .	57
6.1.2	A Format for AMR . . . . .	60
6.2	Streaming of AMR-NB and AMR-WB with RTP . . . . .	60
6.2.1	Modifications in Class BufferedSource and Class RTP-SenderStrategy . . . . .	61
6.2.2	Modifications in the Class BufferedSink and the Class RTPReceiverStrategy . . . . .	63
6.2.3	Modifications in the Class RTPStrategy and the Class RTPPayloadtypeFormatMatching . . . . .	63
6.3	Summary . . . . .	64
<b>7</b>	<b>A QoS Framework for NMM</b>	<b>65</b>
7.1	Goals . . . . .	65
7.2	Scenarios . . . . .	66
7.3	Design Concepts . . . . .	68
7.4	Overall Communication . . . . .	71
7.4.1	Registration of Adaptation-Strategies at QoS-Monitors . . . . .	72
7.4.2	Gathering Data About the Current Network State . . . . .	75
7.4.3	Transmission of Network State Data to the Adaptor . . . . .	77
7.4.4	Determination of Suitable Adaptation Techniques . . . . .	78
7.4.5	Applying Adaptation Techniques . . . . .	79
7.5	The Class QoSReport . . . . .	79
7.5.1	QoSReportValueFlag . . . . .	81
7.5.2	Serialization of QoSReport . . . . .	82
7.6	The Class CompositeAdaptationStrategy . . . . .	82

7.7	Summary . . . . .	83
<b>8</b>	<b>Strategies for Adaptation and Evaluation</b>	<b>85</b>
8.1	Strategies for Evaluation Purposes . . . . .	85
8.2	Adaptation Time . . . . .	87
8.3	AudioBitrateAdaptationStrategy . . . . .	89
8.3.1	AMR-NB Streaming without Adaptation . . . . .	89
8.3.2	AMR-WB Streaming without Adaptation . . . . .	92
8.3.3	Adaptation Logic . . . . .	93
8.3.4	AMR-NB Streaming with Adaptation . . . . .	95
8.3.5	AMR-WB Streaming with Adaptation . . . . .	98
8.4	VideoBitrateAdaptationStrategy . . . . .	100
8.4.1	Video Streaming without Adaptation . . . . .	100
8.4.2	Adaptation Logic . . . . .	101
8.4.3	Video Streaming with Adaptation . . . . .	103
8.5	FramerateAdaptationStrategy . . . . .	106
8.6	BitrateFramerateAdaptationStrategy . . . . .	107
8.7	Summary . . . . .	108
<b>9</b>	<b>An Adaptive Streaming Application</b>	<b>109</b>
9.1	Structure . . . . .	110
9.2	Command Line Parameters . . . . .	114
<b>10</b>	<b>Summary and Future Work</b>	<b>117</b>
10.1	Summary . . . . .	117
10.2	Enhancements and Future Work . . . . .	118
	<b>List of Figures</b>	<b>121</b>
	<b>Bibliography</b>	<b>125</b>
	<b>A Acronyms</b>	<b>133</b>



# Chapter 1

## Introduction

### 1.1 Motivation

In the last years, the utilization of multimedia content in the Internet has grown more and more. An increasing number of users deploy multimedia applications such as Voice over Internet Protocol (VoIP) and videophone applications. Moreover, the transmission of TV programs over IP-based networks – called Internet Protocol Television (IPTV) – is emerging. TV channels extend their presence in the Internet and offer streaming of various programs. Recently, also TV services such as Joost [29] and Vudu [67] are trying to revolutionize TV. By transmitting their program over the Internet, they can offer on-demand videos and interactive programs. In order to compete with the traditional TV service and to meet the expectations of consumers, a high video quality is required. At the same time, an increasing number of mobile devices are capable of using the Internet and receiving TV.

However, the Internet is designed to deliver a best-effort service and does not support timing guarantees, which are often required for multimedia applications. For example, a high utilization of a network link can lead to congestion, which introduces delay and packet loss. Lost packets or packets that arrive too late at the receiving side greatly reduce the quality. Hence, congestion is a big problem for high quality multimedia applications as a continuous data stream is required.

Depending on the type of network a device is using, different transmission rates are possible. For example, comparing DSL, WLAN and UMTS, the available bandwidth varies greatly. Using wireless networks, also signal strength and interferences highly influence the overall network performance. Especially for mobile users, the variation of signal strength and available bandwidth is often appearing during runtime of the application.

Applications utilizing the File Transfer Protocol (FTP) often generate high traffic. However, they are not very time sensitive. In contrast, the multimedia applications outlined above generate not only high data traffic but are very time sensitive as well. Especially digital video or high quality digital audio requires a large amount of bandwidth. Nevertheless, these applications mostly do not pay regard to different network scenarios and changing network conditions. If any adjustment is offered, only a coarse grained configuration for video streams is possible. In this case, a number of streams with the same content for different bandwidths are provided. However, this approach is not sufficient as changing conditions during transmission are not considered. Especially if mobile devices with wireless connections are used, these changes can emerge rapidly.

In order to solve these problems and guarantee a certain quality degree, several approaches have been undertaken to introduce prioritization of packets or reservation of available bandwidth into network protocols. While these approaches work in local networks, up to the present, these approaches are not crowned with success in today's Internet.

Another approach to solve the outlined problems is adaptation. Adaptation is a technique to adjust parameters of the current connection when network conditions are changing. This includes several methods such as reduction of quality, switching between different codecs or partly deactivation of a connection.

The Real-Time Transport Protocol (RTP) is a network protocol especially designed for the purpose of multimedia streaming. *Streaming* is the continuous transmission of audio and video data across a network. RTP does not provide timing and transmission guarantees but offers features for estimating several parameters of the current network condition. Via the Real-Time Transport Control Protocol (RTCP), a feedback mechanism for transmission of parameters between the participating hosts is provided. Therefore, it is often used as transmission protocol for applications that provide adaptation techniques. However, RTP can only provide a limited number of parameters. For example, the signal strength of a WLAN cannot be measured directly with RTP.

Therefore, other components must be utilized that provide this information. In order to consider as many environmental parameters as possible, the access to all participating network components is required. For that reason, the use of a network middleware is reasonable.

Different from the streaming approach, a middleware allows for transparent access to the used network components. Even though underlying network details should be hidden by a middleware, a black box design is not suitable. Rather, a white box design is required for gaining direct access to

the different components. The Network-Integrated Multimedia Middleware is a middleware developed at the Computer Graphics chair of the Saarland University. As NMM uses a white box design and offers particular features for multimedia transmission, it provides a good basis for integration of adaptation techniques.

## 1.2 Objectives

The goal of this thesis is the development of adequate mechanisms for integration of adaptation techniques into NMM. These mechanisms should be realized as a framework, which especially focuses on extendability and modularity. In order to allow for a wide variety of application scenarios, the design of the framework should be flexible and not restricted to specific adaptation techniques or codecs. Furthermore, the integration of the framework into existing applications should be simple.

As changes of network conditions in the Internet are unpredictable, an appropriate environment must be set up, which permits reproducible and predictable results. Using this environment, network conditions should be evaluated in order to design and develop adaptation techniques. Both techniques for adaptation of video streams as well as audio streams should be developed. Therefore, codecs especially suitable for adaptation should be chosen and integrated into NMM.

By means of RTP and the feedback mechanism of RTCP, the estimation of network conditions should be realized. Appropriate interfaces should be designed and developed that allow for monitoring of network parameters such as packet loss, delay and jitter.

These network parameters should be used to determine suitable adjustments which should be integrated into adaptation mechanisms for video and audio connections.

For demonstration purposes, an adaptive streaming application should be developed. By integration of the developed framework, the quality enhancements achievable by using the adaptation techniques for audio and video connections should be presented. Furthermore, the application should be useful for statistical evaluation of adaptation techniques and future usage.

## 1.3 Structure

The second chapter is divided into two parts. The first part introduces to fundamentals of multimedia. Both basics about digital audio as well as

digital video are outlined. In the second part, several networking protocols are introduced and their different properties and applications are depicted.

In the third chapter, related work is presented and discussed. Furthermore, the proposals are compared to the requirements outlined above. Apart from that, some tools for networking are briefly introduced.

In the fourth chapter, the environment for evaluation and development, which has been set up for the scope of this thesis, is described.

The Network-Integrated Multimedia Middleware is outlined in chapter five. The fundamental architecture, concepts and several components are depicted. Special regard is paid to components that are relevant for the work presented in this thesis.

The sixth chapter covers the integration of the codecs AMR-NB and AMR-WB into NMM. These codecs are used for audio adaptation in the scope of this work. Two Plug-ins developed for encoding of both codecs are presented. Furthermore, the modifications to the existing design required for streaming of AMR-NB and AMR-WB via RTP are described.

The framework developed in this thesis is presented in chapter seven. First, the aims of the framework are discussed in detail. Afterwards, the concept and all components are outlined. Finally, the different components are described in more detail.

In the eighth chapter, adaptation techniques developed in the scope of this thesis are introduced. Moreover, the evaluation results are presented.

The ninth chapter covers the developed adaptive streaming application. First, the structure is illustrated. Afterwards, the available commands during runtime as well as command line parameters are described.

In the last chapter, the presented thesis is summarized and the achieved objectives are discussed. Moreover, possible future extensions and enhancements are outlined.

## Chapter 2

# Fundamentals of Multimedia and Networking

The first part of this chapter covers fundamentals of multimedia which are required in the following of this thesis. Basics about digitalization are addressed and some audio and video formats are introduced. In the second part, several common network protocols are described. Furthermore, the differences and purposes of these protocols are discussed.

### 2.1 Multimedia Basics

Often, the term multimedia is used imprecisely for the representation of audio and video. In the scope of this thesis, the definition of a multimedia system according to [65] is applied:

“A multimedia system is characterized by computer-controlled, integrated production, manipulation, presentation, storage and communicating of independent information, which is encoded at least through a continuous (time-dependent) and a discrete (time-independent) medium.”

This definition places emphasis on the aspect of processing by a computer and restricts the term multimedia to applications that contain at least one continuous medium (such as audio and video) and not only discrete data (such as text and images). Especially with regard to the topic of this thesis, this restricted view of a multimedia system is reasonable as discrete data is not very liable to timing constraints.

In the following of this section, fundamentals of multimedia are discussed. At first, digitalization of analog signals is outlined. Afterwards, a

basic overview of common techniques, formats and standards used in the area of audio and video compression is given.

### 2.1.1 Digitalization

With the introduction of Compact Discs (CDs) in the early 1980s the success of digital audio in the mass-market was initiated. At least since the Digital Versatile Disc (DVD) was established, also digital video was beginning to replace analog video.

Figure 2.1 illustrates a generic analog to digital converter. In the following, by means of digital audio, a basic introduction to the different parts of digitalization is given.

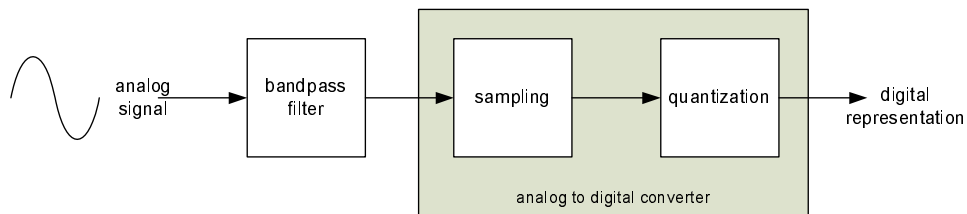


Figure 2.1: Generic analog to digital encoder. The frequencies of the analog signal are limited by a filter. Afterwards, the analog signal is sampled at a fixed rate. The output after quantization is the digital representation of the analog signal. The prototype of the figure is taken from [6].

### Sampling

Since computers can only handle discrete data, the first step from an analog signal to a digital signal is *sampling*. A *sample* of an analog signal is the measurement of an analog signal at a specific point in time. The number of samples taken per second is called *sampling rate*. It is measured in hertz (Hz).

For being able to reconstruct the original signal from the samples without loss, the sampling rate must satisfy the requirement of the *Nyquist-Shannon-Theorem* [72]:

$$f_{\text{sampling}} > 2 \cdot (f_{\text{max}} - f_{\text{min}})$$

Here, the sampling frequency (sampling rate) is denoted by  $f_{\text{sampling}}$ , and the maximal frequency and minimal frequency contained in the analog signal are referred to as  $f_{\text{max}}$  and  $f_{\text{min}}$ , respectively. Sampling rates below the threshold of  $f_{\text{sampling}}$  lead to distortions during reconstruction. Figure 2.2

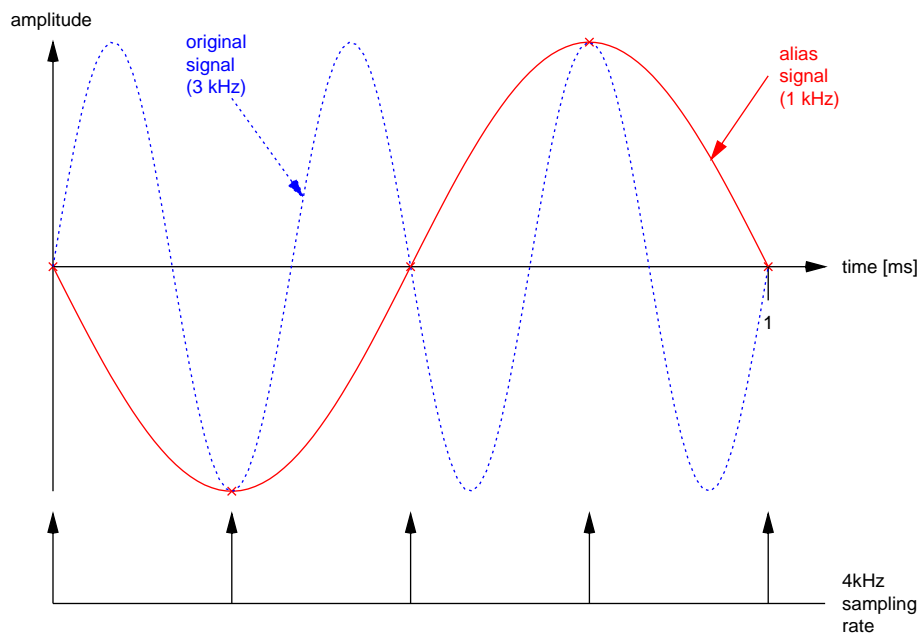


Figure 2.2: Alias signal resulting from a sampling frequency below the Nyquist-Shannon Criterion. The original signal (3 kHz) is sampled at a frequency of 4 kHz which results in an alias signal of 1 kHz. The prototype of the figure is taken from [6]

illustrates the emerging of an alias signal for a sampling rate which is below the Nyquist-Shannon Criterion.

Analog signals appearing in nature often contain a large scale of frequencies. Obviously, higher sampling rates increase the amount of digital data. Furthermore, the frequency of sampling is technically limited. Therefore, band-pass filters or low-pass filters are used to remove frequencies outside of a specified range. This prevents aliasing which otherwise would occur. Of course, a filter changes the original signal and can only be used reasonably if the removed frequencies are insignificant. Hence, sampling rates commonly used for digitalization, depend on the specific purpose (see Figure 2.3).

Sampling Rate	Usage
8 kHz	Telephony
41.1 kHz	Audio CD (Music)
48 kHz	Audio tracks on DVD-Video
up to 192kHz	DVD-Audio, HD-DVD

Figure 2.3: Sampling rates and typical usage for digital audio.

For digitalization of human speech, for instance, a sampling rate of 8000 Hz is often sufficient. This is due to the fact that using a passband of 200 Hz to 3400 Hz results in acceptable speech quality (commonly used in telephony) [9]. As the richness of a sound generated by an instrument is dependent on the overtones of the fundamental frequency, a large scale of significant frequencies are commonly inherent in music [74]. Therefore, sampling music at a rate of 8000 Hz results in a great loss of quality.

### Quantization

The analog samples contain continuous data which must be approximated to discrete values for digitalization. This process, called *quantization*, is illustrated in Figure 2.4.

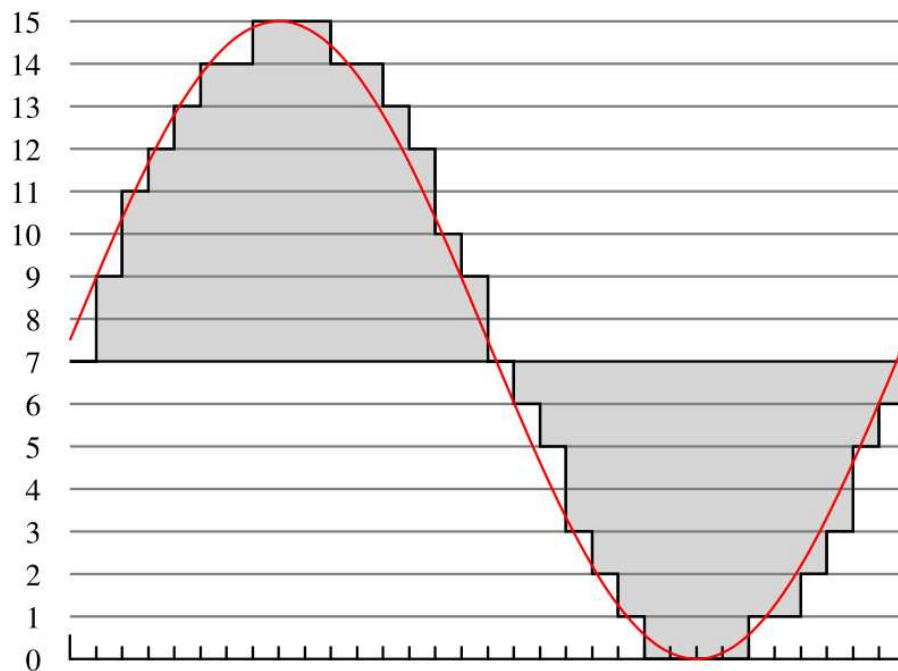


Figure 2.4: Digitalization of an analog signal using 4 bit quantization. Source: [71]

The amount of data in each sample (*bit resolution*) is the second parameter determining the quality of the digitalized audio. For instance, an audio CD contains audio data sampled at 44,100 Hz with 16 bit quantization and two channels (stereo). For one hour of music this results in  $44100 \cdot 2 \cdot 2 \cdot 3600 = 605$  MB of data. The term *bit rate* specifies the num-

ber of bits processed per unit of time. An audio CD has a bit rate of  $44100 \cdot 16 \approx 1411$  kbit/s.

### Digital Representation

Puls-Code-Modulation (PCM) is a standard method used for the digital representation of analog audio. Thereby, the amplitude data is sampled at uniform intervals and encoded into a stream of bytes. As uncompressed PCM has a high bit rate, often a compression technique called *companding* is used for speech encoding.

Companding uses PCM but logarithmic quantization instead of linear quantization. Logarithmic quantization uses a higher bit resolution for lower amplitudes than for higher amplitudes. Since speech amplitudes are rather low, the use of companding can reduce the necessary bit resolution with only small quality degradation. Therefore, it is often used in telephony.

The two companding algorithms *a-law* and  *$\mu$ -law* are standardized by the International Telecommunication Union Telecommunication Standardization Sector (ITU-T) as G.711. In Europe, a-law is employed, whereas  *$\mu$ -law* is used in USA and Japan. For encoding, both codecs use a sampling rate of 8000 Hz, and a logarithmic quantization of 8 bits. This results in a bit rate of 64 kbit/s and a rather high speech quality.

#### 2.1.2 Audio Compression

Since the largest frequencies audible by humans are in the range of 20 kHz, the sampling rate for high quality music must be at least twice as big (see Nyquist-Shannon-Theorem, Section 2.1.1). CDs, for instance, use a sampling rate of 44.100 kHz (see Figure 2.3). As described in Section 2.1.1, the resulting data rates are quite large ( $\approx 1411$  kbit/s) which imposes the following problems:

- The storage needed for large audio collections using audio with a bit rate of  $\approx 1411$  kbit/s is immense. Especially mobile devices often do not have the necessary resources.
- Networks often cannot handle this amount of data for streaming, especially if multiple users are using the network.

Therefore, compression techniques are used in order to reduce the amount of digital data. Depending on the application scenario, these techniques vary.

### **MPEG-1 Audio Layer 3 (MP3)**

The Moving Picture Experts Group (MPEG) has defined several lossy compression standards for video and audio. For MPEG-1, three audio layers exist. The most widely used layer is MPEG-1 Audio Layer 3 (MP3) which achieves very high compression. It provides an audio quality comparable to the quality of an audio CD at bit rates around 128 kbit/s to 192 kbit/s. For reaching such compression ratios compared to an audio CD (1:7 - 1:11), MP3 uses a *psychoacoustic model*. The psychoacoustic model makes use of the human perception of sound. Frequencies of a signal not audible for most listeners are removed which results in a great reduction of the amount of data. Today, MP3 is one of the mostly used audio formats. Applications include mobile players, Internet radio stations, music downloads and many more.

### **AMR**

The two codecs Adaptive-Multi-Rate Narrowband (AMR-NB) and Adaptive-Multi-Rate Wideband (AMR-WB) are originally designed for transmission of speech over circuit-switched mobile radio systems. AMR-NB (also known as AMR) was standardized by the European Telecommunications Standards Institute (ETSI) in 1999.

While AMR-NB is currently highly used in the Global System for Mobile Communications (GSM), AMR-WB has only recently been tested in a field trial in Hamburg and Cologne by the companies T-Mobile and Ericsson [66]. It provides higher voice quality and has been adopted as a standard by the ETSI/3rd Generation Partnership Project (3GPP) in December 2001 and by the ITU-T in January 2002 as G.722.2 for future usage in wide band networks such as the Universal Mobile Telecommunication System (UMTS). The specification of AMR-WB can be found in [5]. Applications for these codecs comprise digital telephony, audio- and videoconferencing, Multimedia Messaging Service (MMS), voice mail, ring tones and more.

Despite the originally intended purpose – circuit-switched mobile radio systems – AMR codecs can also be deployed expediently in packet-switched networks such as the Internet. Especially when they are used for transmitting voice over links with small bandwidths, one can benefit from their robustness and flexibility.

Unlike other audio codecs such as MP3, AMR features changing the bit rate for each frame during runtime of the encoding<sup>1</sup>. In Figure 2.5, the

---

<sup>1</sup>This should not be confused with the variable bit rate mode (VBR) of MP3. VBR is used for improving quality by utilizing more bits to complex segments and conversely allocating less bits for less complex segments.

mode	header (1 byte)	AMR-NB		AMR-WB	
		bit rate [kbit/s]	frame size (incl. header) [bytes]	bit rate [kbit/s]	frame size (incl. header) [bytes]
0	0x04	4.75	13	6.60	18
1	0x0C	5.15	14	8.85	24
2	0x14	5.90	16	12.65	33
3	0x1C	6.70	18	14.25	37
4	0x24	7.40	20	15.85	41
5	0x2C	7.95	21	18.25	47
6	0x34	10.20	27	19.85	51
7	0x3C	12.20	32	23.05	59
8	0x44	n/a	n/a	23.85	61

Figure 2.5: AMR bit rate modes. The AMR-NB specification includes 8 modes with different bit rates. For AMR-WB, 9 bit rate modes are specified.

different bit rate modes of AMR-NB and AMR-WB are illustrated. For each frame, a different bit rate mode can be chosen. This feature is especially interesting for this thesis as it enables the adaptation of the utilized bandwidth for audio.

AMR-NB uses a fixed sampling rate of 8000 Hz. Each encoded frame consists of 20 ms speech. Thus, a frame represents  $8000 \text{ samples/s} \cdot 0.02 \text{ s} = 160$  samples of original sound. The sampling rate of AMR-WB is fixed to 16000 Hz. Each encoded frame consists of 20 ms speech. Therefore, a frame represents  $16000 \text{ samples/s} \cdot 0.02 \text{ s} = 320$  samples of original sound.

AMR employs the speech encoding algorithm Algebraic Code Excited Linear Prediction (ACELP). For the purpose of bandwidth reduction during periods of silence, AMR supports Discontinuous Transmission (DTX) while utilizing Voice Activity Detection (VAD) and Comfort Noise Generation (CNG) [69].

### 2.1.3 Introduction to Digital Video

As discussed in Section 2.1.2, uncompressed digital audio occupies a large amount of storage. With uncompressed digital video, this problem becomes even worse. In video technology, the term *frame rate* specifies the number of frames per unit of time. The bit rate for an uncompressed video stream using 30 frames per second, a resolution of 640 x 480, 8 bit quantization and three distinct colors is  $640 \cdot 480 \cdot 8 \cdot 3 \cdot 30 \approx 221 \text{ Mbit/s}$ . Therefore, digital video must be compressed in order to be usable.

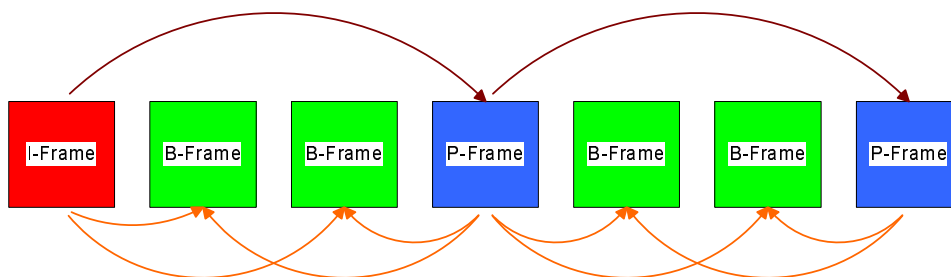


Figure 2.6: A group of pictures with related predictions.

In common video compression standards, video streams can contain three different types of pictures (frames)<sup>2</sup>:

- **I-frames (intra-coded frames)**: I-frames are coded independently of other frames. They have the lowest compression ratio as they do not use any temporal prediction.
- **P-frames (predictive-coded frames)**: P-frames may use predictions from preceding I-frames or P-frames, which typically results in a higher compression ratio.
- **B-frames (bidirectionally predictive-coded frames)**: B-frames may use predictions from the nearest preceding or subsequent I-frames or P-frames. They achieve the highest compression ratio.

Frames are arranged in so-called group of pictures (GOP). Figure 2.6 shows a typical GOP. As mentioned above, B-Frames may use predictions from subsequent frames. Since the inherent information of these subsequent frames is needed to decode the associated B-frame, they must be decoded first. Therefore, the compressed frames in the bit stream have an order (coding order), which diverges from the display order. The coding order for the GOP in Figure 2.6 is shown in Figure 2.7. More information on video encoding can be found in [41].

### H263/H263+

The ITU-T H263+ video coding standard (also known as H263-1998 or H263v2) was published in 1998 as an enhancement of the ITU-T H263 codec. It is especially designed for low bit rate videos with low motion and intended for videoconferencing applications. The H263+ version achieves higher video

---

<sup>2</sup>There exists another type of frame (D-frame), which is not often used and therefore not covered here.

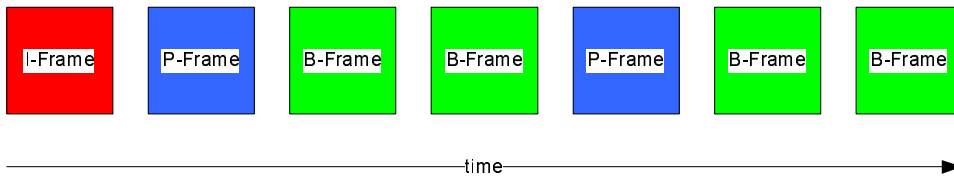


Figure 2.7: A group of pictures in coding order.

quality for lower bit rates than the H263 codec. Moreover, new negotiable modes are available in H.263+. The specification can be found in [26]. For transmission over RTP, which is introduced in Section 2.2.3, the corresponding Request for Comments (RFC) can be found in [14]. In this thesis, the H263+ codec will be used for demonstration of video adaptation.

### Scalable Video Coding

Scalable video coding is a technique which divides a video stream into several layers. The layers include a base layer and several enhancement layers. The enhancement layers can be added dynamically and provide temporal scalability (frame rate), spatial scalability (resolution) and Signal-to-noise ratio (SNR) scalability, which affects the quality of a frame.

With this technique, the rate-distortion relationship can be adapted without explicit recoding [30]. Examples are MPEG-2 scalability and the H264/MPEG-4 AVC codec.

Scalable video coding techniques can be used for reducing bandwidth while maintaining a graceful degradation of perceived quality.

However, it must be assured that the base layer is transmitted correctly. Therefore, retransmission for the base layer is often used in conjunction with adaptation techniques based on scalable video coding.

By means of several approaches that are presented in Section 3.1.2, the utilization of scalable video coding for sender-based and receiver-based adaptation is outlined.

## 2.2 Network Protocols

In the first part of this chapter, fundamentals of multimedia were presented. Now, several common network protocols are introduced. Furthermore, the differences and purposes of these protocols are discussed.

7. Application layer
6. Presentation layer
5. Session layer
4. Transport layer
3. Network layer
2. Data Link layer
1. Physical layer

Figure 2.8: The layers of the OSI reference model.

The Open Systems Interconnection Basic Reference Model (OSI Reference Model) [73] is an abstract layered model (see Figure 2.8). Every layer is build upon its lower layer and addresses a specific function of the network. It is used for reducing the inherent complexity of networks by the common approach of *divide and conquer*. In the following, three common protocols residing in the transport layer and their range of applications are described.

### 2.2.1 Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP) is the most common protocol used in today's Internet. The application range of TCP comprises the World Wide Web (WWW), e-mail, FTP and more. Since TCP is a complex protocol, this section can only give a limited overview. A more detailed description and examples can be found in [31].

During transmission, the protocol establishes a persistent connection between sending host and receiving host using a *three-way handshake*. The protocol provides reliable data transmission from sender to receiver and guarantees in-order delivery. For achieving this, it uses acknowledgments, timeouts, retransmission and sequence numbers as described in the following.

Since every packet contains a sequence number, lost packets can be recognized and the original sequence of packets can be restored. Using a checksum, TCP can also detect data corruption. For packets received successfully, an acknowledgment is sent back to the sender. If the acknowledgment is not received at the sender in a reasonable time (timeout at sending side), the packet is re-transmitted. In order to prevent duplicate packets from interfering with the data stream, they are discarded at receiving side. With these techniques, TCP accomplishes a reliable data transfer from sending host to receiving host.

Apart from reliability, TCP also provides congestion-control. The four congestion control algorithms, slow start, congestion avoidance, fast retransmit, and fast recovery are defined in RFC 2581 [48]. TCP uses a congestion

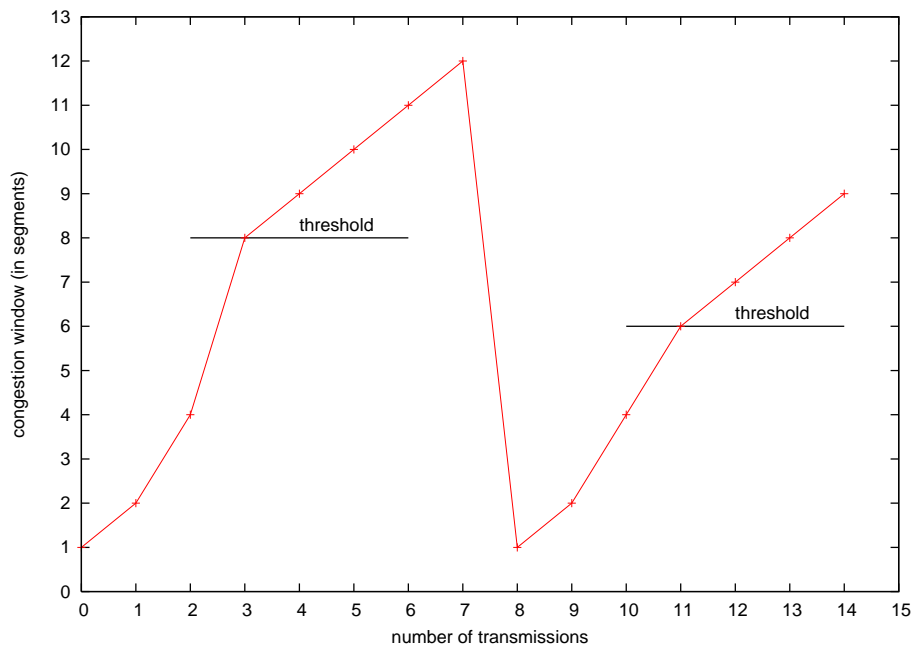


Figure 2.9: Progression of the TCP congestion window with idealized slow start. Disregarding the slow start phase of the TCP algorithm, the following observations can be made: The window size is increased by one per transmission if no congestion is present in the network. If acknowledgment deadlines are missed, the congestion window is reduced by a factor of two (in this case from 12 to 6). The prototype of the diagram is taken from [31].

window for determining the transmission rate. In the so-called *slow start* phase of the algorithm, the window size is increased by one for each packet that is acknowledged in time resulting in an exponential window growth. After reaching a threshold, the window size is only increased linearly. Then, the window size is increased by one if the number of acknowledged packets since the last change of the window size is equal to the current window size. As Figure 2.9 shows, this adjustment is essentially an Additive Increase Multiplicative Decrease (AIMD) algorithm if the slow start (exponential phase) is disregarded [31]. In the next section, the User Datagram Protocol (UDP) is introduced and compared to TCP.

### 2.2.2 User Datagram Protocol (UDP)

As TCP uses retransmission and congestion-control, it is not possible to transmit data with a fixed rate using TCP. UDP, which is defined in RFC 768 [50], provides this feature. In exchange, it does not provide any guarantees

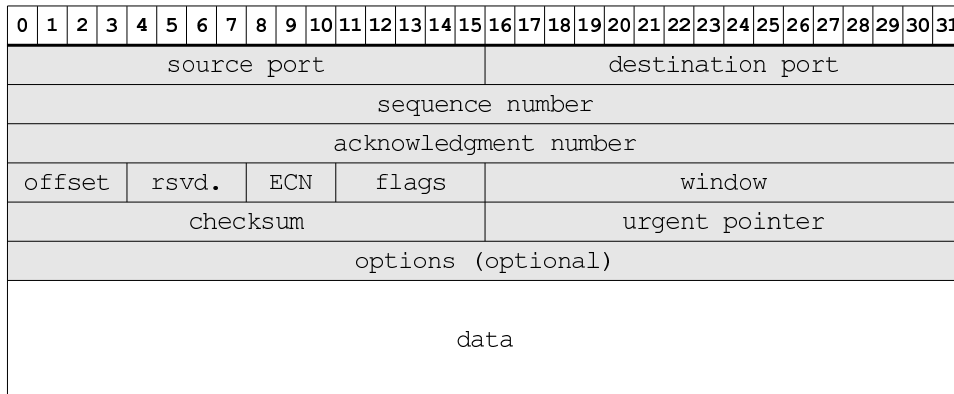


Figure 2.10: Structure of a TCP packet.

regarding timing and delivery. The data is simply sent from the sending host to the receiving host using a fixed rate, regardless of congestion or receiver capabilities. Since retransmission is never performed, packets lost on the path from sender to receiver will never arrive at the receiving side.

Although one might think that UDP is inferior to TCP in all areas, it has its right to exist. Since TCP uses rate control and congestion control, a fixed data rate, which is required for multimedia transmission, cannot be achieved. Moreover, for multimedia streaming applications the retransmission of packets is mostly useless. The reason for this is the time-constraints imposed by most multimedia streaming applications.

Usually, packets that contain multimedia data and that have been transmitted twice do not arrive at the receiver side in time. Moreover, the loss of packets in the Internet is often caused by congestion. Retransmission escalate the load on the path from sender to receiver and might cause further packet loss. Therefore, retransmission is most of the time not only useless but can also be harmful for the performance of a streaming application.

In some application scenarios, the reduced overhead is another reason why preferring UDP can be advantageous. UDP only uses an 8 bytes header (see Figure 2.11) compared to at least 20 bytes (see Figure 2.10) which are used for the TCP header. Therefore, using UDP, a better header to data ratio can be achieved.

Additionally, TCP can only be used for bidirectional connections between two computers (*unicast* connections). However, with UDP it is also possible to use *broadcasting* or *multicasting*. Using broadcasting, a packet is sent to all computers of the *broadcast domain*, which is usually restricted to the Local Area Network (LAN). Multicast uses special *multicast group addresses*. A host must register itself to such a group address in order to re-

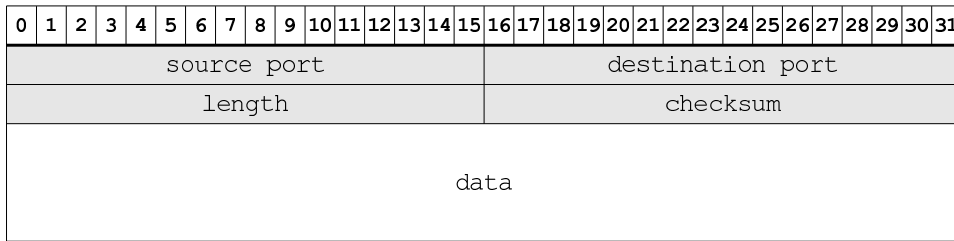


Figure 2.11: Structure of a UDP packet. A checksum for integrity check at the receiving side is included.

ceive packets. The advantage of multicast over unicast is a higher efficiency while transmitting information to multiple receivers.

### 2.2.3 Real-Time Transport Protocol (RTP)

RTP is a protocol for transmission of continuous audio and video data over network services. First standardized in 1996 [22], it was revised in 2003 [62]. Even though it is often implemented in user-level processes, the protocol is part of the transport layer of the OSI reference model (see Figure 2.8).

Designed to be independent of subjacent transport and network layers, RTP can be used with both TCP and UDP. However, on account of its intended purpose of streaming data in real-time, RTP is usually used with UDP, due to the advantages discussed in Section 2.2.2. Both unicast connections as well as highly scalable multicast connections are supported by RTP.

The protocol does not provide a QoS mechanism for real-time delivery as the name may suggest. Moreover, it does not guarantee in-time delivery and does not prevent out-of-order delivery, although sequence numbers may be used for reconstruction of the original sequence of packets.

For control of an RTP session, the Real-Time Transport Control Protocol (RTCP), which is described in the next section, is used. It provides media synchronization and a quality feedback mechanism which allow for adaptation of streaming parameters.

Figure 2.12 shows the header of an RTP packet. The fields are described in the following. For a comprehensive description see [62].

- **V (version)** [2 bits]: The version of the RTP specification used in this packet. The current version is 2.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
V=2		P	X	CC			M	PT							sequence number																
timestamp																															
synchronization source (SSRC) identifier																															
... contributing source (CSRC) identifiers (optional) ...																															
header extension (optional)																															

Figure 2.12: The RTP header

- **P (padding)** [1 bit]: If the padding bit is set, the payload section of the packet contains bytes that do not contain payload data. The last byte contains the number of padding bytes to be ignored. The use of padding can be necessary, for example, if an encryption algorithm is used, which requires a fixed block size.
- **X (extension)** [1 bit]: The extension bit indicates that exactly one extension header follows the RTP fixed header.
- **CC (CSRC count)** [4 bits]: The number of CSRC (contributing source) identifiers following the fixed RTP header. See the CSRC list description below.
- **M (marker bit)** [1 bit]: The marker bit is profile dependent and can be used for intra stream information such as indication of a frame boundary.
- **PT (payload type)** [7 bits]: This field indicates the payload type of the data carried in this RTP packet. The mapping of payload type to payload format may be specified by a profile. Default mappings are defined in [61].
- **sequence number** [16 bits]: The sequence number field can be used to identify packet loss and restore the original packet sequence. The initial sequence number should be random and is incremented for every RTP data packet sent.
- **timestamp** [32 bits] Alike the sequence number, the initial timestamp should be randomly chosen. The timestamp frequency depends on the payload type.
- **SSRC (synchronization source) identifier** [32 bits]: A randomly chosen value that identifies the source of this RTP-packet.

- **CSRC (contributing source) list** [up to 15 entries with 32 bits each]: An RTP mixer can be used to combine different RTP streams. For each packet of the combined stream, the synchronization source field will hold the mixers identifier. The original sources are specified in the CSRC (contributing source) list.

#### 2.2.4 Real-Time Transport Control Protocol (RTCP)

RTCP is not a stand-alone protocol but utilized in conjunction with RTP. Primary, RTCP is used for providing a quality feedback mechanism similar to flow control and congestion control in other protocols such as TCP.

As the name suggests, it is also used to control an RTP session. For example, it establishes a canonical name (CNAME) for each participant and carries information for inter-media synchronization in Network Time Protocol (NTP) timestamps and RTP timestamps. Here, only the quality feedback mechanism of RTCP is addressed in further detail.

In order to provide quality feedback of RTP connections, each participant of a session sends RTCP reports. A detailed description of RTCP reports follows below. To meet the requirement of scalability in a multicast scenario, the rate at which the RTCP reports are being sent is adapted to the number of participants. In the scope of this work, the quality feedback of RTCP is used for dynamical adaptation as a result of changing network conditions.

The RTCP specification includes 5 different RTCP packets. These are sender reports (SR), receiver reports (RR), source description items (SDES), packets that denotes the end of participation (BYE) and packets for experimental use of application-specific features (APP). Here, only the sender reports and receiver reports are discussed in further details.

The interval which is used for sending sender and receiver reports is not constant. Instead, it is depending on the number of participants of an RTP-session. Furthermore, it is randomly varied in order to avoid unintended synchronization of the participants. Further details can be found in [62].

##### Sender reports

Figure 2.13 shows the structure of an RTP sender report. Sender reports are periodically issued by participants that are active senders. A short description of the fields is given below. For further details see [62].

- **V (version)** [2 bits]: As for the RTP header (see Section 2.2.3), this field indicates the version of specification used in this packet. The current version is 2.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
<b>header</b>	V=2		P	RC				PT=SR=200				length																				
	SSRC of sender																															
<b>sender info</b>	NTP timestamp, most significant word																															
	NTP timestamp, least significant word																															
	RTP timestamp																															
	sender's packet count																															
	sender's octet count																															
<b>report block 1</b>	SSRC_1 (SSRC of first source)																															
	fraction lost								cumulative number of packets lost																							
	extended highest sequence number received																															
	interarrival jitter																															
	last SR (LSR)																															
	delay since last SR (DLSR)																															
<b>report block 2</b>	SSRC_2 (SSRC of second source)																															
	...																															
<b>optional extensions</b>	profile-specific extensions																															

Figure 2.13: Structure of an RTCP SR packet

- **P (padding)** [1 bit]: The padding bit is set for the same purpose as described for the RTP header in Section 2.2.3.
- **PT (packet type)** [8 bits]: Identifies the type of the packet. For a sender report this field is set to 200.
- **length** [16 bits]: Indicates the length of the RTCP packet (including header and possible padding).
- **SSRC (synchronization source identifier)** [32 bits]: A randomly chosen value that identifies the source of this RTCP packet.
- **NTP timestamp** [64 bits]: The time when this report was sent is indicated by this field.
- **RTP timestamp** [32 bits]: This field uses the same unit and offset as its pendant of an RTP packet. It corresponds to the same time as the NTP timestamp and can therefore be used for intra- and inter-media synchronization.
- **sender's packet count** [32 bits]: Specifies the total number of packets sent by this sender so far.
- **sender's octet count** [32 bits]: This field contains the total number of payload octets (without padding and header) sent by this sender up to the creation time of this report.

After the header and the sender info block, 0 to n blocks containing statistics follow. The actual number depends on the number of sources that the creator of this RTCP packet heart from since issuing the last report.

- **SSRC\_n** [32 bits]: The SSRC of the source to which the statistics of this report block belong to.
- **fraction lost** [8 bits]: Specifies the ratio of lost packets since sending the last report.
- **cumulative number of packets lost** [24 bits]: The number of packets lost since start of reception. Due to duplicated packets this value can be negative. Packets arriving too late are not counted as lost.
- **extended highest sequence number received** [32 bits]: This field is divided into 2 parts. The highest sequence number received in an RTP data packet is contained in the low 16 bits, the most significant bits indicate the respective sequence number cycles.
- **interarrival jitter** [32 bits]: Jitter commonly denotes the variance of delay times. In the RTP specification [62] it is defined as “...*the mean deviation (smoothed absolute value) of the difference  $D$  in packet spacing at the receiver compared to the sender for a pair of packets*”. According to the RTP specification  $D$  can be calculates as follows:

$$D(i, j) = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i)$$

where  $S_i$  and  $S_j$  are the RTP timestamps of packet i and j, respectively, and  $R_i$  and  $R_j$  are the arrival times (in RTP timestamp units) of packet i and j, respectively. The standard states that the jitter should be updated continuously when reports are received. When issuing a report, the current jitter value is sampled. The formula for computation to be used is as follows:

$$J(i) = J(i - 1) + (|D(i - 1, i)| - J(i - 1))/16$$

- **LSR (last SR timestamp)** [32 bits]: The NTP timestamp (in 32 bit precision) contained in the last RTCP sender report received.
- **DLSR (delay since last SR)** [32 bits]: Delay between the time when the last report was received and the time of sending this report.

With the LSR and DLSR the round trip time (RTT) can be calculated as follows:

$$RTT = A - LSR - DLSR$$

where  $A$  is the time when receiving the RTCP report. For further details on the calculation see [62].

For adaptation, the parameters *fraction lost*, *interarrival jitter* as well as the calculated *round trip time (RTT)* are considered to be valuable.

### Receiver Reports

Receiver reports carry the same information as sender reports except that the sender info block is missing and a different packet type (201 for RR packets) is set (see Figure 2.14). Receiver reports are issued by participants of the RTP session, which are not active senders.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
<b>header</b>	V=2		P	RC				PT=RR=201				length																				
	SSRC of sender																															
	SSRC_1 (SSRC of first source)																															
<b>report block 1</b>	fraction lost				cumulative number of packets lost																											
	extended highest sequence number received																															
	interarrival jitter																															
	last SR (LSR)																															
	delay since last SR (DLSR)																															
<b>report block 2</b>	SSRC_2 (SSRC of second source)																															
	...																															
<b>optional extensions</b>	profile-specific extensions																															

Figure 2.14: Structure of an RTCP RR packet

## Chapter 3

# Related Work

This chapter covers work related to the topic of this thesis. The first part gives an introduction to Quality of Service (QoS). In the second part, some existing tools which can be useful for development and evaluation of adaptation techniques are briefly introduced.

### 3.1 Quality of Service (QoS)

This section gives an overview of QoS. First, the term QoS is motivated and defined. Afterwards, techniques for achieving better QoS and related work are presented and discussed. Lastly, the presented work is summarized and aims of this thesis are introduced.

#### 3.1.1 Abstract Definition and Motivation

Depending on the purpose of an application, it has different requirements to the network. For instance, transmitting multimedia data requires fundamentally different characteristics than e-mail delivery. Timing for e-mail transmission is more or less unconstrained. In general, it does not matter whether an e-mail is delivered in 5 seconds or 10 minutes. In contrast, data integrity is very important as the content of e-mails must be preserved during transmission.

For multimedia streams, almost the opposite is the case. A small degree of lost frames or samples is tolerable. However, timing and available bandwidth are very important. If frames are reaching the receiver too late, they must be discarded. Therefore, network applications can be distinguished based on required bandwidth, time constraints and loss tolerance as illustrated in Figure 3.1.

	<b>bandwidth requirements</b>	<b>loss tolerant</b>	<b>time constraints</b>
<b>e-mail</b>	flexible	no	up to hours
<b>file transfer</b>	flexible	no	up to days
<b>interactive chat</b>	flexible	no	few seconds
<b>VoIP application</b>	medium	to some degree	milliseconds
<b>video conference</b>	high	to some degree	milliseconds

Figure 3.1: Constraints of different networking applications.

Since the Internet is a best-effort network, reaching timing and bandwidth constraints is much more difficult than reaching reliable transmission. The path a packet takes from a sending host to a receiving host is not predictable. Even packets of the same connection can use different routes. Moreover, delay can be introduced due to congestion and packets are routed without preference in today's Internet. Therefore, even packets from streams that are not time sensitive can generate congestion. If the network cannot meet the requirements of a multimedia application, it will often become unusable. In [25], QoS is defined as

“Collective effect of service performances which determine the degree of satisfaction of a user of a service.”

This definition is very abstract and not restricted to networks. In this thesis, however, QoS will always refer to the achieved quality of collaborating network components. Different approaches which aim for a higher QoS for network multimedia applications are introduced in the next section.

### 3.1.2 Approaches

For achievement of better QoS for multimedia transportation over networks, different techniques exist. In the following, several related approaches are outlined.

#### Prioritization and Reservation

The *IntServ (Integrated Services)* approach tries to reserve resources on per flow basis using the Resource Reservation Protocol (RSVP). *DiffServ (Differentiated services)* uses flags in packet headers to prioritize the flow of packets in routers based on these flags.

In [78], a framework called *On-Demand Path QoS framework (ODP)* is introduced, which uses scalability characteristics of Diffserv and end-to-end guarantees on per flow basis with less overhead than IntServ.

A concept of a QoS Framework called “The MATRIX” is presented in [57]. Instead of directly using technical data such as delay, bandwidth and jitter, these values are abstracted and mapped to “levels of quality”. The underlying middleware is then responsible for interpretation and mapping of these “QoS levels”. In the described architecture, a so-called *Resource Manager* is utilized for resource reservation.

All these approaches imply an infrastructure with routers which support the specific requirements of the RSVP or the packet based prioritization. De facto, this is not the case in today’s Internet and it is not expected to change in the future. Already one router on a transmission path without appropriate features is sufficient for a failure. Therefore, these approaches seem to be impractical for utilization in today’s Internet.

Apart from that, the success of the Internet is based upon the equality of participants. A privileged routing of packets includes high risks for the benefits of the Internet. The researcher Ben Teitelbaum from the project Internet2 states:

“The best-effort service model allowed the Internet to become the fast, cheap, and global infrastructure that we know and love. The temptation to teach it new tricks – like offering circuit-like QoS assurances – is very real. Unfortunately, there is a huge risk that in doing so, we would undermine the very properties that have made the Internet so successful.” [47]

Another approach for better QoS – which is considered in this thesis – is based upon adaptation. With this approach, properties of the connection are adjusted if network conditions change to allow for a better transmission result. In the following, related research is presented and discussed.

### Classification of Adaptation

In a video or audio stream, several properties can be adjusted. In [8], five main techniques that can be used for adaptation are classified:

- **temporal adaptation:** The adjustment of the frame rate for a video connection.
- **spatial adaptation:** The change of the video resolution.
- **quality adaptation:** The adjustment of the quality of video frames or audio samples.
- **format adaptation:** The change of a codec, for example switching to a codec with higher compression.

- **structural adaptation:** Modification of the stream content, for instance introducing error correction or disabling the video stream in a video conference. If the available bandwidth drops, this can be reasonable in order to maintain an acceptable quality of the audio stream.

Apart from the classified adaptation techniques, the changing of the utilized network technology is another possibility for adaptation.

[76] gives an overview of common approaches and different rate control techniques. In the source-based rate control, the adjustment is initiated by the sender. In this case, the adjustment can be based upon the estimation of the available bandwidth with probing techniques (*probe-based rate control*) or on a TCP throughput model (model-based or “TCP-friendly” rate control). The second approach, *receiver-based rate control*, is typically used with multicast scalable video. The video consists of several layers which correspond to different channels in the multicast tree. Based upon congestion recognition, the receiver can switch to higher or lower quality layers. The *hybrid rate control* combines source-based rate control and receiver-based rate control.

In the following, several approaches for adaptation will be presented and discussed. All approaches have the common goal to improve the perceived quality. They use a range of different techniques for estimating network conditions and several varying methods for adaptation.

### Adaptation Based on RTCP Feedback

In 1996, the first version of the Real-Time Transport Protocol (RTP), which is specifically designed for streaming of multimedia data, was standardized. It includes the Real-Time Transport Control Protocol (RTCP) which uses a feedback channel for transportation of information about the network condition to the sender. RTP and RTCP are described in detail in Section 2.2.3 and Section 2.2.4.

Several research projects have been conducted to use adaptation based on the feedback mechanism of RTCP. In [10], three network states depending on packet loss are distinguished: *Unloaded*, *Loaded* and *Congested*. Each state corresponds to a decision to either increase, hold or decrease the bandwidth.

In order to achieve “TCP friendly” adaptation, this early considerations are extended in [64]. An adaptation scheme called DAA (direct adjustment algorithm) is proposed, which uses the AIMD technique (compare Section 2.2.1) and combines it with an enhancement of the TCP-throughput model. The paper states, however, that the approach taken does not give

convincing results as many oscillations occur which are not suitable for multimedia transmission.

As RTP and RTCP are utilized by these proposals, they can be used with existing receivers that employ the RTP protocol. However, the adaptation is restricted to the information available in RTCP reports. Therefore, these approaches are not very flexible.

In [7], a framework for RTP streaming is described. Main components are the Media Device Managers (MDM), which are media dependent (one per type of media), and a QoSManager which takes QoS decisions. The adaptation decision is based upon packet loss and is similar to the approach taken in [10]. The packet loss is calculated based on RTCP feedback but extended with a profile (see Section 2.2.4). This profile is used to propagate additional information, namely packets arriving too late and packets not presented due to heavy load at the receiving side. For adaptation, four decision zones are used as illustrated in Figure 3.2.

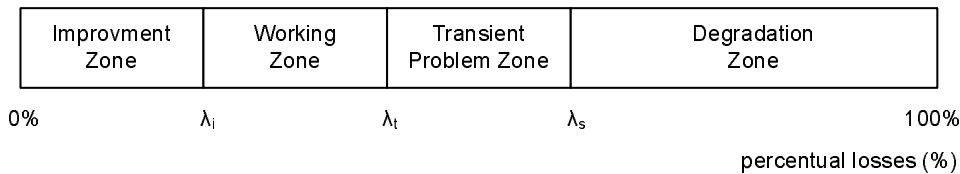


Figure 3.2: Decision zones depending on packet loss used in [7].

While the number of zones is statically used for all media, the values of  $\lambda_i$ ,  $\lambda_t$  and  $\lambda_s$  are media dependent and determined empirically. Similarly to the proposals already discussed, the framework presented in [7] can be used with standard RTP clients. However, a fine-grained and flexible adaptation is not possible as only four different zones can be used.

A combined approach of retransmission and postprocessing of video frames is taken in [17]. While using RTP/RTCP for transmission and feedback, the header is extended to allow for selective retransmission of I-frames. If the packet loss is too high for retransmission, a recovery postprocessing at receiver side is employed. The postprocessing is a temporal error concealment for I-frames by using preceding P-frames or B-frames for reconstruction.

Although the proposal employs RTP/RTCP, the use of header extension and postprocessing of I-Frames on the receiver side prevents that this approach can be utilized with existing RTP clients. Furthermore, the usage of adaptation for audio streams is not possible. Moreover, the usage of retransmission is considered to be precarious for multimedia content as discussed in Section 2.2.2.

The adaptive streaming architecture described in [4] consists of four main components. *MobileIPv6* [28] is used for transparent management of mobility. *Real-Time Streaming Protocol (RTSP)* [63] and *Session Description Protocol (SDP)* [23] are employed for management of sessions. By using *RTCP* for feedback information, a *fine-grained transcoding* at server-side is established. The focus of this work lies on heterogeneous networks and the support for adaptation during vertical handoffs. A vertical handoff is a change of the type of network connectivity such as switching from Wireless Local Area Network (WLAN) to UMTS. During setup and network handoff, a coarse-grained adaptation on client side can be performed by choosing a different stream which is called *session handoff*.

The presented architecture is interesting. The usage of *RTCP* feedback and standard protocols like *RTSP* and *SDP* enables the compatibility with existing clients. Moreover, the range of adaptation techniques is not limited to adaptation of specific codec parameters but allows for stream switching as well. However, the estimation of network condition is restricted to the information contained in *RTCP* reports which limits the adaptability.

### Adaptation Using Scalable Video Coding

As part of the project MediaNet, in [59] a system for adaptive H264/AVC streaming is presented. By creating different quality layers, a switching to lower data rates is possible. Additionally, temporal scaling can be used by discarding frames that are not referenced by others. Moreover, the presented system uses retransmission if packets are lost.

The streaming system presented in [44] is similar to the one of [59]. It combines several techniques for scalable video coding: spatial layers, temporal levels and fine grained scalability (FGS) quality levels of H264/AVC. The different combination of levels and layers are illustrated in Figure 3.3. For the congestion control algorithm, the probing technique called *Receiver-Based Packet Pair (RBPP)* is used for bandwidth estimation. Further details on techniques for bandwidth estimation can be found in [32]. Apart from the estimated bandwidth, packet loss and client buffer usage are considered. For feedback of packet loss, *RTCP* is used with a profile specific extension. Client buffer usage and bandwidth estimation are sent to the source through *RTSP* commands. Besides congestion control, retransmission is used for the base layer of the scalable video stream.

Both proposals (presented in [59] and [44]) use scalable video coding for sender-side adaptation. While only *RTCP* feedback is used in [59], receiver-based adaptation is employed in [44] as well. With the use of scalable video, a graceful degradation of perceived quality can be achieved. However, both approaches are very restricted as only the H264/AVC codec is supported.

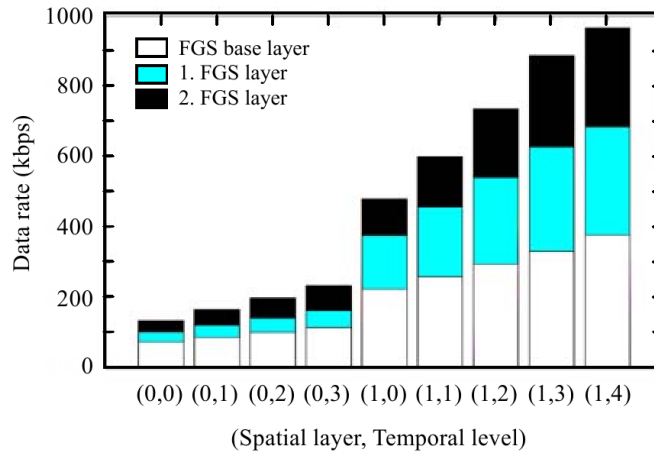


Figure 3.3: Different layers for scalable video coding used in [44].

In [79], a hybrid rate control approach is taken. It uses sender-based adaptation and a layered multicasting approach. The video stream is encoded into several video streams and is sent to different multicast groups. For decisions whether to join a multicast group with a higher or lower quality level, the RBPP method is used on the client side. A feedback channel is used by the sender for adjustment of streams. Figure 3.4 illustrates the architecture used in this approach.

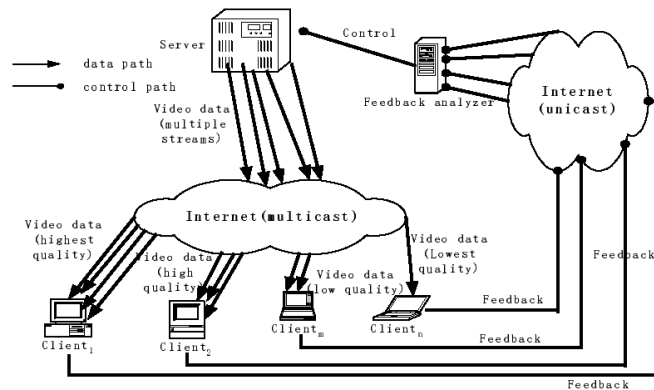


Figure 3.4: Scheme of the architecture used in [79].

Although no specific codec is mentioned, the approach is still restricted to video codecs that support a layered encoding. Furthermore, the proposal cannot be employed for audio adaptation. Additionally, routers capable of using multicast are required which yields in a further limitation of application.

### Adaptation Proposals for Wireless Scenarios

In [58], an adaptive multimedia framework for mobile and wireless applications is presented. Feedback is periodically sent via special signaling packets which include information about loss, delay, user preferences and estimated bandwidth. The relatively simple adaptation logic is illustrated in Figure 3.5.

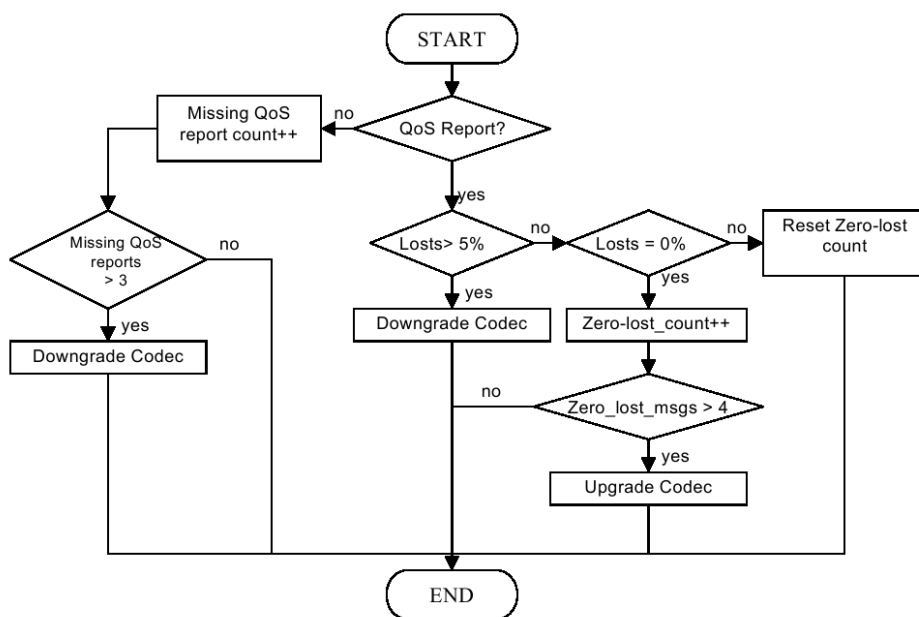


Figure 3.5: Scheme of the adaptation logic used in [58].

If more than three signaling packets are missed or loss is greater than five percent, the codec is “downgraded”. If no packet loss occurs in more than four consecutive messages, the codec is “upgraded”. Two testing scenarios were established. The first scenario is a handover from a 10 Mbit/s Local Area Network (LAN) to a 64 kbit/s Integrated Services Digital Network (ISDN). In the second scenario, a new participant is introduced into an ad hoc network environment with several participants in order to increase interference.

This adaptive framework is one of the few proposals that explicitly present the adaptation logic used. However, the changes made to the codec are only described abstractly as “Downgrade Codec” and “Upgrade Codec”. The general idea of the adaptation logic is considered to be interesting. However, in this approach, the logic is applied statically to all codecs. Therefore, it is not very flexible as the same simple approach considering a fixed per-

centage of packet loss and complete loss of RTCP reports is used for all codecs. Additionally, existing RTP clients can not be employed as the feedback is not realized with RTCP.

A gateway architecture with video adaptation is presented in [8]. It addresses a scenario with a network of access points that mobile clients use to connect to the Internet. It is assumed that support of seamless handoffs between different access points is possible. For adaptation enabled connections to the Internet, gateways must be used. The gateways act as RTP/RTSP proxies and are able to adapt a stream to the needs of the client.

The proposal depends on many assumptions. As described above, seamless handover between access points must be supported by the mobile devices. Additionally, multicast connections must be supported. Moreover, possible congestion on the paths from gateways to the streaming server is not considered. This requirements and assumptions yield to a very restricted approach which can only be applied in specific scenarios.

Another specific approach of H264/AVC streaming is presented in [3]. It is specifically designed for a wireless surveillance network. The fact that the difference between frames is often not very high in this scenario is exploited with the region of interest coding (ROI) of H264/AVC. Additionally, frame rate adaptation is utilized. If the difference between frames falls below a certain threshold, the frame is skipped resulting in a reduction of the frame rate. Figure 3.6 shows an overview of this approach.

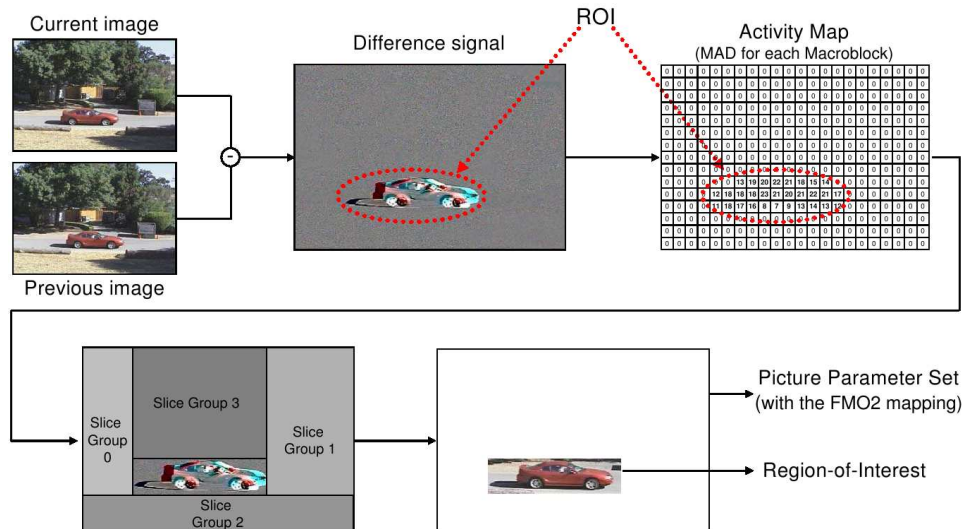


Figure 3.6: Region of interest coding by calculating the Mean Absolute Value (MAD) used in [3].

The proposal is liable to great restrictions on the range of applications. The technique presented is designed for a wireless surveillance network and only working if differences between consecutive frames are small. Moreover, the proposal is constrained as the only codec considered is H264/AVC.

### Perceived Quality

Another point which can be considered for adaptation is measurement of perceived quality. In [40], the relationship between quantization scale of the encoder, output bit rate and perceived quality is evaluated. Moreover, quality assessment schemes for video quality are discussed. The *subjective assessment*, where a group of test persons watches a video and rate it, is not suitable for real-time quality monitoring. Instead, *objective assessments* (measurements of perceptual quality) are used for feedback control channel and server-side adaptation.

The perceived quality of the IFD (I-Frame Delay) approach, is compared to the Signal-To-Noise Ratio (SNR) approach in [16]. With the IFD method, least important frames (B-frames) are dropped, whereas the SNR approach is based upon frame quality changes. Results from such surveys, for example the influence of the duration of a quality change on the perceived quality, can be used to refine adaptation methods.

### 3.1.3 Conclusion

The approaches described above are taken from a wide range of application scenarios. Several techniques for adaptation as well as strategies used for gathering information on network condition are outlined. Moreover, general concepts used for adaptation and perceptual quality are addressed. The different proposals indicate that there is no optimal adaptation technique.

However, the approaches presented are mostly restricted to a limited application scenario. Often, they are also constrained to a certain codec. In the scope of this thesis, however, a flexible architecture which is easily extendable should be designed and developed.

Hence, adaptation techniques must be versatile and not static. If the network condition can be estimated more accurately, adaptation methods can be applied more efficiently. Therefore, the possibility of considering as many parameters influencing the network as possible is required.

A *middleware* is a software layer residing between the operation system and the application. It eases the use of underlying components by making the access to them transparent. In the scope of this thesis, the use of a

network middleware, which has access to all network components, is therefore considered. It can be used to exploit as many aspects of the network as possible. The Network-Integrated Multimedia Middleware (NMM) is a middleware which features not only network transparency but also direct access to all components. It is introduced in Chapter 5 and used in the scope of this work.

As outlined above, prioritization and reservation are not applicable in today's Internet. However, they can be employed in local environments if the appropriate hardware is available. Moreover, they can be combined beneficially with adaptation techniques. While the adaptation techniques are utilized for Internet connections, local connections can be handled with prioritization or reservation.

## 3.2 Tools for Evaluation of Adaptation Techniques

A main issue while developing adaptation techniques is the unpredictability of large networks such as the Internet. However, for development and testing it is essential to operate in an environment that yields to reproducible results.

Often, in a small network of a laboratory environment the problems arising in the Internet cannot be addressed because mostly there is no heavy load and congestion. A tool like *IPerf* [24] can be employed for this purpose. Even though it is designed for measuring bandwidth, delay and packet loss, the utilization of network bandwidth can be generated for UDP connections by specifying a bandwidth for transmission. Nevertheless, the problems related to reproducibility remain.

In the related papers [12] and [44], which were presented in Section 3.1.2, a network simulation tool, *ns/ns-2*, is used for evaluation. *Ns* is a discrete event simulator for the purpose of scientific research. The applications include development of new network protocols, comparing different network protocols, estimation of network traffic and investigation of effects of QoS mechanisms.

In other approaches, [30] and [59], the network emulation tool *NIST Net* [11] is employed for evaluation purposes. *NIST Net* is a network emulator which is based on the IP layer and implemented for a Linux operating system. It is capable of emulating delay, bandwidth, packet loss and other parameters through a kernel module.

With the usage of a simulator, the visual or audible effects of network condition changes cannot be demonstrated. This is only possible with a network emulator. Therefore, *NIST Net* is used in this thesis for evaluation and demonstration purposes.



## Chapter 4

# Environment for Development and Evaluation

For development and evaluation of adaptation techniques, an environment with reproducible conditions must be established. As mentioned in Section 3.1.1, connection routes over the Internet are not predictable. Hence, network conditions cannot be reliably reproduced using Internet connections. For this reason, the network emulation tool NIST Net is utilized. NIST Net allows for reproducible testing and evaluation. The setup and usage of NIST Net is described in this chapter.

### 4.1 Setup

In the following, NIST Net will be installed on a computer which is configured as a router between two subnets. This is required for emulation. In the scope of this work, a simple case using three computers is utilized. The used computers are named *Client*, *Router* and *Server*, where *Router* is the computer on which NIST Net will be installed. The properties of the computers are as follows:

- Linux Debian system with kernel 2.6.18 (Server, Client)
- Linux Debian system with kernel 2.6.11.11k7 (Router) <sup>1</sup>
- CPU: AMD Athlon 64 X2 Dual Core Processor 4200+, 512 KB Cache
- RAM: 2 GB

---

<sup>1</sup>For more recent kernel versions the compilation of NIST Net failed due to a change in a kernel interface.

- two network interfaces with 1000 Mbit in each system (Server, Client)
- two network interfaces with 100 Mbit (Router)<sup>2</sup>

With the command `route` the routing table of a computer running Linux can be modified. The following two examples demonstrate the usage of the command `route`:

- `route add -net 192.168.3.0 netmask 255.255.255.0 dev eth1`  
This command specifies that network with the netmask 192.168.3.0 is reachable over the network interface `eth1`.
- `route add -net 192.168.3.0 netmask 255.255.255.0  
gw 192.168.4.1 dev eth1`

This command has the effect that the routing table is configured to use the computer with IP 192.168.4.1 as a gateway for reaching the subnet 192.168.3.0 over the network interface `eth1`.

Further details on setting routes in a Linux operation system can be found in the manual page of the command `route`.

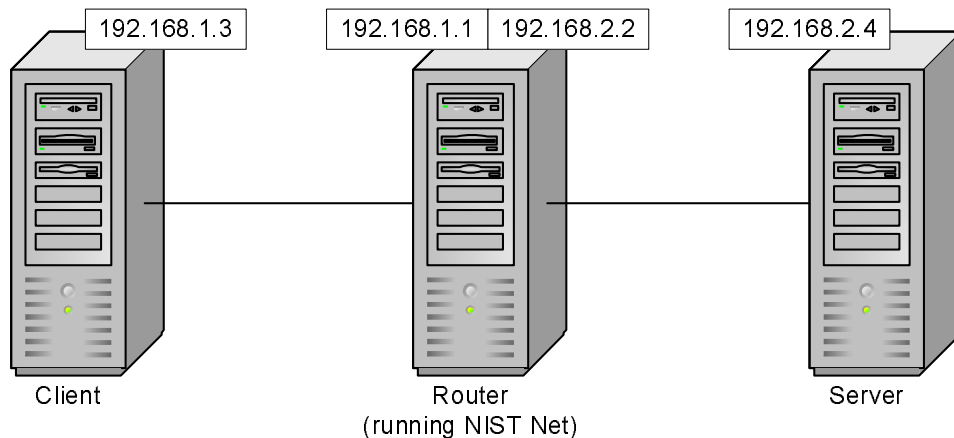


Figure 4.1: Schematic illustration of the setup used for development and evaluation.

The scenario used for this thesis is illustrated in Figure 4.1. The routing tables in the described setting are as follows:

---

<sup>2</sup>The 2.6.11 kernel does not support the recent Nvidia network cards of the system used. Therefore, two older network cards were installed.

- Client:

Kernel IP routing table (relevant extract)

Destination	Gateway	Genmask	Iface
192.168.1.0	0.0.0.0	255.255.255.0	eth1
192.168.2.0	192.168.1.1	255.255.255.0	eth1

The subnet 192.168.1.0 is reachable over interface `eth1` and the IP 192.168.1.1 is used as a gateway for reaching subnet 192.168.2.0.

- Server:

Kernel IP routing table (relevant extract)

Destination	Gateway	Genmask	Iface
192.168.2.0	0.0.0.0	255.255.255.0	eth1
192.168.1.0	192.168.2.2	255.255.255.0	eth1

The subnet 192.168.2.0 is reachable over interface `eth1`. The IP 192.168.2.2 is used as a gateway for reaching subnet 192.168.1.0.

- Router:

Kernel IP routing table (relevant extract)

Destination	Gateway	Genmask	Iface
192.168.2.0	0.0.0.0	255.255.255.0	eth1
192.168.1.0	0.0.0.0	255.255.255.0	eth0

The subnet 192.168.2.0 is reachable over interface `eth1` and the subnet 192.168.1.0 is reachable over interface `eth0`. To enable routing of packets from one subnet to the other, the forwarding of IP packets must be enabled. This is done by executing the following command:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

## 4.2 Installation of NIST Net

In this section, the necessary steps for installation of NIST Net are described. Previously, some issues and requirements that have to be considered before installation are addressed.

### 4.2.1 Prerequisites

NIST Net can be downloaded for free from the corresponding website [45]. The current version to the time of writing is nistnet-3.0a.

NIST Net supports setting network parameters based on ports. Unfortunately, this feature is not working in the version nistnet-3.0a. However, a patch was published on the official mailing-list accessible from [46]. If it is applied previously to installation, it solves the problem related to ports. Additionally, the installation of a package, *xaw3dg-dev*, and creation of symbolic links are required:

```
apt-get install xaw3dg-dev
ln -s <COMPILED_KERNEL_SOURCES_DIR> /usr/src/linux
ln -s /usr/src/linux /lib/modules/<KERNEL_VERSION>/source
ln -s /usr/src/linux /lib/modules/<KERNEL_VERSION>/build
```

### 4.2.2 Installation

The following steps are required to install NIST Net:

1. Configuration: After unpacking the source code, the configuration process is started by executing

```
./configure
```

During configuration, the following options can be chosen from:

```
Add explicit congestion notification (ECN) support?
Add class/type of service (COS) support?
Which delay distribution do you want to use?
    experimental normal pareto paretonormal?
```

Explicit Congestion Notification (ECN) is a technique of routers to indicate congestion while using TCP/IP. By setting a bit in the IP-header, a router can communicate impending congestion. More information can be found in the corresponding RFC 3168 [51]. Class of Service (COS) is a bit field for prioritization of packets. For the installation used, both options are enabled. However, they are irrelevant for the scope of this thesis and therefore not addressed any further. For packet delay, the experimental distribution is chosen for the used installation. It is created from measurements of statistical data and therefore appears to be the best distribution for reflecting the real world.

2. Compilation of the source code:

```
make
```

3. Installation of the binaries:

```
make install
```

4. Creation of an alias in /etc/modules:

```
alias <NIST_NET_SOURCE_PATH>/kernel/nistnet.ko nistnet
```

5. Updating modules:

```
update-modules
```

6. For loading the kernel module, a script is provided by NIST Net:

```
Load.Nistnet
```

## 4.3 Usage of NIST Net

This section describes the usage of NIST Net as well as the parameters which can be applied to emulate certain network conditions. For the usage, *root privileges* are required.

### 4.3.1 Controlling NIST Net

The NIST Net package includes three tools for controlling the kernel emulator:

- **Hitbox:** This is a deprecated command line tool and was not used for this thesis.
- **Cnistnet:** The new command line interface. The usage is described below, along with the specification of available parameters.
- **Xnistnet:** The graphical user interface (GUI) (see Figure 4.2). It is only used for displaying purposes as it is instable when parameters are edited.

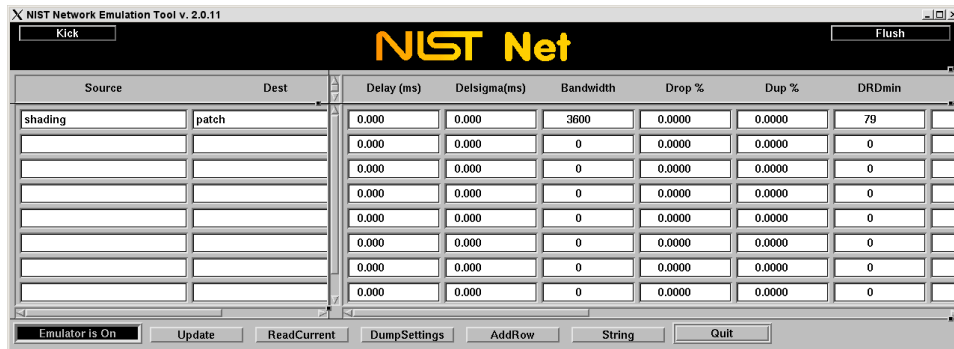


Figure 4.2: The graphical user interface of NIST Net.

### 4.3.2 Emulating Certain Network Conditions

NIST Net only affects incoming traffic. In order to apply parameters for a certain connection, source, destination and parameters must be specified as follows:

```
cnistnet -a src[:port[.protocol]]
          dest[:port[.protocol]]
          [cos]
          param
```

The IP or hostname of the source is specified with **src**; **dest** is the hostname or IP of the destination. The port and protocol may be specified as well as the class of service (cos). For controlling the behavior of NIST Net, several parameters can be specified (by **param**). They are as follows:

- **Delay**: The delay distribution was chosen during the configuration process. The parameter **delay** is used for the mean of this distribution, whereas **delsigma** is used for the standard deviation. The unit of both parameters is milliseconds (ms). Since delay times are often correlated, a correlation factor **delcorr** can be specified. Values in the range of -1 (totally anticorrelated) up to 1 (totally correlated) are valid. The value 0 corresponds to no correlation which is the default. A delay can be specified as follows:

```
--delay delay [delsigma[/delcorr]]
```

Example:

```
cnistnet -a 192.168.2.4 192.168.1.3 --delay 50 10
```

This rule adds a mean delay of 50 ms with normal deviation of 10 ms to the connection from the host with IP 192.168.2.4 to the host with IP 192.168.1.3

- **Dropped packets:** The percentage of dropped packets and the correlation can be specified as follows:

```
--drop drop_percentage[/drop_correlation]
```

Example:

```
cnistnet -a 192.168.2.4 192.168.1.3 --drop 20
```

If this rule is applied, 20 percent of packets sent from the host with IP 192.168.2.4 to the host with IP 192.168.1.3 are dropped.

- **Duplicated packets:** Similarly to dropped packets, duplicated packets are handled:

```
--dup dup_percentage[/dup_correlation]
```

- **Bandwidth:** The available bandwidth can be limited to the value specified by `bw` (in bytes per second) as follows:

```
--bandwidth bw
```

Example:

```
cnistnet -a 192.168.2.4 192.168.1.3 --bandwidth 10000
```

Applying this rule, the bandwidth between the specified hosts is limited to 10000 bytes per second.

- **Derivative random drop style parameters:** These parameters are used by NIST Net for simulating the router's queue and specified as follows:

```
--drd drdmin drdmax [drdcongest]
```

Example:

```
cnistnet -a 192.168.2.4 192.168.1.3 --drd 79 80
```

If the queue length is smaller than `drdmin` (here: 79), no packets are dropped. 95 percent of arriving packets are dropped if the queue is greater than `drdmax` (here: 80). If the queue length is in between these values, the probability that packets are dropped increases from 10 percent to 95 percent. In the scope of this work, the parameters are set to 79 and 80, respectively. They are adopted from a NIST Net distribution CD-ROM provided by Cisco Systems Inc [13], one of the biggest suppliers for networking technology, and therefore assumed to be realistic.

The parameters described above can be combined together to form a specific scenario for a certain connection.

### 4.3.3 Statistics, Settings and Debug Messages

For displaying information about statistics, the current settings and debugging, the following commands are available:

- Display current statistics for the given connection:  
`cnistnet -s src[:port[.prot]] dest[:port[.prot]] [cos]`
- Display statistics for the given connection continuously:  
`cnistnet -S src[:port[.prot]] dest[:port[.prot]] [cos]`
- Display of current settings:  
`cnistnet -R`
- Specification of verbosity used for debugging:  
`cnistnet -D value`
- Display global statistics:  
`cnistnet -G`

### 4.3.4 Miscellaneous

Other commands for control and usage provided by `cnistnet` are described below:

- Enable the kernel emulator:  
`cnistnet -u`

- Disable the kernel emulator:  
`cnistnet -d`
- Disable debugging:  
`cnistnet -U`
- Kick-start the clock:  
`cnistnet -K`
- Flush the queues:  
`cnistnet -F`
- Display the help and usage information:  
`cnistnet -h`



## Chapter 5

# NMM - General Overview

Nowadays, multimedia systems are part of almost every area of life. Especially mobile devices offer possibilities which become more and more important and challenging for the entertainment industry. Most often, these devices already provide network interfaces with the ability for interconnection. Anyhow, often the full resources of all participating components of a network cannot be exploited because there are no mechanisms for sharing and accessing them in a simple way. Therefore, a middleware is needed which overcomes these problems by integrating network and multimedia into a flexible architecture.

This chapter provides a short introduction to NMM, the *Network Integrated Multimedia Middleware*. The NMM project is developed at the Computer Graphics Chair of the Saarland University. It is distributed under a dual licensing scheme. Both an open-source version (General Public License (GPL) [21]/ Lesser General Public License (LGPL) [33]) as well as specific solutions distributed by Motama [42] are available.

The main goal of NMM is to provide a common access to distributed multimedia resources and at the same time maintain network transparency. This means that the same mechanisms used for accessing local resources can be used for accessing remote resources. In contrast to the streaming approach, a client has full control to all components. Hereby, underlying network specifications are hidden from the application designer who does not need any special knowledge about the network design. However, if necessary, the direct access to underlying network components is possible as NMM uses a *white-box design*. In contrast to other approaches like Corba [15] and the Distributed Component Object Model (DCOM) [70], NMM allows the configuration of utilized network protocols.

The approach NMM takes, enables a full range of new possibilities. For example, a mobile device without TV-card can be used for watching TV if

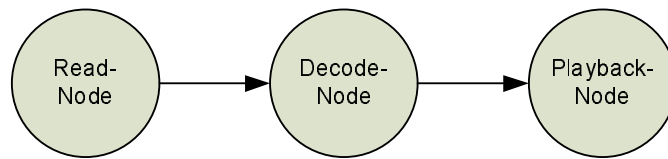


Figure 5.1: A simple NMM flow graph

another system in the network has this capability. Its TV-card is then used as a media source for the mobile device. Moreover, the computing power of yet another system can be exploited to encode the multimedia stream into a format which does not exceed the limited resources of the mobile device. The following sections provide an introduction to important parts of NMM. A comprehensive description of NMM can be found in [38].

## 5.1 Multimedia Flow Graph

The general structure of NMM is a multimedia flow graph. The flow graph is build up by interconnecting different plug-ins, which are called nodes. Each node has a limited functionality, for example reading audio data from a file or passing audio data to the sound card for playback. By combining nodes, the realization of many individual applications that perform different tasks is straightforward. For instance, a simple audio player can be created easily by connecting three nodes as illustrated in Figure 5.1. This architecture ensures a highly modular design which is easily extendable and provides a maximum flexibility and reusability of components.

As mentioned above, the network is an integral part of NMM. Therefore, using NMM, it is straightforward to write applications that have transparent access to the network including synchronization between different multimedia streams. By starting both nodes on different hosts, the simple audio player from above could easily be adapted to make location of the audio file and playback device independent from each other. The resulting flow graph is called a distributed flow graph. For example, this distributed player could be used by a device with limited storage resources. Then, without further configuration, it has access to another host with a whole collection of audio files.

## 5.2 Nodes

In NMM, basic components for multimedia processing are called nodes. They are responsible for processing multimedia data, for example converting

from one format to another or combining separate video and audio streams to one stream. In order to form a multimedia flow graph, nodes are interconnected.

### 5.2.1 Formats for Interconnection

In order to connect Nodes, NMM makes use of formats. A format consists of a classification (type and subtype) and a specification. An uncompressed audio format has the classification (`audio/raw`), for example. In this case, *audio* is the *type* and *raw* is the *subtype* of the format. Parameters such as sampling rate, number of channels and bit rate are used for further characterization of a format. They can also be weighted to enforce an order of prioritization on several possible intersection formats. The specification of a parameter is not fixed to one value. Several values for one parameter or a range of values can be specified. Moreover, a parameter can be specified as a *wildcard*. A wildcard parameter matches every possible entry of the parameter type. In [36], a short introduction to format programming in NMM is given.

Of course it is not possible to interconnect arbitrary nodes. Only if a so-called *intersection format* can be established, a connection of two nodes is possible. Every node has one or more properties that consist of a list of formats. The input property specifies all formats generally supported for input and is used for connection with a preceding node. Complementary, the output property specifies all generally supported output formats used for a connection with a successive node. Moreover, there exist associated properties, the working input property and the working output property, which are negotiated during connection of nodes. They only list the specific formats that are supported while two certain nodes are connected. More details on format negotiation can be found in [68] and [39].

### 5.2.2 Jacks

In order to combine different nodes, they need a common connection for sending and receiving data. These connections are called jacks. An input connector is called input-jack, an output connector is called output-jack. By connecting an output-jack of one node to an input jack of another node, the nodes are able to exchange data. The connection of two nodes is only possible if the format of an output-jack matches the format of the connected input-jack (see Section 5.2.1).

The exchange of data is not limited to one direction. Data can be sent downstream (from an output-jack to an input-jack) as well as upstream

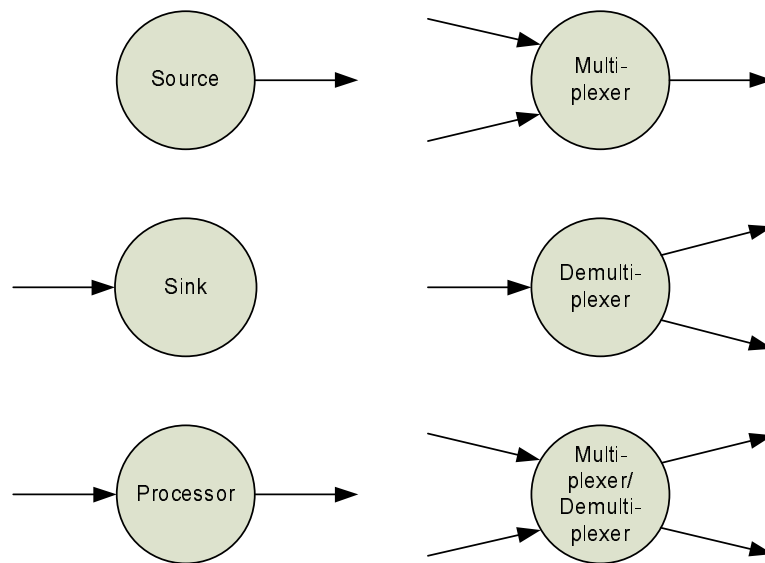


Figure 5.2: Types of nodes distinguished in NMM

(from an input-jack to an output-jack). A node can have several input- and output-jacks. Therefore, each jack has a name (jack-tag) to distinguish several input-jacks (or output-jacks) from each other. A jack has a queue with First In, First Out (FIFO) behavior which is used for buffering.

### 5.2.3 Types of Nodes

Figure 5.2 illustrates different types of nodes which are distinguished in NMM. The type of a node is dependent on the number of jacks and the formats used for input and output:

- **Source:** A node with one output-jack and no input-jack is called source. It is a node which feeds data into the multimedia graph. An example is a node which reads multimedia data from a file such as the `MP3ReadNode`.
- **Sink:** A node with one output-jack and no input-jack is called sink. It has no descendant in the multimedia flow graph. An example is a `PlaybackNode` or a node which writes data into a file.
- **Processor:** *Filter* and *converter* nodes are both *processor* nodes which have exactly one input and one output jack. The only difference between a *converter* and a *filter* node are the formats used for input and output. A *filter* node uses the same format for input and output,

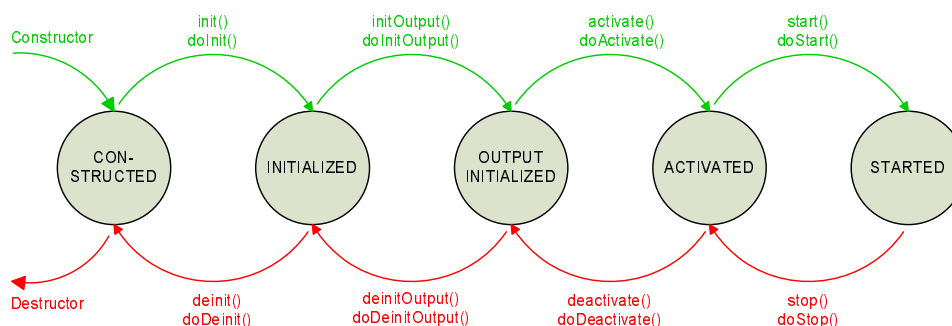


Figure 5.3: State machine of NMM.

whereas a converter node has different input and output formats. An example of a filter node is the `OverlayNode` which can be used to drop a specific ratio of packets. Examples of converter nodes for encoding and decoding are `FFmpegEncodeNode` and `FFmpegDecodeNode`.

- **Multiplexer** and **Demultiplexer**: A Node with more than one input jack and one output jack is called multiplexer. Complementary, a demultiplexer is a node with one input jack and more than one output jack. Common examples of use are multiplexing or demultiplexing of audio and video streams. A combination of Multiplexer and Demultiplexer, called *MultiplexerDemultiplexer*, is also available. It has several input jacks and output jacks.

#### 5.2.4 State of a Node

Before being able to process data, a node must undergo several steps such as initialization of data structures and libraries or negotiation of formats. Therefore, NMM introduces a state machine which controls the life-cycle of every node. This concept enables the enforcement of certain constraints before specific operations can be performed. Figure 5.3 illustrates the state machine. The states are as follows:

- **CONSTRUCTED**: After completion of the constructor, the node reaches this state. Variables influencing format negotiation should be initialized in the constructor.
- **INITIALIZED**: The state is reached after `init()` has finished. In this state, formats supported by the node have to be fully specified in the corresponding property. Moreover, resources used by the node must be allocated. For example, handles to external libraries or hardware interfaces are created.

- **OUTPUT\_INITIALIZED:** In this state, the working properties of the node are initialized. Available output formats in the working output property are restricted by formats chosen for each input jack.
- **ACTIVATED:** Upon calling `activate()`, all input and output formats are fully specified and the node is connected to its predecessor and successor.
- **STARTED:** The `start()` method enables the processing of data by the node. Internally, this starts the processing loop of the node. The main processing of a node is done in a method called `processBuffer`. Here, buffers from the input jacks can be handled according to the purpose of the node.

The transitions of states are implemented in the node base class. A node can perform specific operations during a state transition by implementing particular do-methods (see Figure 5.3). These do-methods are called internally by corresponding state transition methods. The method `doActivate()` is called, for example, by the method `activate()`. For reverting operations performed during state transitions, the methods `stop()`, `deactivate()`, `deinitOutput()`, and `deinit()` are available.

## 5.3 Distributed Communication

For interaction, nodes must exchanged messages between each other. Moreover, the application must be able to access and control nodes. This section describes the approach NMM takes in order to meet these requirements.

### 5.3.1 Types of Messages

NMM distinguishes two types of messages which are contained in a stream:

- **Buffers:** Buffers are used to exchange multimedia data between nodes. They contain the payload, i.e. video or audio data. For reconstruction of playback time, they also contain timestamps.
- **CompositeEvents:** A Composite-Event is composed of one or more events. Events do not contain multimedia data but are used for control. There exist two types of events: *out-of-band* events and *in-stream* events. Out-of-band events are created by the application and used for control of a node. An example is the specification of an input file for a node which reads data from a file. In-stream events, however,

are created by a node and send upstream or downstream. They are processed by all nodes passed. An example is the change of a format which can affect all successor nodes.

The support of user-defined events enables a high flexibility as arbitrary protocols for various purposes can be developed.

### 5.3.2 Transport-Strategies

A general paradigm of NMM is a design which is as generic and modular as possible. Therefore, the exchange of messages between distributed components is not bound to one transport protocol but to a `TransportStrategy`. A `Transport-Strategy` encapsulates an underlying transport protocol. At the moment, three different `Transport-Strategies` for networking (`TCPStrategy`, `UDPStrategy` and `RTPStrategy`) are available. They use the common transport protocols TCP, UDP and RTP, respectively. These protocols are introduced in Section 2.2. For local connections, no further processing is induced as only pointer forwarding is applied. Separate `Transport-Strategies` can be used for transmission of events and buffers. This concept is called parallel bindings [55], [56]. The `Transport-Strategies` participating at a connection between two nodes are enclosed in a `CommunicationChannel`.

### 5.3.3 RTP in NMM

Since RTP is a very important protocol for streaming of multimedia data over networks, it is also supported by NMM. For the integration of RTP, NMM deploys the library LIVE555 Streaming Media from Live Networks (LIVE555) [34].

As discussed earlier, NMM uses `CommunicationChannels` for representation of a communication link between two nodes. In order to be able to use RTP in a `CommunicationChannel`, a `TransportStrategy` for RTP was developed in [75]. The developed `TransportStrategy` is subdivided into three classes. The common functionality of a `TransportStrategy` using RTP is implemented in `RTPStrategy`. The classes `RTPSenderStrategy` and `RTPReceiverStrategy` are derived from `RTPStrategy` and used for sending and receiving in-stream messages via RTP, respectively. For the interconnection to LIVE555, two other classes were developed. The class `BufferedSource` is responsible for extracting payload data and timestamps from NMM-buffers and passing them to LIVE555. Accordingly, the class `BufferedSink` reads incoming data from LIVE555 and repacks it into NMM-buffers.

RTP is a standardized protocol which is supported by several multimedia applications. Therefore, a goal was, to be able to exchange NMM multimedia

streams with external applications. Since `TransportStrategies` are internal NMM components, they cannot be used for this purpose. Hence, two Nodes, one for receiving data from the Network and one for sending data to the Network, were developed in [49]. The Nodes are called `NetSourceNode` and `NetSinkNode`, respectively. Both are employed in the application developed in the scope of this work.

### 5.3.4 Serialization

For transmitting data through a network, it must be transformed into a sequential representation that can be sent across the network. This procedure is called *serialization*. For example, it can be used for sending C++ objects from one host to another one. The receiving host can use this object after reconversion (deserialization). NMM uses so-called `ValueStreams` for type safe serialization [52]. An extensible interface for serialization is offered by NMM. By using the base class `SerializedValue` and the generic class `TValue` the integration of new classes and types is very simple. An introduction to serialization in NMM can be found in [53].

### 5.3.5 Transparent Network Access

NMM provides a transparent access to the network. This enables application designers to easily use the given network without greater knowledge of the underlying infrastructure.

### Proxy Design Pattern

Like other middleware architectures such as CORBA [15] or Jini [27], NMM makes use of the proxy design pattern for transparent access to remote objects [52]. For every remote object, a proxy object is created which can be accessed in the same way as the local object. Events incoming to a proxy object are forwarded to the corresponding object for processing. It is also possible, to register event listeners at a proxy object which are notified when events are received. The advantage of using a proxy is the transparency gained. Accessing local and remote objects are handled in exactly the same way. How a connection to an object is established (for example use of a network connection) is completely hidden to the application.

### Interfaces

As described in Section 5.3.1, events can be used for control of nodes by an application. Apart from events, so-called *interfaces* can be used for

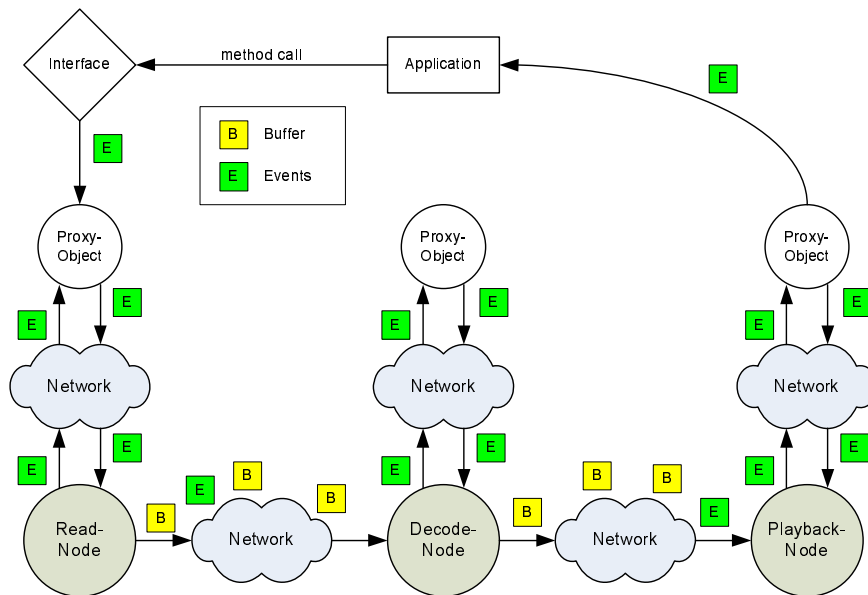


Figure 5.4: The Figure illustrates the generation of events by a call of an interface method and the distribution of buffers and events in the flow graph. The prototype of the Figure is taken from [49].

transparent control of NMM components. For creation of interfaces, NMM defines a special *Interface Definition Language (IDL)*. An interface is defined in an idl-file. The definition of interface methods is similar to the definition of C++ methods. From an idl-file, a special idl-compiler creates four files for use by C++. For an interface with name *IEncode* specified in *IEncode.idl*, the four files *IEncode.hpp*, *IEncode.cpp*, *IEncodeImpl.hpp* and *IEncodeImpl.cpp* are created. The class *IEncode* represents the local object and is used by a client. If a client invokes an interface method, internally an event is created and sent to the node as illustrated in Figure 5.4.

A node supporting the interface *IEncode* must inherit from the class *IEncodeImpl* and implement its abstract methods. More information on interfaces and the interface definition language can be found in [37].

The invocation of methods is less complicated than the generation of events. Therefore, interfaces eases the control of distributed components. Moreover, the generation of events is not type safe since the correctness of an event name cannot be checked by the compiler. Instead, using interfaces, type safety is achieved. Hence, mistakes can be found and corrected during compile time and not only at run time.

## 5.4 Middleware Services and Applications

This section briefly introduces to middleware services and applications available in NMM.

### 5.4.1 Device Management

The hosts connected in a network do not necessarily have the same abilities. Essentially, one of the features of NMM is to provide distributed access to components – such as a TV-card – that not every host exhibits. Therefore, a node which provides access to a TV-card (`TVCardReadNode`), is only available on a system which possesses a TV-card. For management of nodes available on a specific system, the so-called *ServerRegistry*, which was developed in [52], is responsible. On every system of a network that provides nodes, the application *serverregistry* must be running.

Locally, the access to distributed nodes is enabled through a *ClientRegistry* which can be used for requesting nodes from a Server-Registry. For this purpose, the class `NodeDescription` is used. It contains a description of all characteristics (such as name, type and supported formats) of a certain node. With the description, the node can be requested.

In an application, a flow graph is utilized. The request of a complete flow graph is simplified by using the class `GraphDescription`. A flow graph is constructed by adding Node-Descriptions in a desired order to a Graph-Description. Optionally, formats between nodes can be specified. The Graph-Description tries to interconnect these nodes by using the specified formats or default formats (if no format is specified for a connection).

### 5.4.2 Middleware Services

Since multimedia data is sent across the network, different clocks, and timing are present. Therefore, NMM provides *distributed synchronization* for example for a video and audio stream. Again, the usage is transparent for the user. The `GraphDescription` provides a method called `connectSinkSynchronizer()` which enables the synchronization of sinks.

Another feature of NMM is a technique called *Session Sharing*. It can be used to connect a graph to parts of existing graphs so that existing resources – like a TV-card – can be reused. Furthermore, it reduces overhead because repeated execution of processing steps can be avoided.

The application *clie* (command line interaction and configuration) can be used for rapid prototyping. It allows for setting up multimedia flow graphs

by simple textual descriptions. Further documentation of clic can be found in [54].

For automatic generation of flow graphs, the **GraphBuilder** can be used. Therefore, a Uniform Resource Locator (URL) which points to supported devices or file types must be specified. It is used for requesting appropriate source nodes and sink nodes. Additional options like the track number of a CD can be included in the URL as well. More details regarding automatic flow graph generation can be found in [38].



## Chapter 6

# Integration of AMR into NMM

In Section 2.1.2 the codecs AMR-NB and AMR-WB were introduced. This chapter covers the integration of both codecs into NMM. Moreover, details on related parts of the existing architecture of NMM are given. Additionally, two external libraries which are both used for the integration of AMR and for adaptation are introduced.

### 6.1 Encoding and Decoding of AMR Codecs

One principle of NMM is to reuse existing code. Therefore, NMM makes use of several open-source libraries. For encoding and decoding of video codecs the plug-ins `FFmpegEncodeNode` and `FFmpegDecodeNode` are used. They employ the library *libavcodec* from the FFmpeg project [18]. The library supports very fast encoding and decoding of many different codecs. It is well-known in the open-source community and used in many applications such as *MPlayer* [43] and *xine* [77]. Apart from the library, the FFmpeg project includes tools for transcoding, recording and streaming of multimedia data. Due to licensing issues [2], the *libavcodec* does not include the source code for AMR codecs. Therefore, it must be downloaded separately from the homepage of the 3GPP [1]. Subsequently, the library can be used for encoding and decoding of AMR as it implements AMR through the external reference code.

#### 6.1.1 Plug-ins for Encoding and Decoding

Due to the modular design of NMM, the integration of new codecs is a matter of developing new plug-ins. A good introduction covering plug-in

development in NMM is given in [35]. For decoding audio with the libavcodec, a general structure of a plug-in was already available. It is called `FFMpegAudioDecodeNode` and was developed to decode MP3 with the libavcodec. As the plug-in was not working, it is revised in the context of this thesis. Furthermore, it is extended to support decoding of AMR-NB and AMR-WB. Likewise, a corresponding plug-in for the encoding of MP3, AMR-NB and AMR-WB, the `FFMpegAudioEncodeNode`, is developed. As described in Section 2.1.2, the AMR codecs support different bit rate modes. At the time of development, the libavcodec did not support the change of AMR bit rate modes "on the fly". Therefore, a patch which enables support of this feature is developed for the library. The patch is included in the official version of the libavcodec since SVN revision 7140.

### Interface to libavcodec

In order to handle information related to FFMPEG, a private struct is used in both `FFMpegAudioEncodeNode` and `FFMpegAudioDecodeNode`:

```
struct FFMpegHandle {
    AVCodec* ffAVCodec;
    AVCodecContext* ffavcc;
    // only necessary for decoding MP3 audio
    AVCodecParserContext* parser;
    FFMpegHandle();
};
```

The FFMPEG struct `AVCodec` is used to characterize one codec. It includes information like name, ID, pointers to functions for initialization, encoding, decoding and closure. The struct `AVCodecContext` is used for specifying parameters that are common to different codecs such as bit rate, frame rate, frame size and many others. In order to use the libavcodec, the library must be initialized by calling `avcodec_init()` and registering all available codecs by calling `avcodec_register_all()`. The function `avcodec_find_encoder(CodecID)` returns a pointer to a struct `AVCodec` for the codec with the specified ID. For the `AVCodecContext`, memory must be allocated using `avcodec_alloc_context()`. Afterwards, the library can be opened with the function `avcodec_open(AVCodecContext*, AVCodec*)`. On closure the function `avcodec_flush_buffers(AVCodecContext*)` and `avcodec_close(AVCodecContext*)` must be called.

### Encoding

For encoding audio, the libavcodec provides the following function:

```
int avcodec_encode_audio(AVCodecContext *avctx,
                        uint8_t *buf,
                        int buf_size,
                        const short *samples);
```

The first parameter is a pointer to the `AVCodecContext` to be used. The next two parameters are a pointer to the buffer where the encoded data is stored and its size. The last argument is the pointer to the data that is to be encoded. The function returns the number of encoded bytes. Before using this function, enough data must be available for encoding. The amount needed can be calculated as follows:

$$\text{needed\_input\_bytes} = \text{channels} \cdot \text{byte\_per\_sample} \cdot \text{frame\_size}$$

The `frame_size` is available through the `AVCodecContext` after opening the library. In the `processBuffer()` method of the `FFmpegAudioEncodeNode` a temporary buffer is used. If the incoming buffer is not large enough, the encoding is postponed until enough data is available.

## Decoding

The decoding of audio is done with the following function:

```
int avcodec_decode_audio(AVCodecContext *avctx,
                        int16_t *samples,
                        int *frame_size_ptr,
                        uint8_t *buf,
                        int buf_size);
```

Its arguments are a pointer to the used `AVCodecContext`, a pointer to the buffer for the decoded data, a pointer to an `int` which will hold the actually used bytes of the buffer after execution, a pointer to the encoded frame and the frame size of the encoded frame. For decoding of MP3 audio, a parser must be used previously. Therefore, an `AVCodecParserContext` is to be initialized with `av_parser_init(CodecID)`. The following function is used to parse the input data:

```
int av_parser_parse(AVCodecParserContext *s,
                  AVCodecContext *avctx,
                  uint8_t **poutbuf,
                  int *poutbuf_size,
                  const uint8_t *buf,
                  int buf_size,
                  int64_t pts,
                  int64_t dts);
```

The function returns the number of decoded bytes. The parameters `s` and `avctx` are pointers to the `AVCodecParserContext` and `AVCodecContext`, respectively. After execution of `av_parser_parse()`, `poutbuf` will point to the parsed frame and `poutbuf_size` will hold the frame size of the parsed frame. The parameter `buf` is a pointer to the encoded data. The size of the encoded data is specified by `buf_size`. Finally, the last two parameters are the presentation timestamp and the decoding timestamp. In case of AMR, the parsing is not necessary.

Similarly to the temporary buffer required for encoding, also for the decoding a temporary buffer is used. It is necessary if not all data passed to the `processBuffer()` method in the `FFmpegAudioDecodeNode` could be decoded at once. The remaining bytes are copied to the temporary buffer and the *working-flag* is set. The working-flag ensures that the `processBuffer()` method is triggered again without passing a new buffer. The remaining bytes are then processed until not enough data for decoding is available, whereupon the working-flag is set to false.

### 6.1.2 A Format for AMR

Apart from developing plug-ins for encoding and decoding, another step is required in order to use the new codecs. As mentioned in Section 5.2.1, NMM uses formats to connect different nodes. Therefore, a new codec must be mapped to an internally used Format. For AMR-NB and AMR-WB the new Format type is called `Format::audio_amr_type`. To distinguish AMR-NB from AMR-WB, the parameter `Format::encoding_param` is set to "amr\_nb" and "amr\_wb", respectively. Further differences are the sampling rate (8000 Hz versus 16000 Hz) as well as the different supported bit rate modes as described in Section 2.1.2. They are reflected in the parameters of the format as well.

## 6.2 Streaming of AMR-NB and AMR-WB with RTP

In April 2007, the RFC 4867 was published. The document specifies an AMR-NB and AMR-WB file format as well as a payload type for RTP which is to be used for AMR-NB and AMR-WB. The payload format is compatible with existing AMR-NB and AMR-WB formats used on non-IP networks. As described in Section 2.2.3, RTP uses a number to identify payload types. Since AMR-NB and AMR-WB are two different codecs, two different payload types for streaming with RTP have to be specified in the range of dynamic payload types.

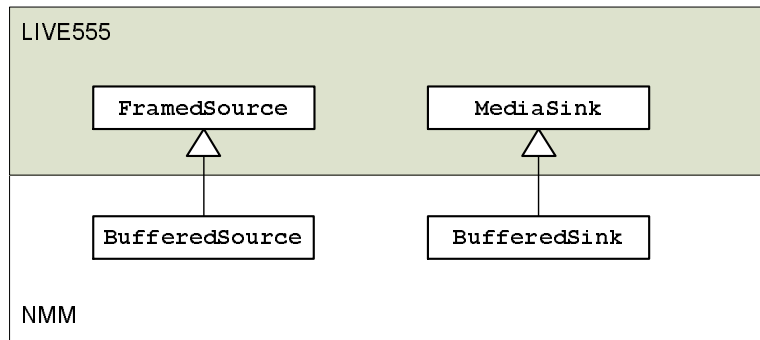


Figure 6.1: Class dependencies illustrating the connection of NMM to LIVE555

In Section 5.3.3 the general integration of RTP into NMM was discussed. Figure 6.1 illustrates the specific relationship of NMM classes and LIVE555 classes. The NMM classes `BufferedSource` and `BufferedSink` are derived from the LIVE555 classes `FramedSource` and `MediaSink`, respectively. The classes `FramedSource` and `MediaSink` provide virtual functions which are implemented in the derived classes to handle the incoming and outgoing buffers. For being able to stream AMR-NB and AMR-WB, modification to NMM components are required. They are described in the following.

### 6.2.1 Modifications in Class `BufferedSource` and Class `RTP-Strategy`

For streaming AMR-NB and AMR-WB the LIVE555 imposes several requirements. The derived class for sending (in this case `BufferedSource`) must be derived from the LIVE555 class `AMRAudioSource`. However, this class is already a specialization of a `FramedSource` which yields to the new class dependencies shown in Figure 6.2. Furthermore, AMR-NB is distinguished from AMR-WB using the flag `sourceIsWideband` in the constructor of the LIVE555 class `AMRAudioSource`. The flag has to be initialized in the constructor of the derived class. This design is considered to be inconsistent as this peculiar inheritance is not required for other formats.

Another issue has to be considered for streaming of AMR encoded by `libavcodec`. The 1-byte header, which specifies the bit rate mode (see 2.1.2), cannot be included in the AMR packet sent to LIVE555. Therefore, the header has to be removed and saved in the member `fLastFrameHeader` of the LIVE555 class `AMRAudioSource`. Since every `BufferedSource` is an `AMRAudioSource`, a mechanism to determine whether the header must be removed or not is required. For this purpose, a flag (`__used_for_amr`)

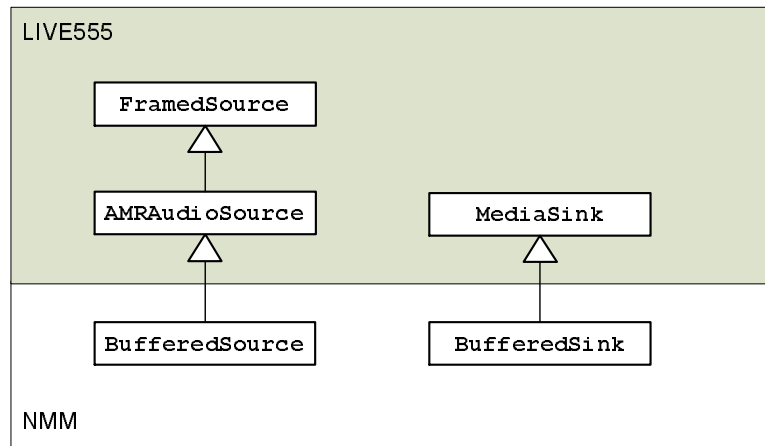


Figure 6.2: Class dependencies illustrating the connection of NMM to LIVE555 with modifications for AMR streaming

is included into `BufferedSource` which is initialized upon creation by the `RTPSenderStrategy`. The creation of a concrete `BufferedSource` is done in the method `RTPSenderStrategy::configure(const Format& format)`:

```

RTPSenderStrategy::configure(const Format& format) {

    string type = format.getFormatType();

    [...]

    if (type == Format::audio_amr_type) {

        string encoding =
            format.getStringValue(Format::encoding_param);

        if (encoding == "amr_nb") {
            /* Create an RTP sink for AMR_NB audio */
            __rtp_sink = AMRAudioRTPSink::createNew(env, rtp_sock,
                97, false);

            /* Create a buffered source */
            //wideband = false
            __buffered_source =
                BufferedSource::createNew(env, false);
            __buffered_source->setUsedForAMR(true);

            /* The source is the framer */

```

```
    __framer = __buffered_source;

} else if (encoding == "amr_wb") {
    /* Create an RTP sink for AMR_WB audio */
    __rtp_sink = AMRAudioRTPSink::createNew(env, rtp_sock,
                                           98, true);

    /* Create a buffered source */
    // wideband = true
    __buffered_source =
        BufferedSource::createNew(env, true);
    __buffered_source->setUsedForAMR(true);

    /* The source is the framer */
    __framer = __buffered_source;
}
}
[...]
}
```

### 6.2.2 Modifications in the Class BufferedSink and the Class RTPReceiverStrategy

Different from other formats, for AMR, LIVE555 distinguishes between a source for reading data from and a source for RTP related information. Therefore, an `AMRAudioSource*` member is introduced into the class `RTPReceiverStrategy`. In order to create consistent NMM-buffers with AMR data, the removal of the 1-byte header in `BufferedSource` is restored in `BufferedSink`.

### 6.2.3 Modifications in the Class RTPStrategy and the Class RTPPayloadtypeFormatMatching

The RTP payload types used in NMM are specified in the class `RTPStrategy` and the class `RTPPayloadtypeFormatMatching`. For the new codecs, corresponding payload types need to be added. Since AMR-NB and AMR-WB are codecs without static RTP payload type, they have to be chosen from the dynamic range (96-127). In NMM they are set to 97 (AMR-NB) and 98 (AMR-WB).

### 6.3 Summary

In this chapter, the integration of AMR-NB and AMR-WB was covered. As both codecs have different bit rate modes, they are ideally usable for adaptation techniques. For demonstration of adaptation techniques in the following of this thesis, the protocol RTP is used. Therefore, the streaming of AMR via RTP was described in more detail. In the next chapter, the developed QoS framework is introduced.

## Chapter 7

# A QoS Framework for NMM

The QoS framework, which is developed in the scope of this thesis, is based on the Network-Integrated Multimedia Middleware (NMM) which was introduced in Chapter 5. In Section 3.1.3, the advantages of using a middleware were already discussed.

In this chapter, the general design structure of the developed framework is presented. It is organized as follows. First, the goals of design and functionality are reviewed. Furthermore, application scenarios are outlined. Afterwards, the general concept is introduced followed by a detailed description of individual components. Lastly, the complete framework design is summarized.

### 7.1 Goals

In the following, the goals of the presented framework are described in detail:

- **Extendability:** The solution presented should be easily extendable. This includes adaptation techniques as well as general support for different network protocols including appropriate monitoring.
- **Flexibility:** As discussed in Section 3.1.2, several research has been done in the area of QoS. However, the aspect of flexibility regarding application scenarios and adaptation techniques is often neglected (see Section 3.1.2). Instead, the presented framework should be as flexible and heterogeneous as possible.
- **Modularity:** For extendability and flexibility, a modular design is essential. Future extensions are simplified and modular components can be exchanged and handled in a flexible way. Moreover, a modular approach eases the development.

- **Easy integration:** The usage of adaptation in existing and future applications should be simple. Therefore, design decisions must attend the aspect of integration by using abstract base classed for well-defined interfaces.

For reaching these goals, NMM is very suitable. NMM has been developed with specific regard to extendability, and many extensions have been developed over time. In Chapter 5, the modular design of NMM has been introduced. Therefore, integrating a modular framework is convenient. From its modularity, NMM also gains the ability to create and handle applications in a very flexible way. This is not restricted to multimedia formats but also applies to network protocols as introduced in Section 5.3.2. Moreover, the access to all components of a heterogeneous network is transparently possible by using interfaces (see Section 5.3.5). In Section 5.4.2 some middleware services of NMM were briefly introduced. Similar to the synchronization service, an adaptation service can be developed for easy integration.

## 7.2 Scenarios

In order to achieve flexibility, the use cases of the framework must be identified. Therefore, several scenarios are investigated. They are considered to cover a wide range of settings the presented framework should be able to handle. The scenarios are distinguished depending on the physical link used for communication. In wired network environments, the available bandwidth is the main issue for multimedia applications. Using wireless connections, not only bandwidth but also signal strength and interferences highly influence the overall transmission quality. The different scenarios are discussed in the following.

### Wired Network Connection

1. Suddenly, the available bandwidth drops heavily:  
For instance, this can happen due to a newly established connection that uses a large amount of bandwidth (for example a multimedia application using UDP).
2. Slowly, the available bandwidth declines and increases again:  
This can happen, for example, if a router is congested. The delay of packets increases as the router's queue fills up. After a while, some connections are disconnected. Hence, the available bandwidth increases and the congestion is resolved.

### Wireless Network Connection

3. The signal strength of a WLAN decreases continuously:  
If a receiving device is moved away from the access point, the signal strength normally decreases. This can affect adversely the available bandwidth or the packet loss ratio.
4. A WLAN connection is disconnected for a few seconds due to WLAN roaming. After the roaming has been completed, the performance of the connection can be much better than before. This case should be distinguished from a disconnection due to physical link loss such as a removed network cable. As the roaming is transparently managed by the access points, this is a difficult task. An idea would be to monitor available access points. If the signal strength of one access point was falling while the signal strength of another access point was rising previously to the disconnection, a roaming scenario is presumably.

An adaptation technique used for one scenario is not always optimal for another one. In scenario 1, for example, the amount of lost packets that are transmitted with an unreliable protocol such as UDP will increase. To restore an acceptable video quality, the amount of transmitted data must be reduced greatly. This can be achieved, for example, by reducing bit rate and frame rate of a video connection. In scenario 2, however, a reduction of the bit rate by a small amount might be sufficient. Moreover, it is desired to reverse the previously applied reduction when the available bandwidth is increasing again. For the last scenario, waiting might be the best strategy as often the performance of the connection improves after completion of roaming.

In other cases, it can happen that no further reduction of quality is possible with the current stream. Therefore, a reconfiguration of the flow graph and the utilization of a different codec can be required. In a worse case, the disabling of a stream might be unavoidable. In a videophone application, for example, a good voice quality is more important than the transmission of a video stream.

In the framework design, all scenarios are considered. In Chapter 8, adaptation techniques are developed and evaluation is performed for scenario 1 and 2. Evaluation and development of adaptation techniques for the other scenarios and reconfiguration of the flow graph is a task of future work.

### 7.3 Design Concepts

This section gives an overview of the general structure and components of the developed framework. Therefore, an application which uses Net-Sink-Nodes and Net-Source-Nodes (compare Section 5.3.3) for streaming of compressed video and audio data is considered in the following.

In Section 7.1, the benefits of a modular design were discussed. Therefore, the separate parts involved in the process of adaptation are determined. In order to perform adaptation, three main issues must be considered as illustrated in Figure 7.1.

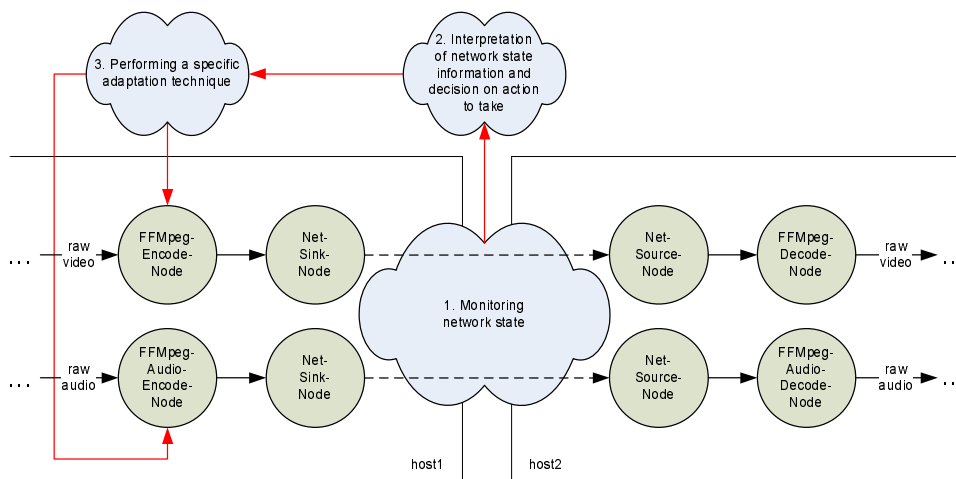


Figure 7.1: The figure shows an extract of a flow graph used for streaming of compressed audio and video. Three main steps that must be performed in order to achieve adaptation (in this example adaptation of video and audio bit rate) are identified.

Each point directly corresponds to a main component of the developed framework. These components are briefly introduced in the following.

For gathering information about the current network state, the class `QoSMonitor` is developed. Since the framework should handle several network connections, the monitoring is not done by one `QoS-Monitor` but by several ones. As several different network protocols may exist in a flow graph, the monitoring components must be able to work in parallel. Furthermore, different monitoring techniques, which depend on the transport protocol, must be applied.

As discussed in Section 5.3.2, NMM utilizes `Transport-Strategies` for encapsulating underlying transport protocols. Therefore, `Transport-Strategies`

are appropriate components for the integration of monitoring components. Transport-Strategies exist for sending and receiving. For each direction (downstream or upstream) and for each message type (Buffers or Composite-Events) a separate strategy may exist [38]. With this approach, sending-side and receiving-side can provide different data as every Transport-Strategy can create a specific QoS-Monitor for gathering information.

The interpretation of the information provided by QoS-Monitors and the determination of an appropriate reaction is a responsibility of the class **Adaptor**. For this purpose, the **Adaptor** must handle incoming data from several different monitors and determine which strategies for adaptation are used. Therefore, it must run in a separate thread and employs the *half sync/half a-sync pattern* for decoupling of asynchronous and synchronous processing [60]. Furthermore, it provides methods for determination of possible parameters that can be adapted in the utilized flow graph. In addition, the **Adaptor** serves as an interface between framework and application.

Instead of a static adaptation logic, a modular approach is taken for performing adaptation techniques. The base class **AdaptationStrategy** is developed for this purpose. By employing the *strategy design pattern* [19], specific adaptation techniques for various scenarios, codecs and transport protocols can be developed. Moreover, a flexible utilization of adaptation techniques is possible by combining different Adaptation-Strategies.

Apart from the three main components described above, another component for communication is required. In order to transport the information gathered by distributed QoS-Monitors to the Adaptor, it must be serialized. Therefore, a separate class called **QoSReport** is developed. For different transport protocols, specializations of the base class, which encapsulate the information about the current network state, can be developed. The class **QoSReport** and derived classes for RTP are described in more detail in Section 7.5.

Figure 7.2 illustrates the different components in a streaming scenario with server-based adaptation. By accessing Transport-Strategies, the QoS-Monitors gather information about the current network state. This information is sent to the Adaptor. The Adaptor decides on the appropriate reaction and instructs corresponding Adaptation-Strategies to perform an adaptation technique. The client running on host 2 is not necessarily using NMM (as illustrated in Figure 7.2) but could also be an arbitrary RTP client.

Figure 7.3 shows a similar NMM-application where compressed video and audio data is not streamed between two hosts, but instead transmitted within an NMM flow graph. In this case, information from the receiver-side could be used for adaptation as well.

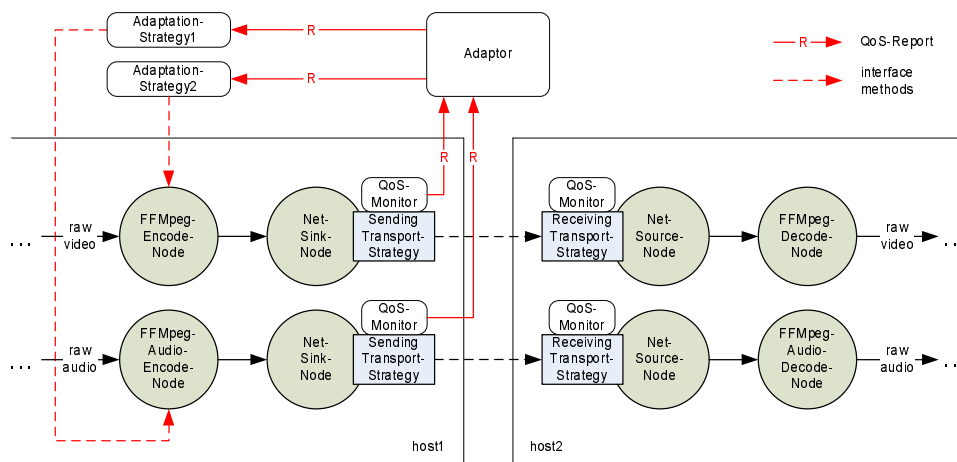


Figure 7.2: The figure shows an extract of a flow graph and the integration of the main components for adaptation. Different adaptation techniques (in this example sender-based adaptation of video and audio bit rate) are handled by different Adaptation-Strategies.

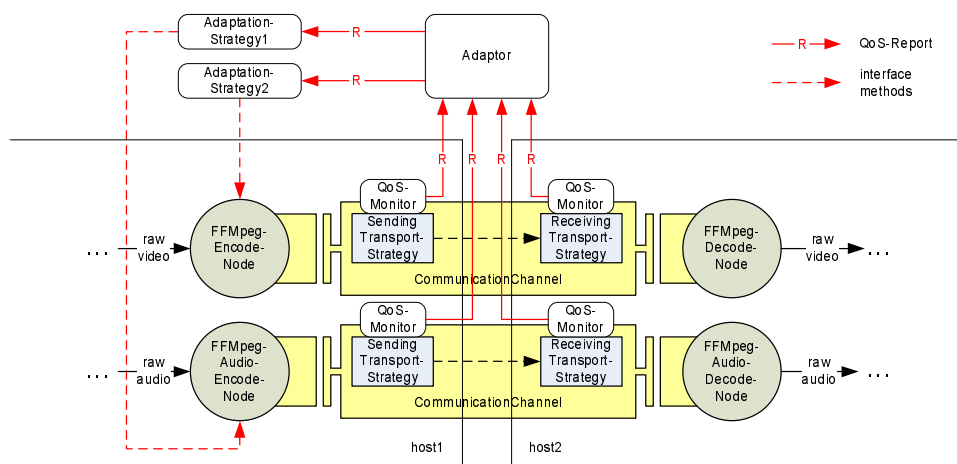
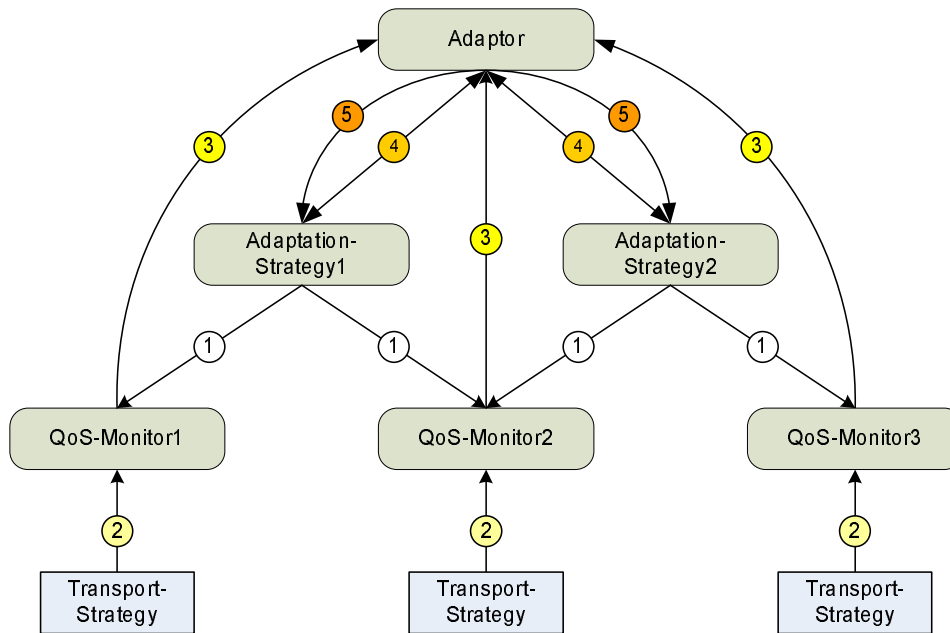


Figure 7.3: The figure shows an extract of a flow graph and the integration of the main components for adaptation. Instead of streaming, a NMM flow graph is employed for the transmission of compressed video and audio data from host 1 to host 2.

Although the second case is considered in the design of the framework, the development and evaluation is focused on the streaming approach.

## 7.4 Overall Communication

In this section, the communication flow between the main components of the developed framework is discussed in detail. Therefore, it is abstractly illustrated in Figure 7.4. First, Adaptation-Strategies are registered at desirable QoS-Monitors. Afterwards, network information is gathered by the QoS-Monitors and periodically sent to the Adaptor. The Adaptor interprets this data and communicates with the Adaptation-Strategies for this purpose. Finally, the Adaptor instructs Adaptation-Strategies to perform certain adaptation techniques. In Section 7.4.1 up to Section 7.4.5, the communication flow between the main components is consecutively discussed in detail.



- ① Registration of Adaptation-Strategies at QoS-Monitors
- ② Gathering data about the current network state (such as jitter, packet loss and delay)
- ③ Transmission of network state data to the Adaptor (by periodically sending QoS-Reports)
- ④ Determination of suitable adaptation techniques (by communication with Adaptation-Strategies)
- ⑤ Applying adaptation techniques

Figure 7.4: Connections between the main components of the developed framework.

### 7.4.1 Registration of Adaptation-Strategies at QoS-Monitors

The registration of an Adaptation-Strategy comprises three main steps:

1. Determine network connections (in the flow graph).
2. Determine nodes that allow for adaptation. To apply a certain adaptation technique, nodes may be required in the flow graph. For example, an encode node is required to modify the bit rate of a stream.
3. Register the Adaptation-Strategy at applicable QoS-Monitors.

#### 1. Determine Network Connections

The `Adaptor` is provided with access to the NMM flow graph because it acts as interface between application and framework. Therefore, the `Adaptor` can provide functions for examination of the NMM flow graph. For determination of Transport-Strategies that belong to network connections, four methods – for upstream, downstream, sending and receiving – are provided:

```
list< pair<ITransportStrategy*, INode*> >
    getDownstreamSenderNetworkTS();

list< pair<ITransportStrategy*, INode*> >
    getDownstreamReceiverNetworkTS();

list< pair<ITransportStrategy*, INode*> >
    getUpstreamSenderNetworkTS();

list< pair<ITransportStrategy*, INode*> >
    getUpstreamReceiverNetworkTS();
```

They return a list of interfaces to Transport-Strategies belonging to a network connection along with interfaces to corresponding nodes.

#### 2. Determine Nodes that Allow for Adaptation

An Adaptation-Strategy that cannot reconfigure the graph, requires nodes that can change a certain parameter of a format. Only if such a node exists, registration of this specific Adaptation-Strategy is reasonable. For example, a strategy for adaptation of the bit rate requires an encoder which is capable of changing the bit rate. In this case, the encoder must precede the QoS-Monitor the Adaptation-Strategy wants to register to. If no such encoder exists, it is not reasonable to register the strategy at that monitor.

Therefore, previous to registration, an Adaptation-Strategy searches for a node which is capable of changing desired parameters without changing the input format. The class `Adaptor` provides the generic functions `getAdaptableNodes(...)` and `getChangeableParameters()` which can be used for this purpose (compare Figure 7.5).

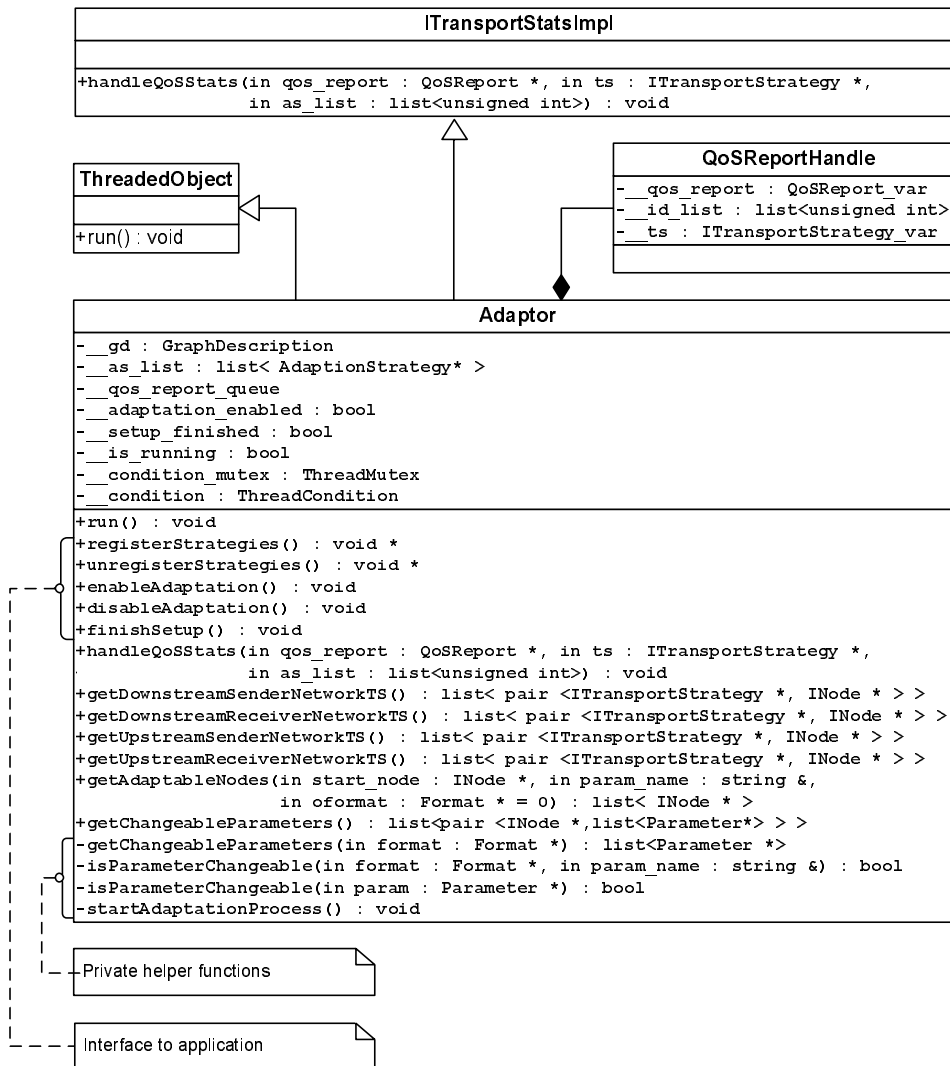


Figure 7.5: Class diagram of the class `Adaptor`.

Beginning at the `start_node`, the method `getAdaptableNodes(...)` traverses the graph in upstream direction searching for nodes with output format `oformat` that is capable of changing the parameter specified by `param_name` without changing the input format. If `oformat` is set to 0, which is the default, the output format of the node is arbitrary.



- The type of the parameter is `WILDCARD` which means that the parameter can hold arbitrary values.
- The type of the parameter is `R_INT32` or `R_FLOAT` which means that all values in a range of `int` values or `float` values are valid for this parameter.
- The parameter has several explicitly specified values.

### 3. Register the Adaptation-Strategy at applicable QoS-Monitors

In step three of the registration, an Adaptation-Strategy registers itself at the applicable QoS-Monitors determined in step two (see Figure 7.6). For this purpose, the interface `IQoSMonitor` is developed which is implemented by the class `QoSMonitor`. The interface is specified in the interface definition language (see Section 5.3.5) in an idl-file:

```
module NMM {
    interface IQoSMonitor {
        void registerAS(in unsigned int id,
                       in QoSReport * qos_report_filter);

        void unregisterAS(in unsigned int id);

        void setListener(in ITransportStats listener);
    };
}
```

For registration, an Adaptation-Strategy is identified by its unique id which is assigned by a static counter upon creation. Additionally, the registration can be restricted to certain reports only. Therefore, a filter can be specified in terms of a QoS-Report (`qos_report_filter`). This feature of the class `QoSReport` is explained in Section 7.5.

The recipient of QoS-Reports is specified with `setListener(...)`. As only one Adaptor is used in the scope of this thesis, it will be the recipient of reports for all QoS-Monitors.

Figure 7.7 illustrates the overall registration of an Adaptation-Strategy in a sequence diagram.

#### 7.4.2 Gathering Data About the Current Network State

For creation of new QoS-Reports, the class `QoSMonitor` offers the abstract function `handleReport()`. Specialized subclasses must implement this function which can be triggered when new network information is available.

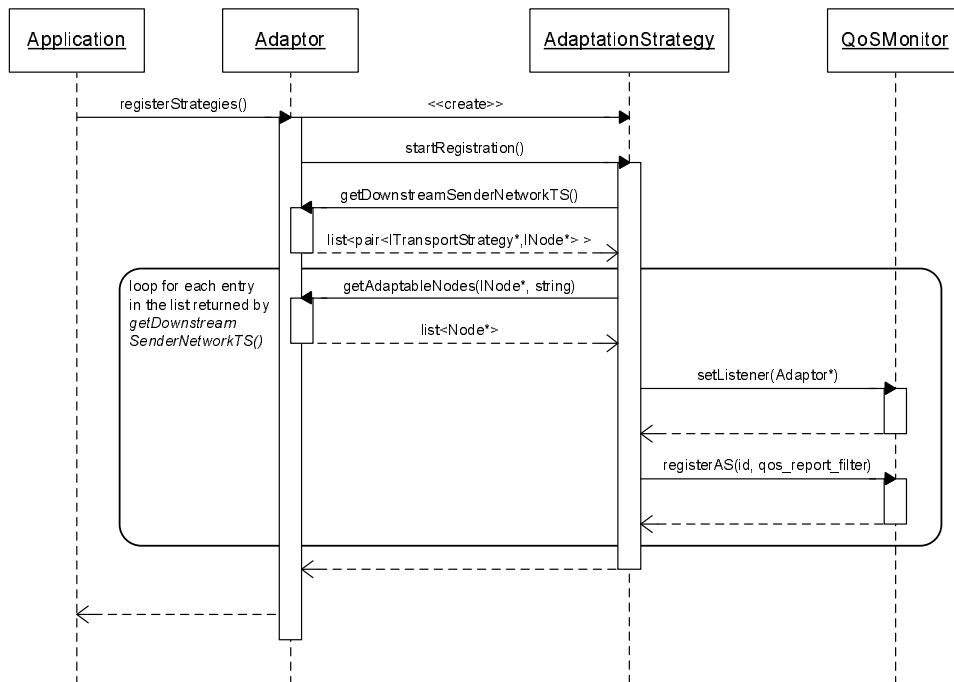


Figure 7.7: Sequence diagram: Registration of an Adaptation-Strategy at QoS-Monitors.

While the developed framework has a general purpose concept, in the following, sender-based adaptation using RTP/RTCP will be in the spot-light.

In Section 5.3.3, the integration of RTP into NMM was described. Furthermore, the distinction of Transport-Strategies for sending and receiving into the classes `RTPSenderStrategy` and `RTPReceiverStrategy` was addressed. In the following, the interface to the library LIVE555 for getting information from RTCP reports is introduced.

For information available on the sending side, the library class `RTPSink` must be accessed. The `RTPSink` has a member, `TransmissionStatsDB`, which is used by LIVE555 for processing incoming RTCP reports. By getting reference to this database, one can access information such as jitter and packet loss. This is done in the `RTPSenderStrategy`. It provides functions for accessing all data available through LIVE555. For example, the current round trip delay is returned by the function

```
Interval RTPSenderStrategy::getRoundTripDelay() const
```

Similarly, the `RTPReceiverStrategy` can access information on receiver side through the `RTPReceptionStatsDB` of the class `RTPSource`.

The LIVE555 class `RTCPInstance` has member functions for specifying functions that are triggered when a new sender- or receiver-report is available:

```
typedef void TaskFunc (void *clientData)
void setSRHandler (TaskFunc *handlerTask,void *clientData)
void setRRHandler (TaskFunc *handlerTask,void *clientData)
```

Figure 7.8 shows the class hierarchy of `QoSMonitor` and derived classes for RTP. The static member function `handleReportWrapper(...)` of the class `RTPQoSMonitor` is the callback function triggered by LIVE555 when a RTCP report is available. It will then execute the function `handleReport()` for creation of a QoS-Report. Since the callback function is required to be *static*, `handleReport()` cannot be specified directly as a callback function. For sender-based adaptation using RTP/RTCP, the actual implementation of `handleReport()` is contained in `RTPSenderQoSMonitor`. Likewise, the class `RTPReceiverQoSMonitor` can be used for receiver-based adaptation.

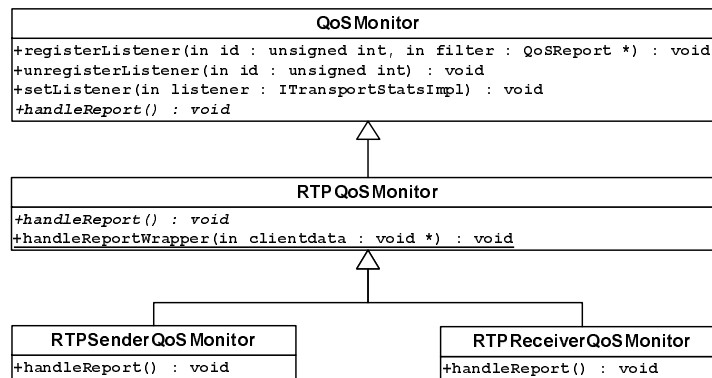


Figure 7.8: QoSMonitor and derived classes for RTP.

### 7.4.3 Transmission of Network State Data to the Adaptor

For communication between a QoS-Monitor and the Adaptor, the interface `ITransportStats`, which is implemented by the Adaptor, is used:

```
module NMM {
  interface ITransportStats {
    void handleQoSStats(in QoSReport * qos_report,
                      in ITransportStrategy* ts,
                      in list<unsigned int> as_list);
  };
}
```

The function `handleQoSStats(...)` does not only transport a QoS-Report but also a reference to the Transport-Strategy the report is belonging to. Moreover, it carries information about the Adaptation-Strategies that are registered for this type of report.

Using the information provided, the Adaptor is able to determine the source of the currently incoming QoS-Report. Moreover, it can forward the report to the registered Adaptation-Strategies.

In the implemented case using RTP/RTCP, the library LIVE555 triggers the related RTP-Sender-QoS-Monitor as soon as a new RTCP report is available. The accessible information is used by the QoS-Monitor to create a new QoS-Report which is sent to the Adaptor. This process is illustrated in Figure 7.9.

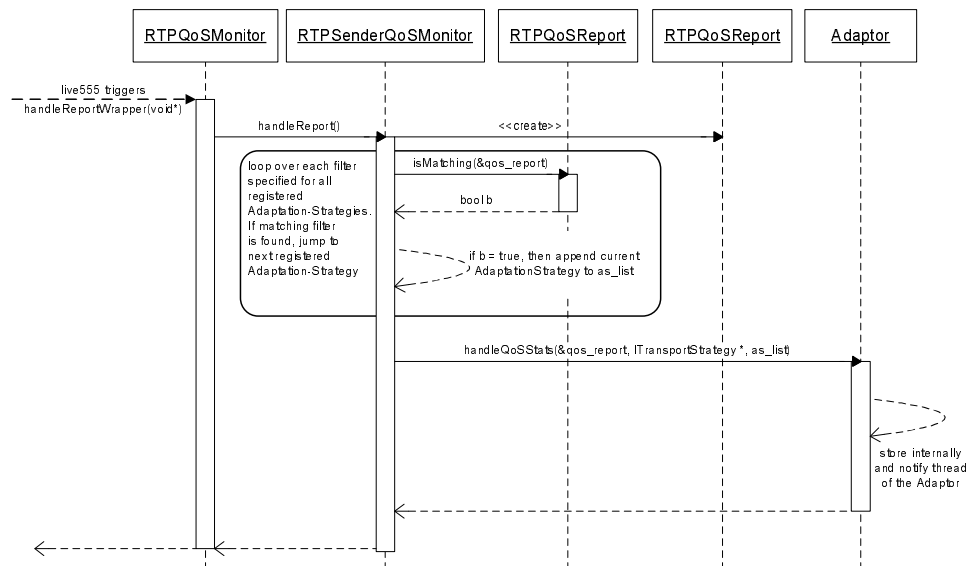


Figure 7.9: Sequence diagram: Handling of incoming RTCP reports.

#### 7.4.4 Determination of Suitable Adaptation Techniques

In Section 7.2 several scenarios were introduced. Adaptation-Strategies are designed to provide a modular resource of adaptation techniques. From the pool of available Adaptation-Strategies, the Adaptor must choose which strategies to use in a certain situation. For this purpose, the following issues are considered previously of adaptation:

- The Adaptor must be able to determine whether adaptation is reasonable in the current situation. In the fourth scenario, for example, adaptation is not meaningful as already discussed in Section 7.2.
- The Adaptor must be able to determine whether a certain Adaptation-Strategy is able to perform adaptation. Depending on the current situation, an Adaptation-Strategy might not be able to adapt. For instance, in scenario 1 or 2, a strategy for bit rate adaptation cannot decrease the bit rate any further if the minimum supported bit rate is already reached.
- The Adaptor must be able to determine how long it will take until the adaptation effect will be visible in received QoS-Reports. In almost every case, the effect imposed by adaptation is delayed. In this time, the QoS-Reports reaching the Adaptor should be handled without further adaptation.
- The Adaptor must be able to determine what the estimated result of an adaptation technique is. In scenario 1, for example, it can be reasonable to apply several Adaptation-Strategies partly or use more than one Adaptation-Strategy (e.g. bit rate and frame rate) at the same time. Hence, it is beneficial to know the effects of an adaptation technique previously to adaptation.

In order to meet these requirements, interface methods are provided by the class `AdaptationStrategy` as shown in Figure 7.10.

#### 7.4.5 Applying Adaptation Techniques

The actual adaptation is initiated by the method `startAdaptation(...)`. The QoS-Reports received from the QoS-Monitors are not necessarily transmitted to the Adaptation-Strategies. Instead, they might be “faked” in order to use several Adaptation-Strategies only partly. While currently appropriate interface methods are used for applying adaptation, in-stream events may be used to perform this task in future.

### 7.5 The Class QoSReport

As already mentioned in Section 7.3, the class `QoSReport` encapsulates information about the current network state. For RTCP statistics, the derived class `RTPQoSReport` is implemented. It contains all information available in an RTCP report (see Section 2.2.4). Additionally, calculated values like

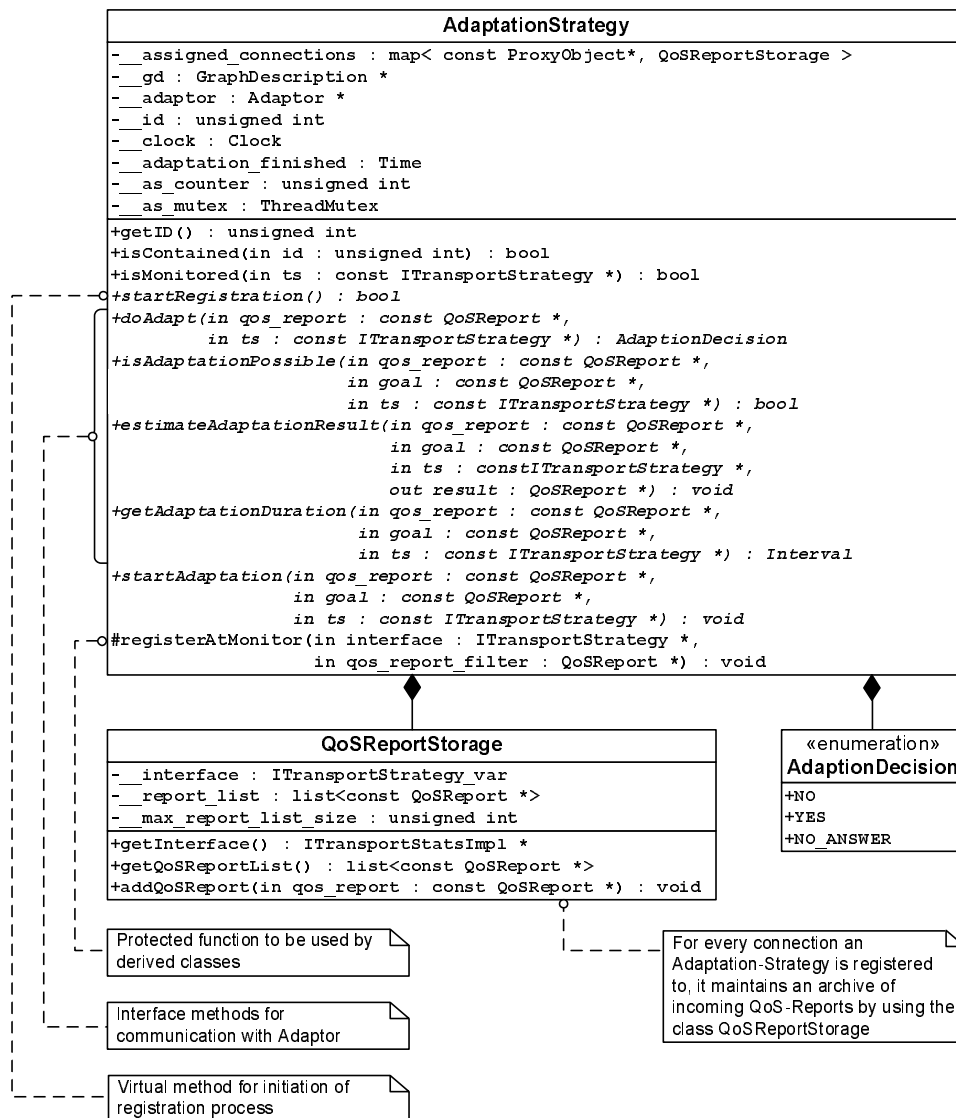


Figure 7.10: Class diagram of the class AdaptationStrategy.

round trip delay and information related to the current stream like payload type are included.

For each member, an additional member of type `QoSReportValueFlag` exists with a name according to the pattern `<entry_name>_flag`. Furthermore, there exist get- and set-methods for each member. Figure 7.11 shows the `QoSReport` class diagram.



Figure 7.11: Class diagram of the class QoSReport.

### 7.5.1 QoSReportValueFlag

The QoSReportValueFlag is used for the several purposes:

- **Filtering:** As already mentioned, filters can be specified during registration of Adaptation-Strategies. For example, while using a specific adaptation method, only reports with a certain value that is greater than a threshold are interesting. For instance, a QoS-Report with entry *packet\_loss\_ratio* set to 0.03 and corresponding flag set to **GREATER** can be used for filtering reports for which *packet\_loss\_ratio* > 3. The function `isMatching()` is used for determination whether a filter applies or not. In order to specify more than one filter for an Adaptation-Strategy, the method `registerStrategy(...)` of the `IQoSMonitor` interface (see Section 7.4.1) can be invoked several times for the same Adaptation-Strategy.

- **Specifying a goal for adaptation:** For example, the goal might be to reduce the currently imposed delay of the connection. This can be achieved by setting the flag corresponding to the delay value to `FALLING`.
- **Specifying the precision of an estimation:** For example, an `Adaptation-Strategy` might expect a decreasing packet loss ratio but is not able to specify an exact value. In this case, it can set the corresponding flag to `FALLING`.

In order to enable combination by the binary OR-operator, the constants of the enum `QoSReportValueFlag` are specified as a power of two (see Figure 7.11).

### 7.5.2 Serialization of QoSReport

For processing of QoS-Reports, they must be sent across network connections. Therefore, the base class inherits from the class `SerializedValue` and entries are encapsulated into the template `TValue` (see Section 5.3.4).

Furthermore, operators for serialization and deserialization are explicitly implemented for the enum `QoSReportValueFlag`. They have the following signature:

```
OValueStream& operator<<
(OValueStream& ov, const QoSReport::QoSReportValueFlag& t)

(IValueStream& iv, QoSReport::QoSReportValueFlag& t)
IValueStream& operator>>
```

## 7.6 The Class `CompositeAdaptationStrategy`

In some cases, it can be expedient to combine the utilization of several `Adaptation-Strategies`. For example, the combination of frame rate and bit rate adaptation can be reasonable (see Section 8.6).

For this purpose, the *composite pattern* [19] is exploited by developing the class `CompositeAdaptationStrategy` which is derived from the class `AdaptationStrategy`. The strategies that should be combined are stored in a list. For addition and removal, the class `CompositeAdaptationStrategy` provides appropriate methods as shown in Figure 7.12.

As outlined in Section 7.4.1, an `Adaptation-Strategy` is registered at a `QoS-Monitor` using its ID. However, a `Composite-Adaptation-Strategy` does

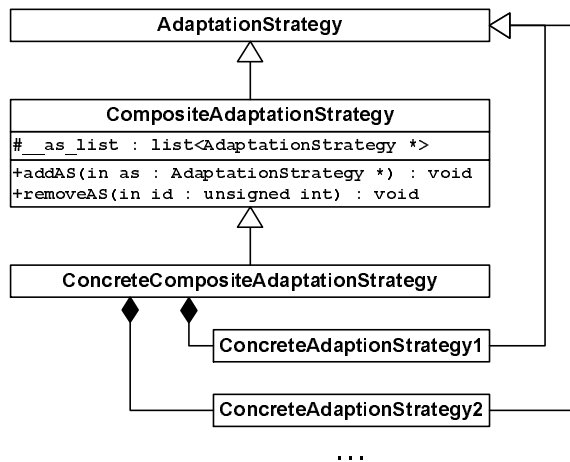


Figure 7.12: Class diagram of the class `CompositeAdaptationStrategy`.

not register itself but the Adaptation-Strategies (identified by their IDs) it contains. Hence, the Adaptor will never receive QoS-Reports associated to the Composite-Adaptation-Strategy. Nevertheless, the intended recipient is the Composite-Adaptation-Strategy that contains the Adaptation-Strategies the reports are associated with.

In order to solve this problem, the class `AdaptationStrategy` offers the method `isContained(unsigned int id)` (see Figure 7.10). In the implementation used for “conventional” Adaptation-Strategies, the function returns `true` only if the specified ID is equal to the one of the strategy. For a Composite-Adaptation-Strategy, the method returns `true` if an Adaptation-Strategy with the specified ID is contained. Hence, the function can be used by the Adaptor to determine the intended recipient.

## 7.7 Summary

In the last sections, the components of the developed framework and their interaction were presented. This section summarized the design and concepts of the developed framework.

A general overview of participating classes and their relationship is illustrated in Figure 7.13. Using *modular components* (`AdaptationStrategy` and `QoSMonitor`) the presented framework achieves a *flexible structure* which is *easily extendable* by developing further Adaptation-Strategies.

Therefore, the different scenarios presented in Section 7.2 can be handled appropriately. In Chapter 8, Adaptation-Strategies for scenario 1 and



## Chapter 8

# Strategies for Adaptation and Evaluation

This chapter presents several Adaptation-Strategies developed in the scope of this work. At first, two Strategies which can not be used for adaptation are introduced. They are dedicated for statistical purposes and evaluation. Afterwards, a strategy for audio adaptation and two strategies for video adaptation are presented. The last presented strategy combines the two strategies for video adaptation in a Composite-Adaptation-Strategy.

### 8.1 Strategies for Evaluation Purposes

The Statistic-Adaptation-Strategy can be used for storage of incoming QoS-Reports of an audio and a video stream. For each stream, two files are saved that contain a different format for representation. NMM features a `MagicNumberOValueStream` which can be used for writing the state of a C++ object to a file. Accordingly, a file can be used to restore a previously saved object state. The first file of a stream is written by using such a `MagicNumberOValueStream`. In this case, the reports saved in a file can be used for simulation by reproducing network conditions and investigation of adaptation techniques. The second file uses a tabular where each line represents one report. Such a file can be utilized by the function and data plotting tool *Gnuplot* [20] for visualization.

For evaluation of techniques for adaptation under certain network conditions, the Evaluation-Adaptation-Strategy is beneficial. It is specialized for the codecs AMR-NB, AMR-WB and H263+ but can easily be extended. The following functions are provided by the strategy:

- The incoming reports are saved in a file that can be used by Gnuplot.
- Gnuplot-commands for plotting are saved in a file.
- The tool NIST Net can be controlled according to parameters specified.

For control of NIST Net, the `EvaluationAdaptationStrategy` forks a new process. The bandwidth restriction with NIST Net and the periodic change can be adjusted by several parameters:

**\_\_start\_bandwidth:** The bandwidth in bytes per second which NIST Net uses in the first place (integer).

**\_\_stop\_bandwidth:** The bandwidth in bytes per second to stop with (integer).

**\_\_step\_width:** The bandwidth change in bytes per second per step (integer).

**\_\_step\_delay:** The delay between each step in seconds (integer).

**\_\_start\_delay:** The delay between receiving the first report and restricting the bandwidth for the first time in seconds (integer).

**\_\_turn\_mode:** Indicates whether the bandwidth should be increased inversely back to `__start_bandwidth` when it has reached `__stop_bandwidth` or not (bool).

These parameters can be specified in the configuration file `eval.config` with entries according to the format `<parameter>=<value>`. A comment line is initiated with the symbol `#`.

Currently, three files that contain commands for visualization of jitter, round trip delay and packet loss are written to the hard disk. These files can be used as input for Gnuplot. Together with the saved reports, the tool creates diagrams that show the temporal bandwidth change and its influence on jitter, round trip delay and packet loss, respectively. These diagrams are used in the following sections for illustration of bandwidth impact on network parameters. Since the strategy can be used in conjunction with an adaptation technique, one can also show the effect of adaptation on these parameters.

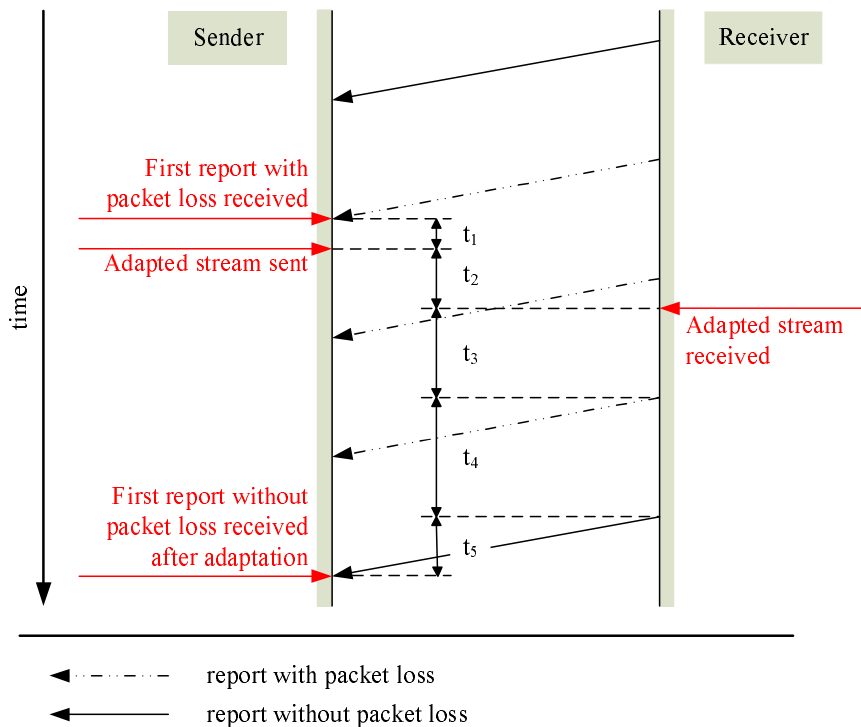


Figure 8.1: Illustration of adaptation delay

## 8.2 Adaptation Time

The Adaptation-Strategies presented in this thesis all base on information contained in RTCP reports. Per default, LIVE555 sends an RTCP report approximately every 5 seconds for a unicast connection. This sending rate imposes a very high delay between time of adaptation and effect on the media quality. Therefore, the sending rate of RTCP reports is increased to one report per second in order to achieve a faster reaction.

However, only NMM-RTP clients will benefit from this optimization for a faster reaction as the client side determines the sending rate of the relevant reports for sender-based adaptation. Moreover, the optimization is only reasonable in a unicast scenario or a multicast scenario with limited number of participants. This is due to the fact that the fraction of the bandwidth used for control will increase in a large scale multicast scenario. Therefore, the RTP standard recommends adding a fixed fraction of 5 percent of the session bandwidth for RTCP [62].

Figure 8.1 illustrates the delay between the time a RTCP report with packet loss is received and the time until a successful adaptation effect be-

comes visible in the received RTCP reports. The time intervals are as follows:

- $t_1$ : Time between receiving a report and sending the adapted stream; depends on sizes of buffers in the flow graph.
- $t_2$ : Delay of a packet from sender to receiver.
- $t_3$ : Time elapsing between receiving the adapted stream and sending the next RTCP report.
- $t_4$ : Time interval of sending RTCP reports.
- $t_5$ : Delay of a packet from receiver to sender.

For a consistent adaptation, the introduced delay needs to be considered. For example, if the bit rate is reduced due to packet loss, the next RTCP reports with packet loss must be ignored until the effect of the reduction is visible in the RTCP reports. Otherwise the bit rate would be reduced twice even if the first reduction was already sufficient for the current network condition. For calculation of the upper bound, the following observations are relevant:

- $t_2 + t_5 \approx$  round trip delay that can be calculated with the information contained in an RTCP report
- $t_4 =$  delay between sending of RTCP reports
- $t_3 <$  delay between sending of RTCP reports

In addition, a *safety\_interval* is added which yields to an upper bound of:

$$\sum_{i=1}^5 t_i < t_1 + RTT + 2 \cdot sending\_rate + safety\_interval$$

In order to minimize the delay imposed by buffers ( $t_1$ ), the buffer sizes for the relevant connections are set to one. Therefore, the time  $t_1$  can be neglected. In future, the sizes of buffers should be taken into account for the calculation. With *safety\_interval* set to one second and the *sending\_rate* of one report per second, the duration returned by the function `getAdaptationDuration(...)` is set to  $RTT + 3$  seconds.

## 8.3 AudioBitrateAdaptationStrategy

In Chapter 6, the integration of AMR-NB and AMR-WB was described. As they support different bit rate modes, they are integrated in order to enable adaptation for audio connections. The `AudioBitrateAdaptationStrategy` makes use of these different bit rate modes in order to reduce the amount of utilized bandwidth if the available bandwidth drops. In the following, the evaluation of RTCP reports for AMR streaming and the resulting adaptation logic are presented. For evaluation, the setup described in Chapter 4 is used of all scenarios while employing the streaming application described in Chapter 9. For the limitation of the available bandwidth, NIST Net is controlled by the Evaluation-Adaptation-Strategy which must be configured accordingly for each scenario (see Section 8.1).

### 8.3.1 AMR-NB Streaming without Adaptation

This section illustrates the evaluation of AMR-NB streaming without adaptation techniques. For all tests with AMR-NB, a mono wav-file with 16 bit quantization and sampling rate of 8000 Hz is used. Furthermore, the bit rate mode at the beginning is set to the highest mode (12.200 kbit/s, see Section 2.1.2). While the *serverregistry* is running on the client side, the application is started as follows on the server side:

```
adaptive_streamer_receiver -i <file> -f amr_nb -a disable -e
```

#### Scenario 1: Heavy Reduction of Available Bandwidth

In the first scenario, the bit rate is heavily reduced from 100 Mbit/s to 28.8 kbit/s with NIST Net, ten seconds after the first RTCP report is received. Shortly after the bandwidth limitation is applied, audible dropouts occur, although the limitation is well above the bit rate of the payload. This is due to the fact that the AMR packets are encapsulated into RTP frames. Furthermore, control traffic is to be considered.

Figure 8.2 illustrates the jitter, calculated round trip delay and packet loss of two runs with identical parameters. While jitter is rising, it does not seem to give valuable information for an adaptation technique as it is very fluctuating. Immediately after applying the reduction, the round trip time increases linearly until the router queue is filled up. By the time the queue of the router is completely filled, the round trip delay stays relatively constant at a high level. At this point in time, also the packet loss rises above 0 as the router must drop packets. While jitter and packet loss ratio show a similar shape, the shape of the round trip time is almost equal for both runs.

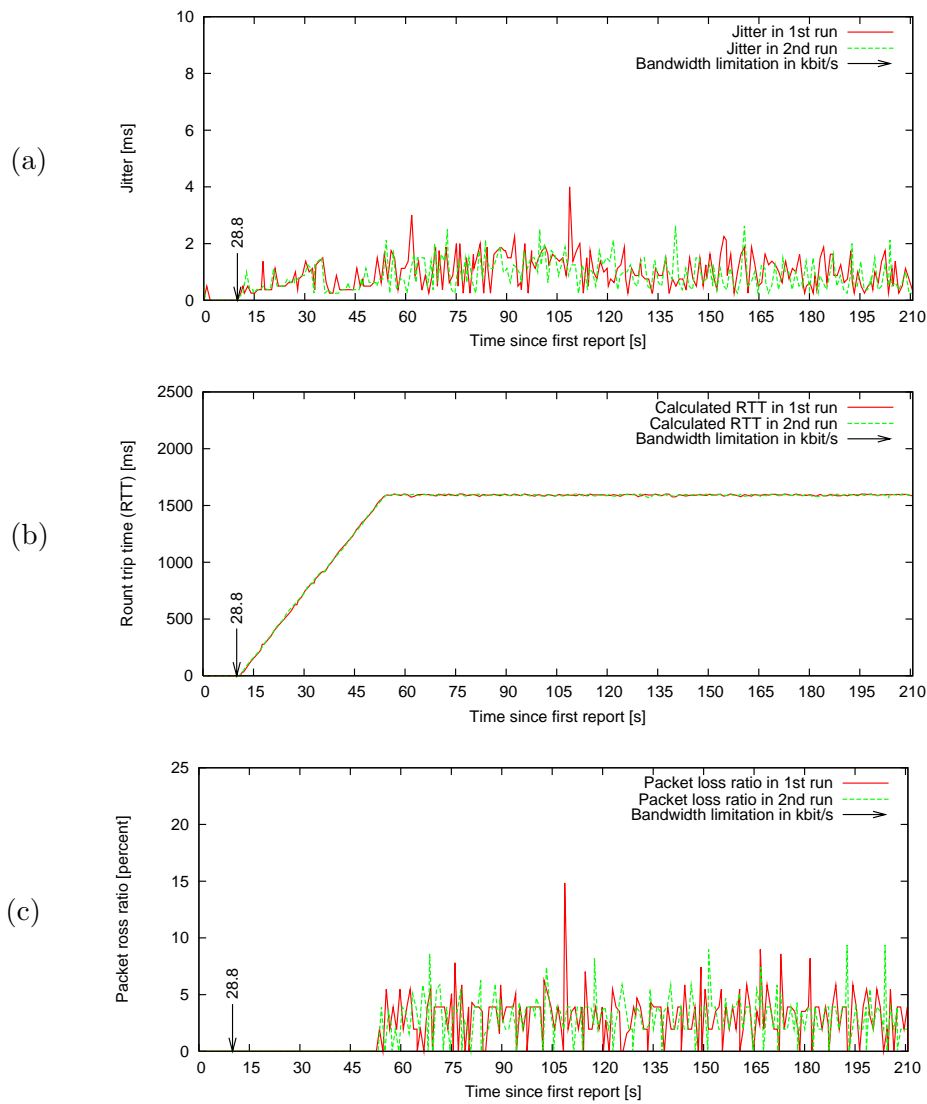


Figure 8.2: (a) Jitter, (b) round trip time and (c) packet loss ratio calculated from RTCP reports for AMR-NB streaming without adaptation while limiting the available bandwidth with NIST Net (Scenario 1).

### Scenario 2: Slow Reduction and Increase of Available Bandwidth

In the second scenario, the bit rate is reduced to 33.6 kbit/s, ten seconds after the first RTCP report is received. Afterwards, it is reduced in steps of 1.6 kbit/s every 15 s down to 24 kbit/s and increased in the same steps up to 33.6 kbit/s again. Noticeable dropouts occur not until the bandwidth is limited to 28.8 kbit/s. After further reductions, the dropouts become

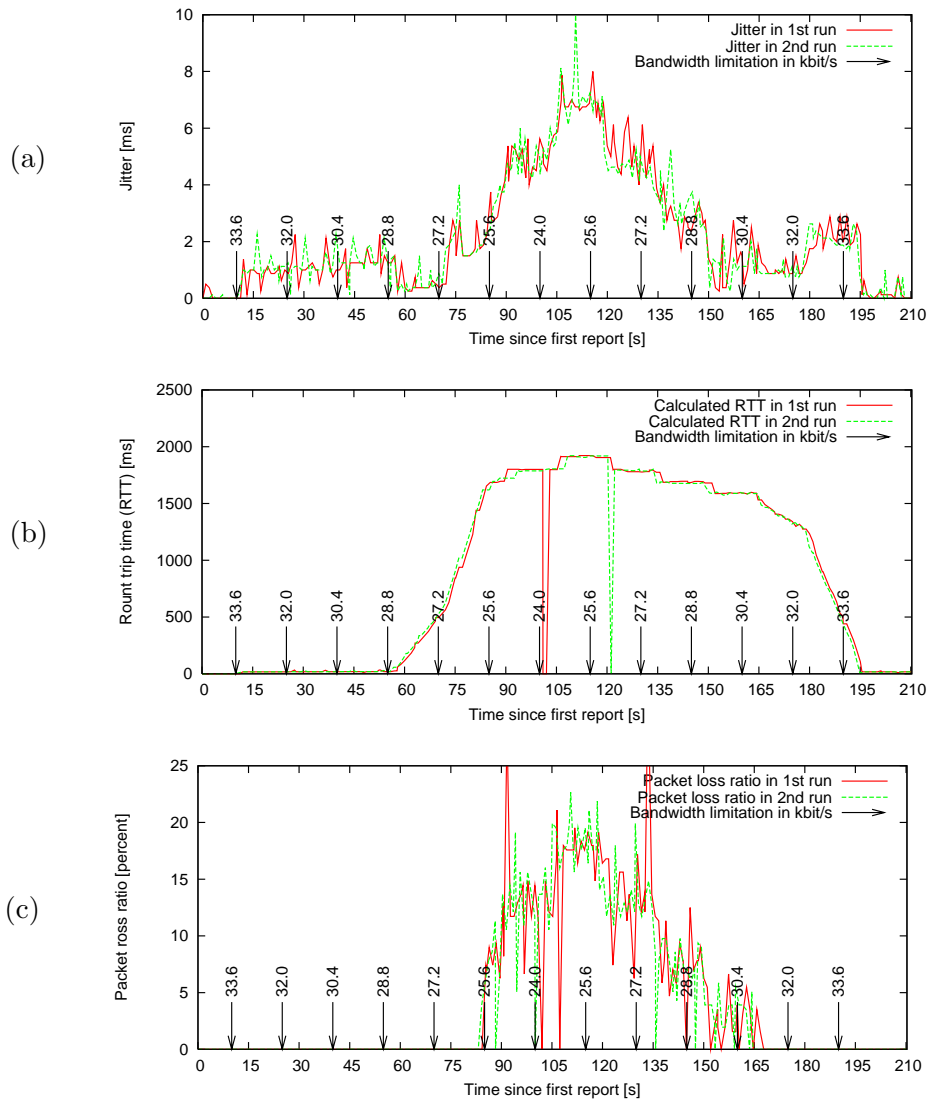


Figure 8.3: (a) Jitter, (b) round trip time and (c) packet loss ratio calculated from RTCP reports for AMR-NB streaming without adaptation while limiting the available bandwidth with NIST Net (Scenario 2).

heavier until the audio is completely distorted. This progression is inversely perceptible when the bandwidth is increased again.

Figure 8.3 illustrates the jitter, calculated round trip delay and packet loss of two runs with identical parameters. While the limitation of bandwidth is increasing, the jitter is rising. At the time when the limitation is decreased again, also the jitter is falling.

The round trip delay is rising faster as in scenario one. After the available bandwidth is increased again, the delay falls but significantly slower as it is increased.

As in scenario one, the packet loss occurs after the router queue is filled up but with a higher ratio. Similarly to round trip delay, the packet loss ratio is increasing faster as it is reduced.

Besides, there are outliers in the round trip time. In the first run, LIVE555 returns a round trip time of 0.0s for two consecutive RTCP reports (around 100s after the first report has been received); in the second run, LIVE555 returns 0.0s round trip time after around 120s (compare Figure 8.3). As a round trip time of 0.0s is not possible, LIVE555 returns wrong values in these cases.

### 8.3.2 AMR-WB Streaming without Adaptation

This section presents the evaluation of AMR-WB streaming without using adaptation techniques. For all tests with AMR-WB, a mono wav-file with 16 bit quantization and sampling rate of 16000 Hz is used. Furthermore, the bit rate mode at the beginning is set to the highest mode (23.85 kbit/s, see Section 2.1.2). While the *serverregistry* is running on the client side, the application is started as follows on the server side:

```
adaptive_streamer_receiver -i <file> -f amr_wb -a disable -e
```

#### Scenario 1: Heavy Reduction of Available Bandwidth

The scenario is identical to scenario one for AMR-NB except that the bit rate is reduced to 40 kbit/s instead of 28.8 kbit/s. The results are shown in Figure 8.4. They are very similar to the results gained from the scenario for AMR-NB.

#### Scenario 2: Slow Reduction and Increase of Available Bandwidth

The scenario is identical to scenario two for AMR-NB except that the bit rate limitation is as shown in Figure 8.5.

Similar to AMR-NB, there are outliers in the second run for the round trip time (around 120s after the first report has been received). Moreover, the jitter values at the end of the graph are peculiar. They suddenly rise to more than 30 ms when the routers queue is empty and stay at this high level (compare Figure 8.5).

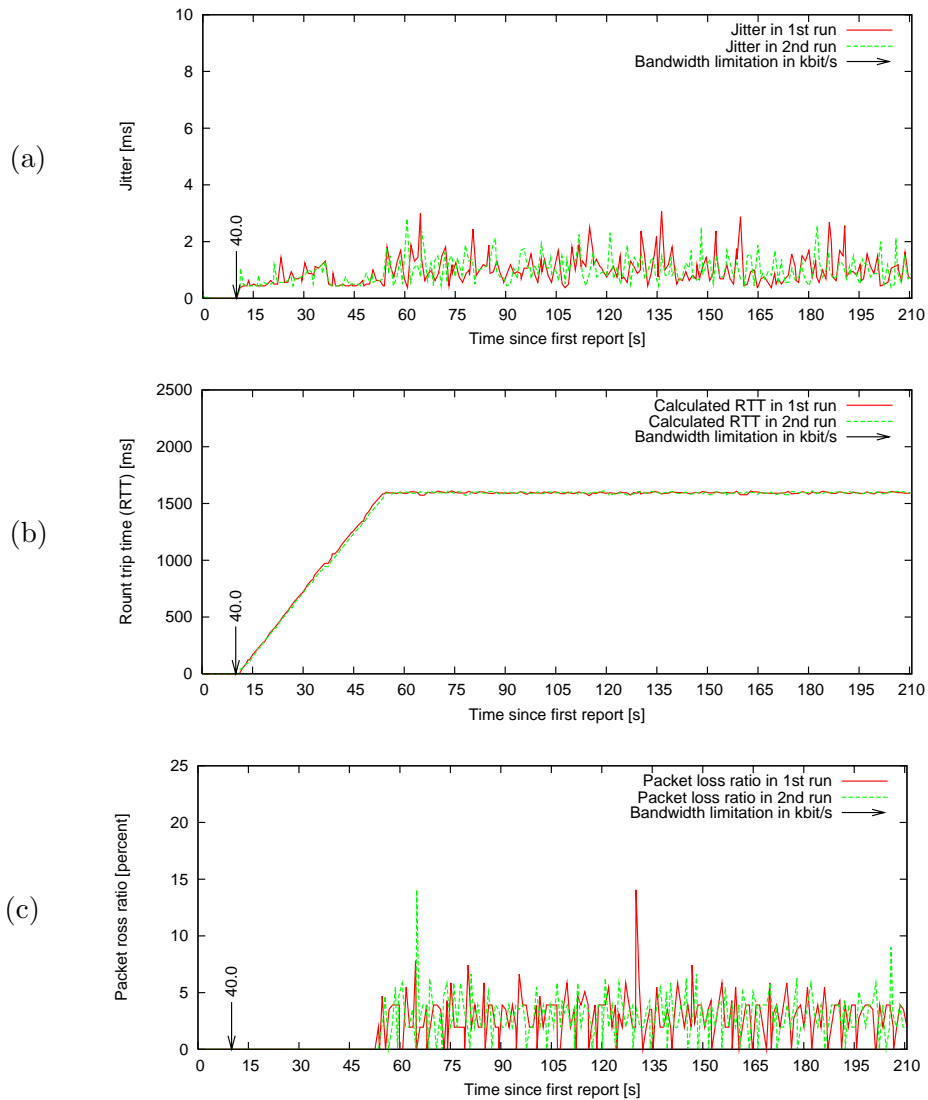


Figure 8.4: (a) Jitter, (b) round trip time and (c) packet loss ratio calculated from RTCP reports for AMR-WB streaming without adaptation while limiting the available bandwidth with NIST Net (Scenario 1).

### 8.3.3 Adaptation Logic

During evaluation it turned out that the thresholds for noticeable dropouts are at bandwidth limitations of 26.6 kbit/s for AMR-NB and 40 kbit/s for AMR-WB (each in the highest bit rate mode). However, the packet loss ratio of RTCP reports does not indicate any loss in the beginning. This arises from the fact that the audio packets are relatively small compared to

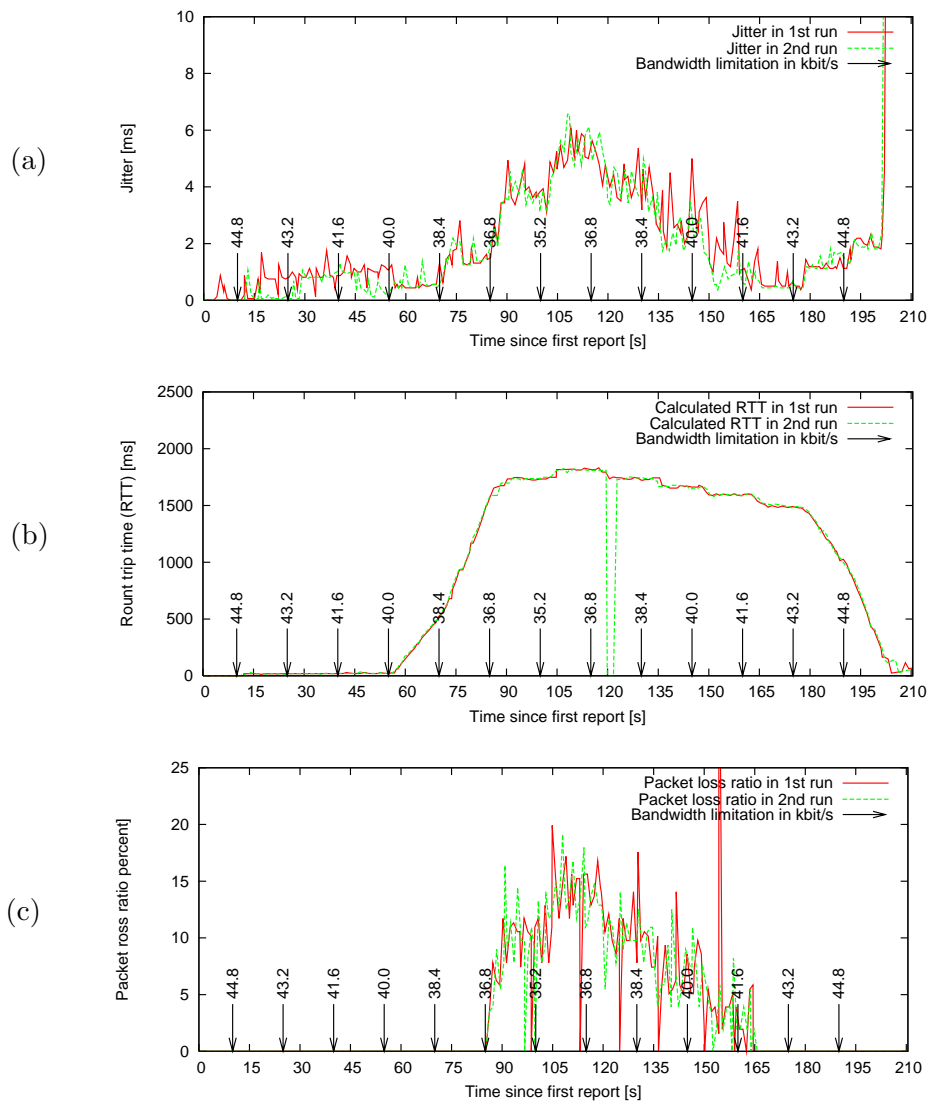


Figure 8.5: (a) Jitter, (b) round trip time and (c) packet loss ratio calculated from RTCP reports for AMR-WB streaming without adaptation while limiting the available bandwidth with NIST Net (Scenario 2).

the queue of the router. Anyhow, the constantly increasing delay (calculated from RTCP reports) prevents the packets to arrive at the receiver side in time. If the queue of the router is completely filled, the round trip delay stays relatively constant at a high level. At this point in time, also the packet loss ratio rises above 0 as the router must drop packets (compare Figure 8.2 to Figure 8.5).

The variation of the jitter could not be matched with bandwidth changes. Therefore, the adaptation logic of the `AudioBitrateAdaptationStrategy` does not make use of jitter values. Since the interval between noticeable quality degradation and indication of packet loss is far too great, the packet loss ratio cannot adequately be used for purposes of adaptation in this case. However, a continuous increasing round trip delay is a good indication for congestion as shown in the previous sections. The adaptation logic for the `AudioBitrateAdaptationStrategy` is described in the following.

The class `AdaptationStrategy` (see Figure 7.10 on page 80) possesses an archive of QoS-Reports. This archive is utilized for determining the progression of round trip delay times. The rules for decrease and increase are as follows:

- If the round trip delay is rising two or more times consecutively by 20 ms or more, the bit rate mode is reduced by two. Furthermore, the next increase to a higher bit rate mode is delayed at least 20 reports.
- If the round trip delay is rising two or more times consecutively between 10 ms and 20 ms, the bit rate mode is reduced by one. Furthermore, the next increase to a higher bit rate mode is delayed at least 20 reports.
- The switching to a higher bit rate mode is commenced if the round trip delay indicated in the last 5 or more consecutive reports has not increased from one report to the next one by more than 1 ms.
- If packet loss occurs, the bit rate mode is lowered by 3 levels and the next increase to a higher bit rate mode is delayed at least 20 reports.

#### 8.3.4 AMR-NB Streaming with Adaptation

In this section, the scenarios conducted without adaptation are now evaluated using the Adaptation-Strategy as described above. While the *server-registry* is running on the client side, the application is started as follows on the server side:

```
adaptive_streamer_receiver -i <file> -f amr_nb -e
```

##### Scenario 1: Heavy Reduction of Available Bandwidth

The Adaptation-Strategy outlined above completely avoids packet loss as it adjusts the bit rate before packet loss can emerge. Therefore, in the following, the bit rate change is shown in the diagrams instead of packet loss

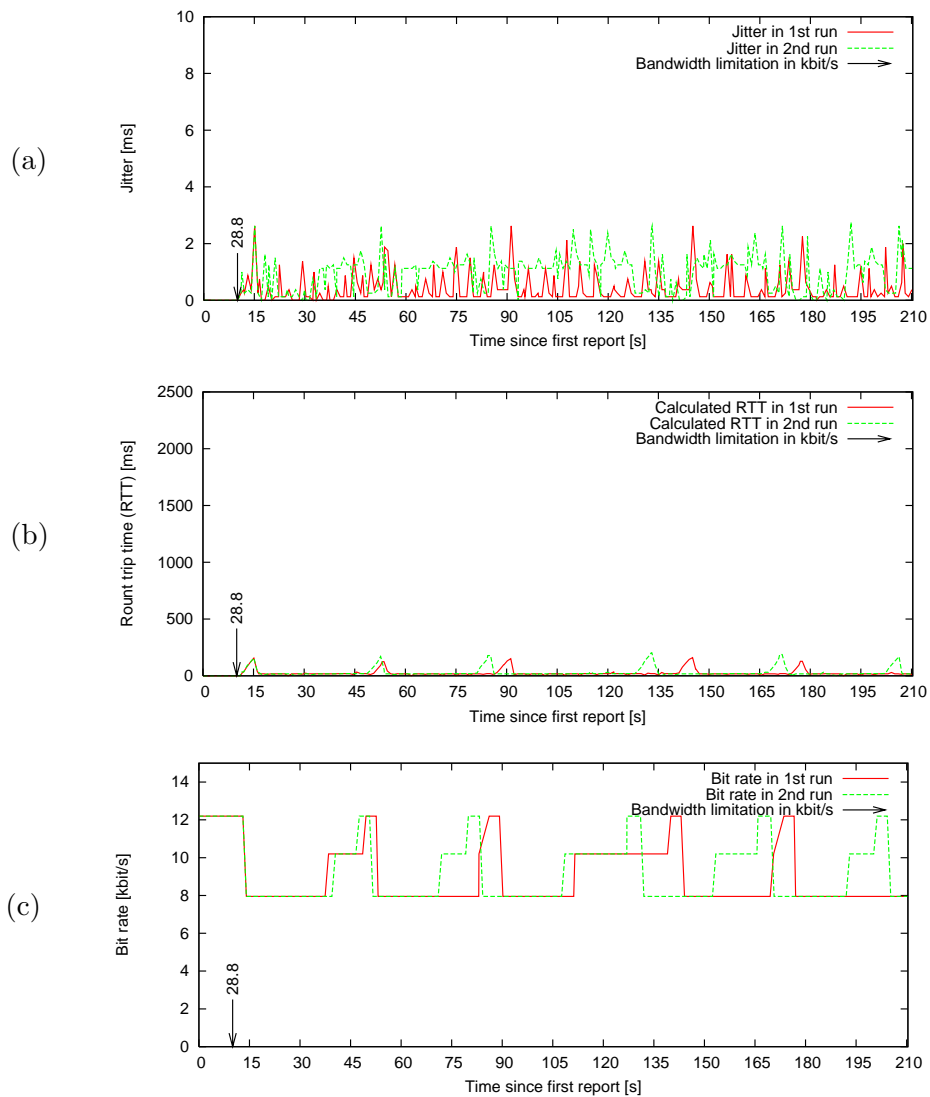


Figure 8.6: (a) Jitter, (b) round trip time and (c) bit rate change for AMR-NB streaming with adaptation while limiting the available bandwidth with NIST Net (Scenario 1).

ratio. Figure 8.6 illustrates streaming of AMR-NB according to scenario one but with adaptation enabled.

The round trip delay diagram includes only a number of small peaks that immediately result in a bit rate reduction. The bit rate mode is oscillating around the optimal value while maintaining an audio streaming without distortion.

**Scenario 2: Slow Reduction and Increase of Available Bandwidth**

Figure 8.7 illustrates the second scenario using AMR-NB while employing the adaptation logic as described above. As desired, the bit rate is reduced when the bandwidth is limited and slowly increasing again when the available bandwidth increases. Moreover, the round trip delay is kept to a small value.

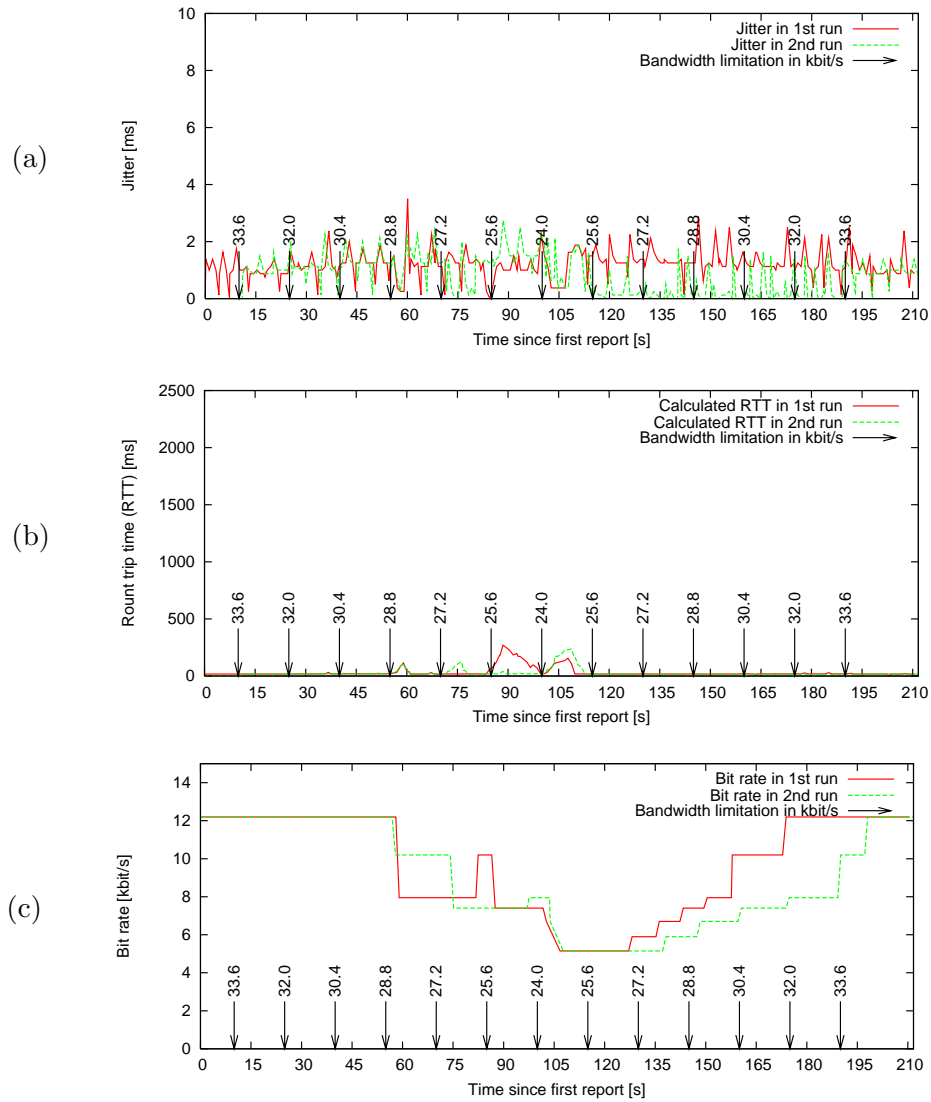


Figure 8.7: (a) Jitter, (b) round trip time and (c) bit rate change for AMR-NB streaming with adaptation while limiting the available bandwidth with NIST Net (Scenario 2).

### 8.3.5 AMR-WB Streaming with Adaptation

For streaming AMR-WB with adaptation, the application is started on the server side while the *serverregistry* is running on the client side:

```
adaptive_streamer_receiver -i <file> -f amr_wb -e
```

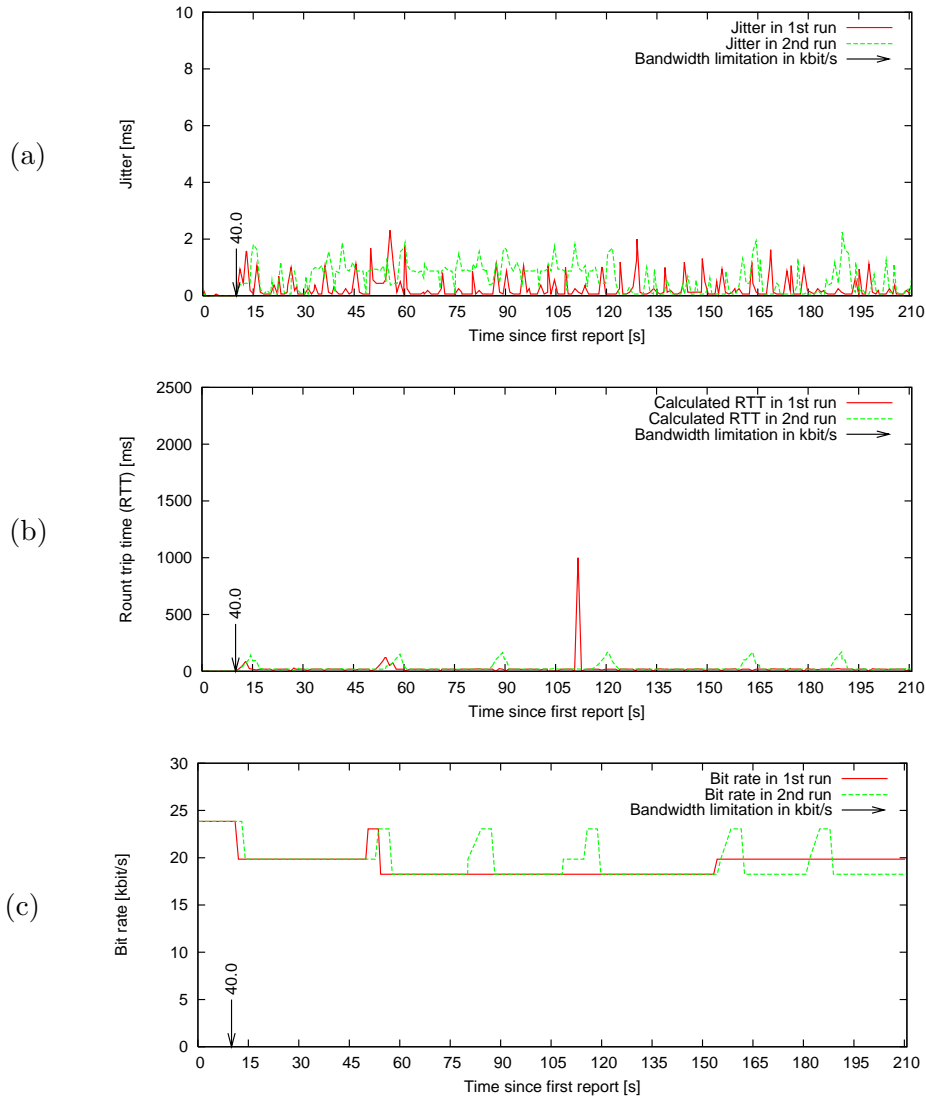


Figure 8.8: (a) Jitter, (b) round trip time and (c) bit rate change for AMR-WB streaming with adaptation while limiting the available bandwidth with NIST Net (Scenario 1).

The results for AMR-WB are very similar to the results for AMR-NB as illustrated in Figure 8.8, and Figure 8.9, respectively.

Again, packet loss can be avoided completely and the round trip delay is mostly very small. Therefore, the AMR-WB stream can be transmitted with limited bandwidth without greater distortion.

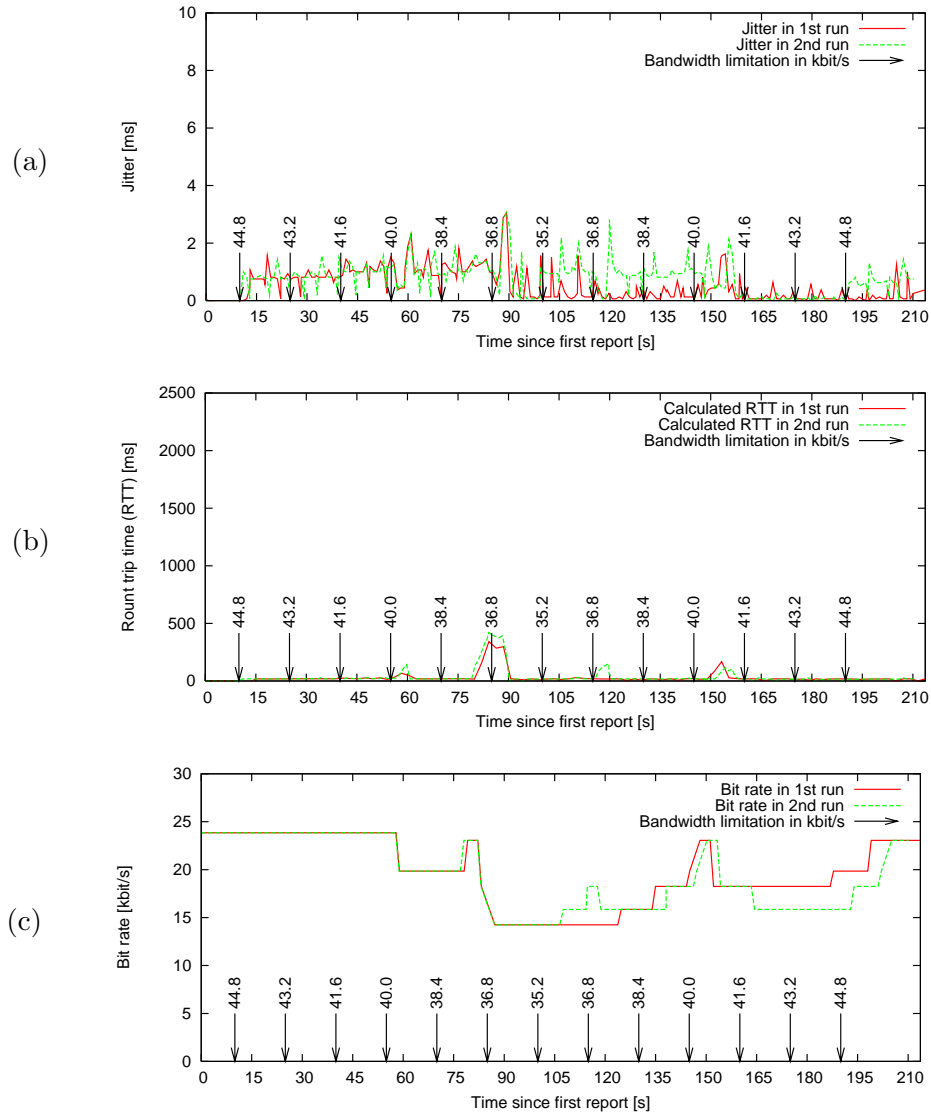


Figure 8.9: (a) Jitter, (b) round trip time and (c) bit rate change for AMR-WB streaming with adaptation while limiting the available bandwidth with NIST Net (Scenario 2).

## 8.4 VideoBitrateAdaptationStrategy

As the name suggests, the `VideoBitrateAdaptationStrategy` can adapt the bit rate of a video stream. If the available bandwidth drops, the bit rate is decreased to maintain an acceptable video quality. In the following, the evaluation of RTCP reports for H263+ streaming and the resulting adaptation logic are presented. As for the Audio-Bitrate-Adaptation-Strategy, the setup described in Chapter 4 is used for evaluation of all scenarios while employing the streaming application described in Chapter 9. Again, the limitation of the available bandwidth is performed by the Evaluation-Adaptation-Strategy (using NIST Net) which must be configured accordingly for each scenario (see Section 8.1).

### 8.4.1 Video Streaming without Adaptation

This section illustrates the evaluation of H263+ streaming without adaptation techniques. For all tests with H263+ streaming, an MPEG2 file with resolution 720x480 and 30 frames/s is used while disabling audio streaming. The bit rate of the H263+ encoder is set to 2 Mbit/s at the beginning of each test. While the *serverregistry* is running on the client side, the application is started as follows on the server side:

```
adaptive_streamer_receiver -i <file> -m video -a disable -e
```

#### Scenario 1: Heavy Reduction of Available Bandwidth

In the first scenario, the bit rate is heavily reduced from 100 Mbit/s to 1.28 Mbit/s with NIST Net, ten seconds after the first RTCP report is received. Figure 8.10 shows the results.

Immediately after applying the bandwidth limitation, the jitter is rising heavily. However, it is very fluctuating afterwards. Until 30 s after the first report no packets are lost. The first time the packet loss is rising above 0 also the delay is increasing by a great amount.

#### Scenario 2: Slow Reduction and Increase of Available Bandwidth

Similarly to the second scenario for AMR, the bit rate is reduced slowly and increased again as shown in Figure 8.11. Different from audio streaming with AMR, the bit rate of H263+ is not constant as it depends on the content of the pictures of the video. For example, if a picture contains a large area of the same color, only fewer bits for encoding are needed than for a picture with many details. Therefore, the mapping of packet loss ratio and bandwidth limitation is not as clearly as for AMR.

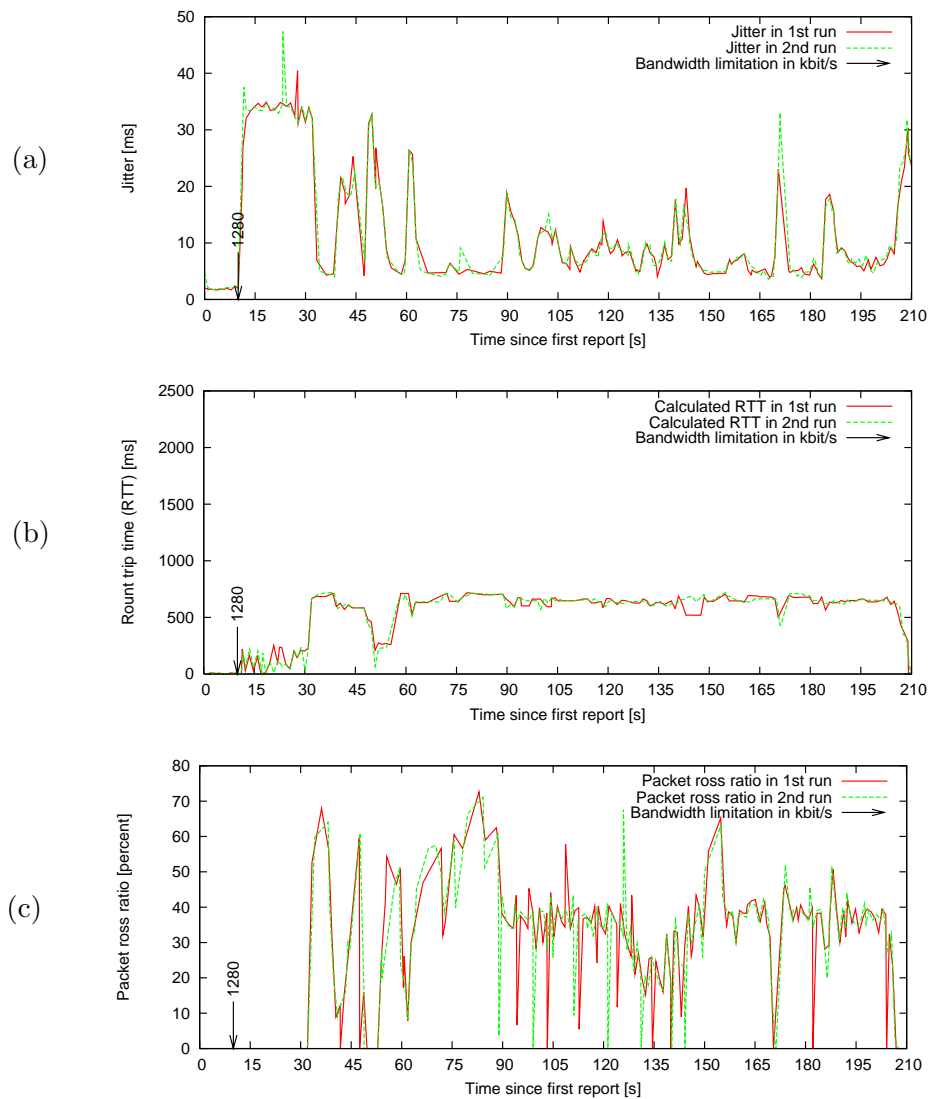


Figure 8.10: (a) Jitter, (b) round trip time and (c) packet loss ratio calculated from RTCP reports for H263+ streaming without adaptation while limiting the available bandwidth with NIST Net (Scenario 1).

### 8.4.2 Adaptation Logic

For indication of decreasing bandwidth, the value of packet loss is considered. The general idea for adaptation is as follows: If the network is congested and packet loss occurs, the fast recovery of an acceptable quality is considered as most important. Therefore, the bit rate is reduced by a relatively large amount depending on the packet loss ratio. If a number of reports without

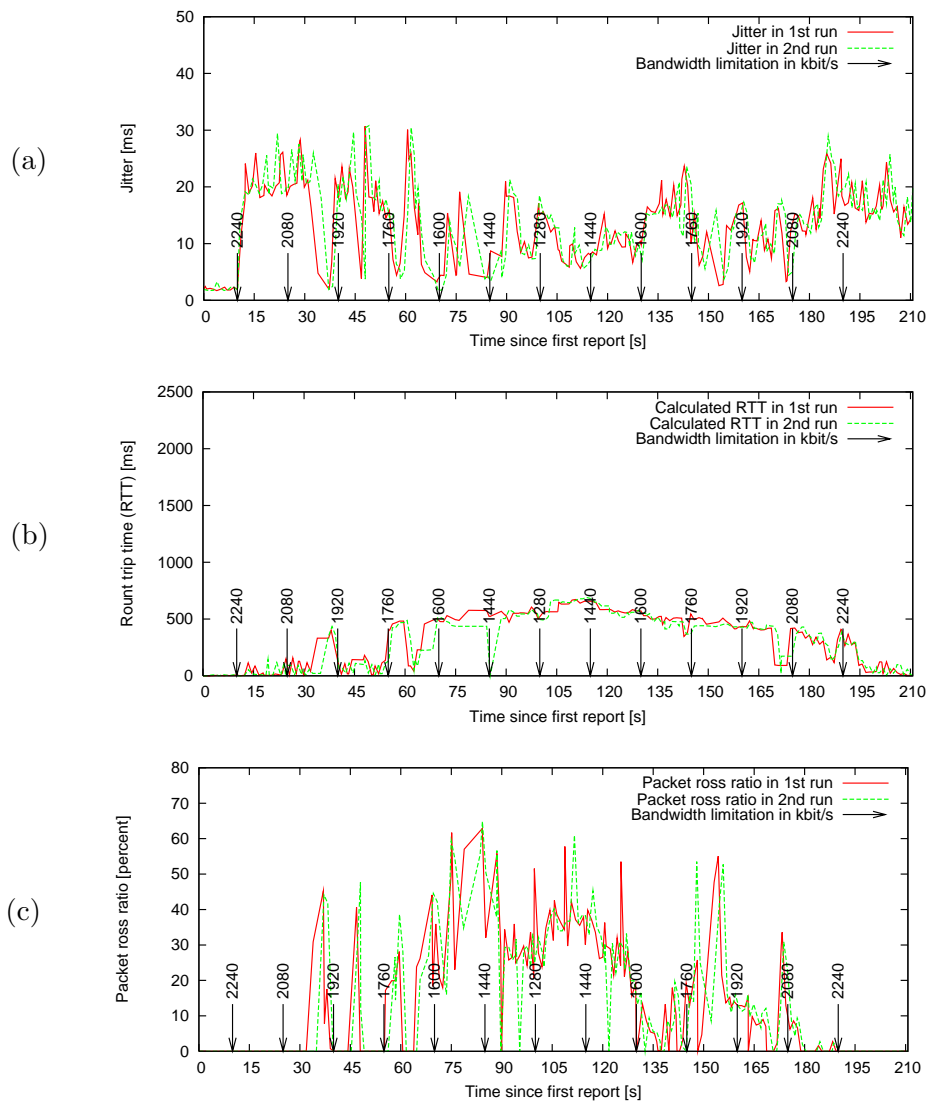


Figure 8.11: (a) Jitter, (b) round trip time and (c) packet loss ratio calculated from RTCP reports for H263+ streaming without adaptation while limiting the available bandwidth with NIST Net (Scenario 2).

packet loss are received, the bit rate is increased by a small amount. Essentially, this is an AIMD scheme which is also used in the congestion control of TCP as described in Section 2.2.1. Moreover, the adaptation history is considered. If the bit rate was just increased and packet loss occurs, the bit rate is only reduced by a small amount. Additionally, the number of reports to wait until the bit rate is increased the next time is depending on the previously conducted adaptation. This is controlled by the member

variable `__next_increase`. Until now, the loss of complete RTCP reports is not considered. This is a task of future work. In detail, there are four cases which are handled differently. Each corresponds to a private method in the `VideoBitrateAdaptationStrategy`.

1. `handleNoPacketLoss()`: If no packet loss occurs and the next increase is not delayed, the bit rate is increased by a moderate amount of 10000 bits per second. A larger amount has the advantage that the bit rate reaches the highest possible value faster. The disadvantage is that the packet loss, which can be induced by a large step beyond the available bandwidth, is much greater. This can lead to higher quality degradation. The same considerations apply for a non-linear increase of bit rate. As a stable video quality is considered more important than a faster increase of the bit rate, the amount is chosen moderately. The variable `__next_increase` is set to 2.
2. `handlePacketLossAfterIncrease()`: If packet loss occurs right after increasing the bit rate, the previously used bit rate is considered to be the maximum value possible for the currently available bandwidth. Therefore, the bit rate is set to the old value reduced by 10 percent as a safety reduction. Additionally, the minimal delay for the next increase is set to a much larger value (`__next_increase = 10`).
3. `handleLevel1PacketLoss()`: If a relatively small amount of packets are lost ( $0\% < \textit{packet loss} < 1\%$ ) the bit rate is reduced by a quarter. The next increase is at least delayed by 3 reports (`__next_increase = 3`).
4. `handleLevel2PacketLoss()`: For a larger percentage of packet loss ( $\textit{packet loss} > 1\%$ ), the reaction is the same as for case 3 except that the bit rate is halved.

This adaptation logic is similar to the one presented in [58]. In [58], however, it is applied statically to all codecs. Even though adaptation based on packet loss works well for a video stream, an audio stream needs a different approach as discussed in Section 8.3.3. Therefore, employing flexible Adaptation-Strategies is beneficial for the purpose of adaptation.

### 8.4.3 Video Streaming with Adaptation

In this section, both scenarios are conducted while applying the adaptation logic as described in the previous section. Figure 8.12 and Figure 8.13 show the results for scenario one and scenario two, respectively.

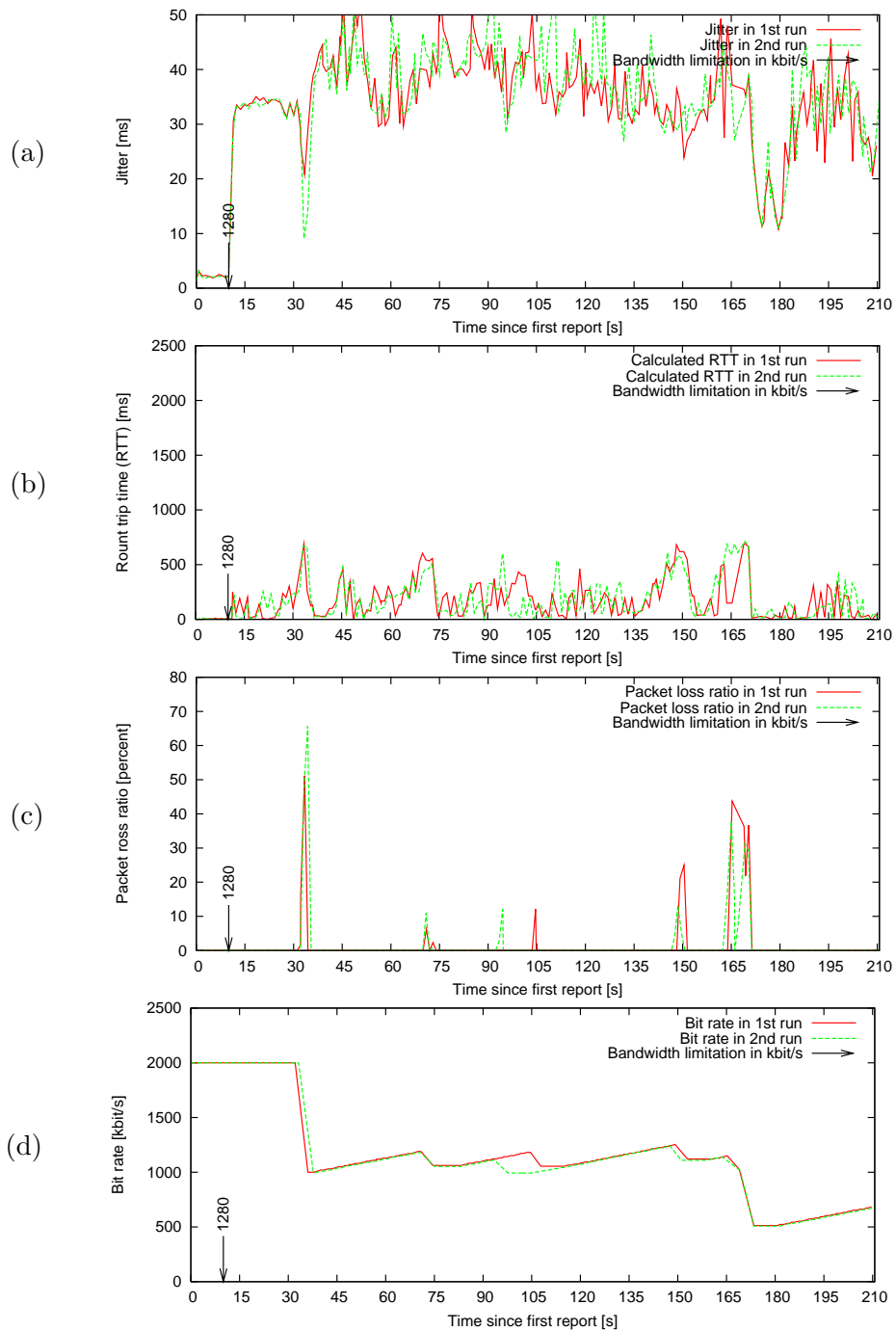


Figure 8.12: (a) Jitter, (b) round trip time, (c) packet loss ratio and (d) bit rate change for H263+ streaming with adaptation while limiting the available bandwidth with NIST Net (Scenario 1).

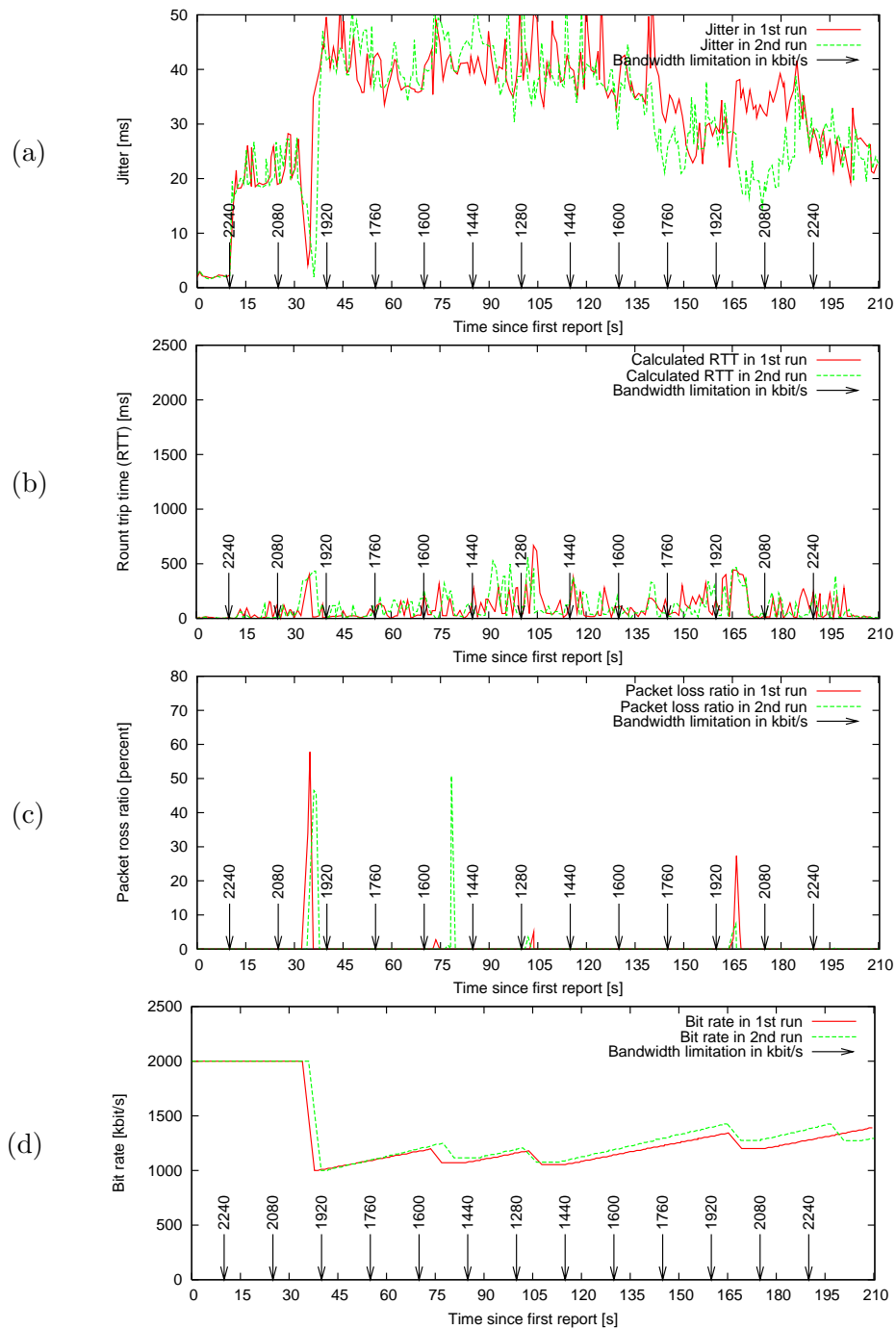


Figure 8.13: (a) Jitter, (b) round trip time, (c) packet loss ratio and (d) bit rate change for H263+ streaming with adaptation while limiting the available bandwidth with NIST Net (Scenario 2).

For streaming H263+ with adaptation, the application is started on the server side while the *serverregistry* is running on the client side:

```
adaptive_streamer_receiver -i <file> -m video -a disable -e
```

As one can see, only peaks of packet loss are contained in the diagrams. Afterwards, the packet loss ratio drops quite fast to 0 again. Peaks of packet loss are unavoidable as the adaptation applies only after packet loss occurs. Anyhow, the quality compared to a video without adaptation is much better. This is also confirmed by the overall number of packets lost as indicated by RTCP reports (compare Figure 8.14). These results show that even though the adaptation logic is relatively simple, the outcome is quite good.

	Scenario 1		Scenario 2	
	with adaptation	without adaptation	with adaptation	without adaptation
1st run	701	15282	442	8740
2nd run	751	15273	684	4754

Figure 8.14: Overall number of packets lost in the different scenarios with and without adaptation.

## 8.5 FramerateAdaptationStrategy

Similarly to the Adaptation-Strategy presented in the last section, the strategy presented in this section, called **FramerateAdaptationStrategy**, utilizes packet loss of RTCP reports for indication of network congestion. During development, it turned out that not all frame rates could be streamed with libavcodec and LIVE555 using the H263+ codec while maintaining a uniform distribution of frames over the time. The usable frame rates are divisible by 5 and greater or equal to 15 frames per second. The remaining variation for example 15 fps, 20 fps, 25 fps, 30 fps is still considered to be reasonable.

Giving consideration to this result, the **FramerateAdaptationStrategy** uses so-called frame rate levels. The strategy determines the largest supported frame rate which is the one specified by the application in the first place. The lower levels are calculated accordingly.

The adaptation logic is described in the following. If packet loss occurs and the minimal level is not reached yet, the frame rate is decreased by one level. If a number of reports without packet loss are received, the frame rate is increased by one level. When the frame rate is increased, the step is much larger as a compared step of a bit rate increase used by the adaptation logic

of the `VideoBitrateAdaptationStrategy`. Hence, also the delay between increasing steps is set to a larger value. The corresponding member variable `__next_increase` is set to

- 20, if a packet loss immediately after a frame rate increase occurs.
- 15, if a packet loss occurs and the last action was not a frame rate increase.
- 10, if the frame rate was just increased.

As the bit rate is not depending on the frame rate, the adjustment of the frame rate can not be used to reduce the utilized bandwidth. Therefore, it is not reasonable to use this strategy stand-alone. Anyhow, it can be used in conjunction with the `VideoBitrateAdaptationStrategy`. By reduction of the frame rate, more bits can be used for encoding each frame. Although both strategies can be used together but independently, it is more reasonable to combine them in a `CompositeAdaptationStrategy`. This approach is introduced in the next section.

## 8.6 BitrateFramerateAdaptationStrategy

The concept of `CompositeAdaptationStrategies` was introduced in Section 7.6. A beneficial application is presented now.

The authors of [8] conducted a survey which examined the user preference of different adaptation techniques for digital video. It showed that the acceptance for a video with lowered bit rate is higher if the frame rate is lowered as well. As described in the last section, a lower frame rate leads to a better quality per frame but a lower temporal resolution.

With the use of a `CompositeAdaptationStrategy`, the combination of two strategies is simple. The `BitrateFramerateAdaptationStrategy` has a reference to the `VideoBitrateAdaptationStrategy` as well as the `FramerateAdaptationStrategy`. The range of possible bit rates is now divided uniformly into a number of levels. The number of levels equals to the number of different frame rate levels. Hence, each bit rate level can be assigned a corresponding frame rate level. An example of the different levels and their usage is visualized in Figure 8.15.

For adaptation, the bit rate is changed according to the rules of the `VideoBitrateAdaptationStrategy`. The frame rate is set to the level that corresponds to the currently used bit rate level.



Figure 8.15: Example of four uniformly distributed bit rate and frame rate levels.

## 8.7 Summary

In this chapter, several strategies not only for adaptation but also for evaluation were presented. They demonstrate the flexibility of the developed approach. Both adaptation techniques for audio transmission as well as video transmission were considered. Moreover, the ability to combine different Adaptation-Strategies is demonstrated by means of the developed `BitrateFramerateAdaptationStrategy`.

For evaluation, several parameters of the RTCP reports were considered. They show that video and audio need different techniques for adaptation. By the modular structure of the framework, these requirements can be met easily. Furthermore, the results show that even with relatively simple approaches the quality of a transmitted video and audio can be increased significantly.

## Chapter 9

# An Adaptive Streaming Application

In order to demonstrate the functionality of the presented framework and the developed Adaptation-Strategies, an adaptive streaming application, called *adaptive\_streamer\_receiver*, is developed in the scope of this thesis.

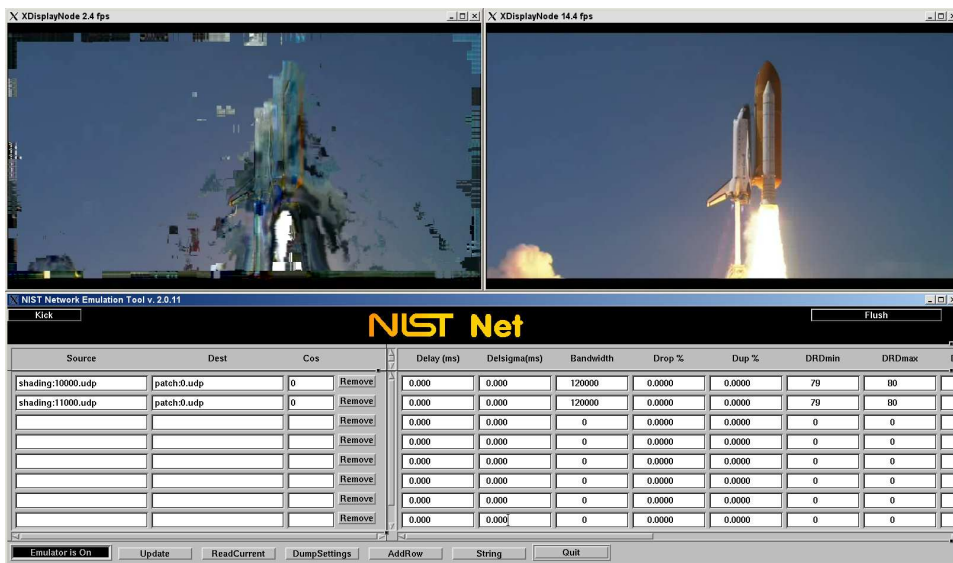


Figure 9.1: Screenshot of the developed application. The video on the left hand side is streamed with adaptation disabled. The video on the right hand side is streamed with adaptation enabled. While streaming, the available bandwidth is limited with the network emulator NIST Net.

The application supports several modes and formats which are described together with the general structure in the first part of this chapter. The second part covers the various command line options that are available.

A screenshot of two running instances of the application – one with adaptation enabled, the other one with adaptation disabled – is shown in Figure 9.1.

## 9.1 Structure

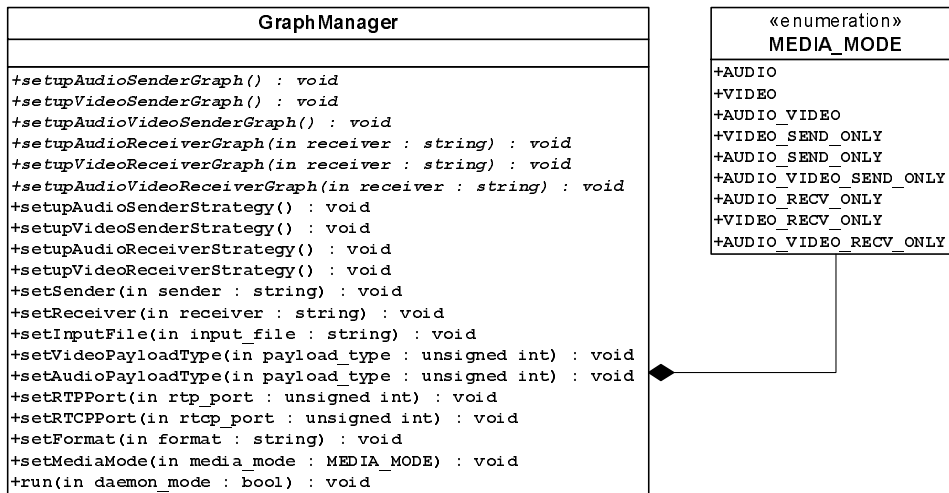
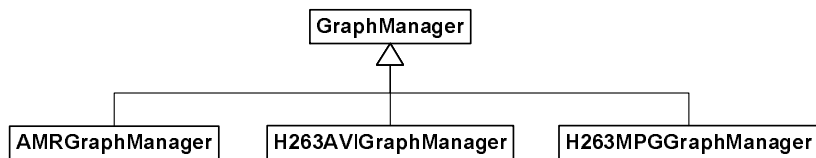
In order to present the developed Adaptation-Strategies, the application supports the streaming and receiving of the following formats:

- AMR-NB audio
- AMR-WB audio
- H263+ video with MP3 audio

Furthermore, the application supports different modes of operation. They are distinguished with the enum `MEDIA_MODE`.

1. `AUDIO`: sending and receiving of audio only
2. `VIDEO`: sending and receiving of video only
3. `AUDIO_VIDEO`: sending and receiving of both audio and video
4. `AUDIO_SEND_ONLY`: sending of audio only
5. `VIDEO_SEND_ONLY`: sending of video only
6. `AUDIO_VIDEO_SEND_ONLY`: sending of both audio and video
7. `AUDIO_RECV_ONLY`: receiving of audio only
8. `VIDEO_RECV_ONLY`: receiving of video only
9. `AUDIO_VIDEO_RECV_ONLY`: receiving of both audio and video

In the first three modes, the transparent access capabilities of NMM (see Section 5.3.5) are used to setup an NMM-graph for sending on the streaming host and an NMM-graph for receiving on the receiving host. In these modes, the application can be started on the streaming host, whereas on the receiving host only a running *serverregistry* is required. The difference between the three modes is whether audio only, video only or both is used

Figure 9.2: Class diagram of the class **GraphManager**.Figure 9.3: Class hierarchy of **GraphManager** and derived classes.

for streaming and receiving. Accordingly, the modes 4-6 and 7-9 are used for streaming only or receiving only, respectively.

In order to handle all combinations of different modes and formats, a clear structure is required. Therefore, the class **GraphManager** is developed. The **GraphManager** encapsulates the setup of the NMM-graphs. Hence, it contains all Node-Descriptions used in the different possible NMM-graphs. Moreover, it provides abstract functions for the various combinations of graphs (see Figure 9.2). Figure 9.3 shows the derived classes that implement these abstract methods for the different formats.

As the graphs of AMR-NB and AMR-WB only differ in the payload type used for the **RTPReceiverStrategy**, only one derived class for both codecs, the **AMRGraphManager**, is developed. The Graph for sending and receiving is shown in Figure 9.4.

In contrast, for H263+ two classes are developed as both input from avi-files and mpg-files is supported. They are called **H263AVIGraphManager** and **H263MPGGraphManager**, respectively. The Graphs for sending and receiving are illustrated in Figure 9.5 and Figure 9.6.

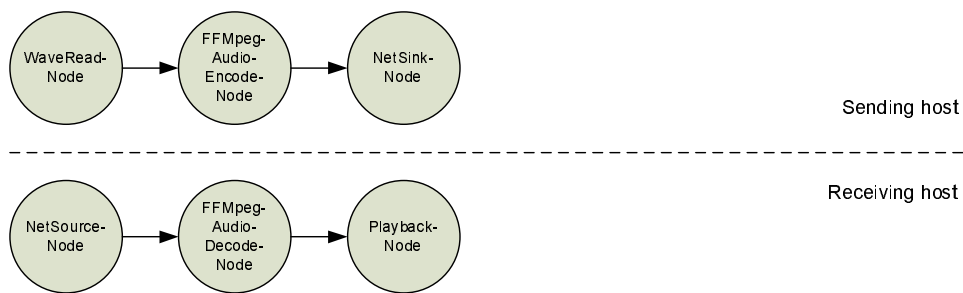


Figure 9.4: Graphs for streaming and receiving of AMR-NB and AMR-WB using a WAV file as input.

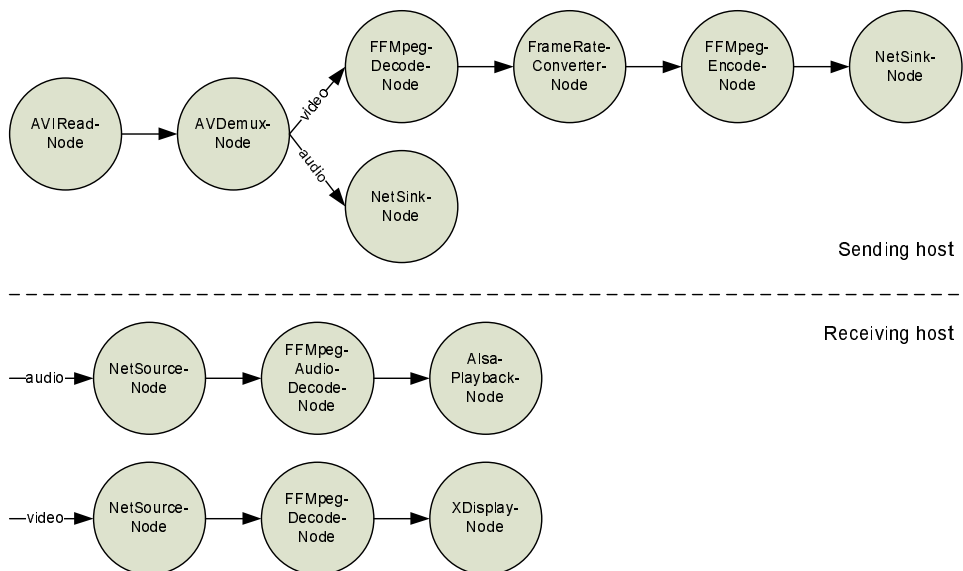


Figure 9.5: Graphs for streaming and receiving of H263+ video and MP3 audio using an AVI-file as input.

The `GraphManager` has member variables for various parameters such as ports, hosts and input file name. They can be set through corresponding methods. All parameters that can be used for configuration of the application are described in the next section.

In order to allow for adaptation, an Adaptor is integrated in the main method of the application:

```
// realize and start graph
graph.realizeGraph();
graph.startGraph();
```

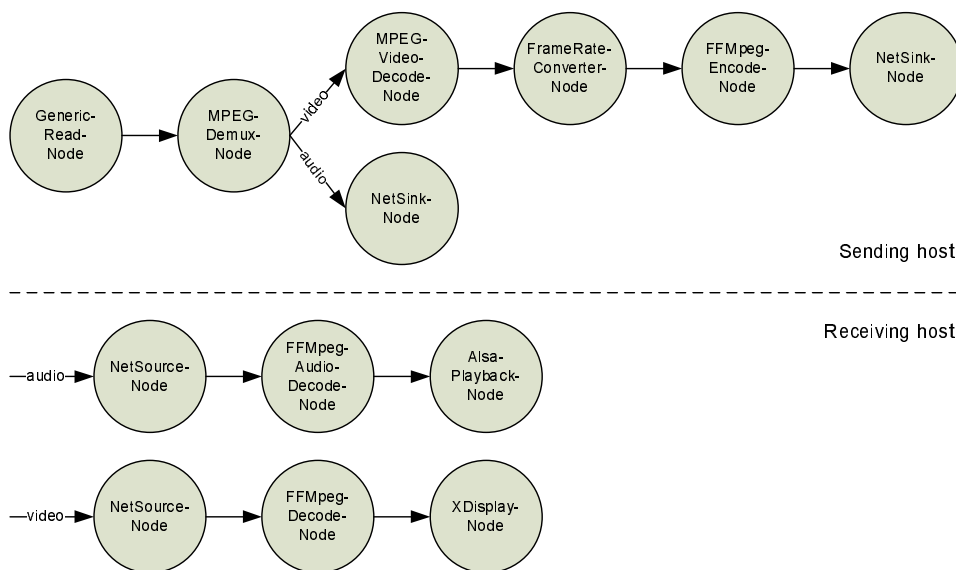


Figure 9.6: Graphs for streaming and receiving of H263+ video and MP3 audio using an MPG-file as input.

```

Adaptor adaptor(graph);

if (media_mode != GraphManager::AUDIO_VIDEO_RECV_ONLY
    && media_mode != GraphManager::AUDIO_RECV_ONLY
    && media_mode != GraphManager::VIDEO_RECV_ONLY) {

    if (enable_statistics) {
        adaptor.enableStatistics();
    }
    if (enable_evaluation) {
        adaptor.enableEvaluation();
    }
    if (adaptation_enabled) {
        adaptor.registerStrategies();
    }

} else {
    // on receiving side, adaptation is disabled
    adaptor.disableAdaptation();
}

adaptor.finishSetup();

```

The object `graph` is the Graph-Description which was set up by the Graph-Manager. After the graph has been realized and started, an Adaptor object is created. Depending on the specified command line parameters (see Section 9.2), the strategies for statistics or evaluation are used. After the strategies for adaptation are registered, the initialization of the Adaptor is completed by executing the method `finishSetup()`.

The method `run(bool daemon_mode, Adaptor& adaptor)`, which is a member of the `GraphManager`, is utilized to start an infinite loop. If it is not started in *daemon\_mode*, commands can be used to control the application and for testing purposes:

- **stop**: stops the Graph.
- **start**: starts the Graph.
- **register strategies**: registers strategies at the Adaptor.
- **enable adaptation**: enables adaptation if strategies are registered.
- **disable adaptation**: disables adaptation.
- **set video bit rate**: reads from the standard input and sets the video bit rate to the given value.
- **set audio bit rate**: reads from the standard input and sets the audio bit rate to the given value.
- **q**: terminates the application.

## 9.2 Command Line Parameters

In this section the command line parameters available for control of the application are described. They are as follows:

- i <input\_file>** Sets the input file to stream from.
- s <sending\_host>** This parameter can be used to specify the host which is used for streaming.
- r <receiving\_host>** This parameter can be used for specifying the receiving host.
- p <rtp\_port\_video>** Using this option, the RTP port for the video connection can be specified. The RTCP port of the video connection is automatically set to *rtp\_port\_video+1*. For the audio stream, the ports for RTP and RTCP are set to *rtp\_port\_video+2* and *rtp\_port\_video+3*, respectively.

- f <format>** Sets the format to be used for streaming and receiving. The following formats are available: *h263+\_mpg*, *h263+\_avi*, *amr\_nb* and *amr\_wb*. The default format is *h263+\_mpg*.
- m <modus>** Sets the modus of a connection as described in Section 9.1. The default mode is *audio\_video*. The distinction between audio and video is not applicable for AMR-NB and AMR-WB.
- a <enable/disable>** The adaptation can be enabled or disabled using this parameter. By default, the adaptation is enabled.
- d** Enables daemon mode. In this mode, the application is started without reading commands from *cin*. This mode can be used in scripts to avoid a CPU load of 100 percent.
- t** Enables writing of statistics using the `StatisticAdaptationStrategy`.
- e** Enables evaluation using the `EvaluationAdaptationStrategy`.
- h** Shows a help which describes usage and command line options.



## Chapter 10

# Summary and Future Work

In the first part of this chapter, the outcomes of the presented work are summarized and the achieved objectives are discussed. The second part covers future work and potential enhancements that build on the framework developed in this thesis.

### 10.1 Summary

The goal of this thesis was to integrate adaptation-based Quality of Service features into the Network-Integrated Multimedia Middleware. For this purpose, a development of adequate mechanisms with appropriate interfaces was demanded.

For the purpose of development and evaluation, an appropriate environment was created by using the network emulation tool NIST Net (see Chapter 4). This tool allows for emulation of network conditions that produce predictable and reproducible results. Using NIST Net, the outcome was evaluated and used for development of appropriate techniques for adaptation of video and audio connections.

In order to use adaptation techniques for audio connections, the codecs AMR-NB and AMR-WB were integrated into NMM in Chapter 6. Both codecs are ideally designed for the purpose of adaptation as they support the switching to several bit rate modes. Moreover, they can be used for connections with very low bandwidth. For integration of these codecs, two plug-ins for encoding and decoding were developed which make use of the library libavcodec.

For demonstration and evaluation purposes, RTP/RTCP was chosen. The main reason is that RTCP already provides feedback mechanisms which

can be used for applying adaptation techniques. In order to use the newly integrated audio codecs AMR-NB and AMR-WB, the existing components for streaming multimedia data via RTP were extended to support these codecs. For accessing all RTCP related information available from LIVE555, the existing interface was extended additionally.

For the integration of adaptation techniques into NMM, a framework for adaptation was developed in Chapter 7. During the design stage, particular attention was paid on flexibility, extendability and modularity. Therefore, the framework was divided into three main components. A component for monitoring, a component for adaptation and a component for overall management. By using this modular approach, a very flexible solution was created. It is not restricted to a specific network protocol, codec or adaptation technique.

By means of the several strategies for adaptation, which have been developed in the scope of this work, the possibilities of the developed architecture were demonstrated. In Chapter 8, strategies for both adaptation of audio streams and video streams were developed. Moreover, a strategy for evaluation and a strategy for statistical purposes were developed.

The evaluation shows that the overall quality of video and audio connections can be greatly improved with relatively simple adaptation approaches during congestion. With the use of the developed adaptation techniques, the packet loss ratio can be greatly reduced in the case of network congestion.

Finally, an adaptive streaming application was developed in Chapter 9. Both streaming and receiving of several formats is supported by the application. It is especially designed for presentation purposes, but also contains features which can be useful for evaluation. Therefore, it is expected to be valuable for future extensions and enhancements of the presented framework. The application demonstrates the quality enhancement which can be achieved with the developed strategies for adaptation. Moreover, the application shows how the framework can be integrated easily into an existing application.

## 10.2 Enhancements and Future Work

The framework developed in the scope of this thesis is the basis for further development of adaptation services in the Network-Integrated Multimedia Middleware. This section will give some ideas of future enhancements and possible applications.

The codecs utilized in the scope of this work are ideally applicable in an adaptive videophone application. AMR-NB and AMR-WB are designed

for voice transmission with low data rates. The H263+ codec is optimized for low data rates and performs especially well with relatively low motion. Therefore, a possible videophone application can use the developed adaptation techniques for providing a higher quality of audio and video in the case of congestion. In order to ensure an acceptable audio quality, a strategy that deactivates the video stream under very bad network conditions would be reasonable as well.

The Adaptation-Strategies developed in the scope of this thesis uses relatively simple approaches for adaptation. With further evaluation these maybe refined. Furthermore, an extension to the RTCP reports, a so-called profile specific extension, can be used for better or earlier detection of problems related to the current network condition.

Of course, the development of further adaptation techniques is desirable. For example, a strategy which uses statistics gathered from the wireless networking device could detect WLAN-roaming. Moreover, the signal strength of the WLAN could be taken into account for adaptation purposes.

For stream switching, NMM contains already a seamless handover service for sink nodes. By extending this functionality for source nodes, the stream switching could be harnessed for adaptation purposes. For instance, if the available bandwidth of a connection drops and no further adaptation with the current stream is possible, this service could be used to utilize a stream which uses a different codec with lower data rates.

In order to be able to use adaptation for TCP connections, monitors that estimate the current performance of a TCP connection could be developed. For example, these estimations may be based upon the TCP window size or the number of required retransmissions.

In different scenarios, different streams are more or less important. Often, this depends on the content. As described above, the audio stream may be more valuable than the video stream in a videophone application. While streaming a soccer match, however, the video is considered to be more important than the commentary. Therefore, an interface could be developed which allows for prioritization of different streams by the user.

In order to handle all NMM-Streams globally, a list of all Transport-Strategies that are using a certain network interface could be handled by the Server-Registry. Furthermore, streams that are sharing a network interface in a part of the flow graphs could be grouped together. By introducing a policy of prioritization, each group could be handled accordingly in the case of packet loss.



# List of Figures

2.1	Generic analog to digital encoder . . . . .	6
2.2	Alias signal resulting from a sampling frequency below the Nyquist-Shannon Criterion . . . . .	7
2.3	Sampling rates and typical usage for digital audio. . . . .	7
2.4	Digitalization of an analog signal using 4 bit quantization. . . . .	8
2.5	AMR bit rate modes . . . . .	11
2.6	A group of pictures with related predictions. . . . .	12
2.7	A group of pictures in coding order. . . . .	13
2.8	The layers of the OSI reference model. . . . .	14
2.9	Progression of the TCP congestion window with idealized slow start. . . . .	15
2.10	Structure of a TCP packet. . . . .	16
2.11	Structure of a UDP packet . . . . .	17
2.12	The RTP header . . . . .	18
2.13	Structure of an RTCP SR packet . . . . .	20
2.14	Structure of an RTCP RR packet . . . . .	22
3.1	Constraints of different networking applications. . . . .	24
3.2	Decision zones used in [7] . . . . .	27
3.3	Different layers for scalable video coding used in [44]. . . . .	29
3.4	Scheme of the architecture used in [79]. . . . .	29
3.5	Scheme of the adaptation logic used in [58]. . . . .	30
3.6	Region of interest coding used in [3] . . . . .	31
4.1	Schematic illustration of the setup used for development and evaluation. . . . .	36
4.2	The graphical user interface of NIST Net. . . . .	40
5.1	A simple NMM flow graph . . . . .	46
5.2	Types of nodes distinguished in NMM . . . . .	48
5.3	State machine of NMM. . . . .	49
5.4	Generation of events by calling an interfaces method. . . . .	53

6.1	Class dependencies illustrating the connection of NMM to LIVE555 . . . . .	61
6.2	Class dependencies illustrating the connection of NMM to LIVE555 with modifications for AMR streaming . . . . .	62
7.1	Abstract issues that must be considered for adaptation. . . . .	68
7.2	Integration of the main components for adaptation (case 1). . . . .	70
7.3	Integration of the main components for adaptation (case 2). . . . .	70
7.4	Connections between the main components of the developed framework. . . . .	71
7.5	Class diagram of the class <code>Adaptor</code> . . . . .	73
7.6	Determination of nodes that allow for adaptation. . . . .	74
7.7	Sequence diagram: Registration of an <code>Adaptation-Strategy</code> at <code>QoS-Monitors</code> . . . . .	76
7.8	<code>QoSMonitor</code> and derived classes for RTP. . . . .	77
7.9	Sequence diagram: Handling of incoming RTCP reports. . . . .	78
7.10	Class diagram of the class <code>AdaptationStrategy</code> . . . . .	80
7.11	Class diagram of the class <code>QoSReport</code> . . . . .	81
7.12	Class diagram of the class <code>CompositeAdaptationStrategy</code> . . . . .	83
7.13	Class diagram illustrating the relationship between all components of the framework. . . . .	84
8.1	Illustration of adaptation delay . . . . .	87
8.2	Evaluation results for AMR-NB streaming without adaptation (Scenario 1). . . . .	90
8.3	Evaluation results for AMR-NB streaming without adaptation (Scenario 2). . . . .	91
8.4	Evaluation results for AMR-WB streaming without adaptation (Scenario 1). . . . .	93
8.5	Evaluation results for AMR-WB streaming without adaptation (Scenario 2). . . . .	94
8.6	Evaluation results for AMR-NB streaming with adaptation (Scenario 1). . . . .	96
8.7	Evaluation results for AMR-NB streaming with adaptation (Scenario 2). . . . .	97
8.8	Evaluation results for AMR-WB streaming with adaptation (Scenario 1). . . . .	98
8.9	Evaluation results for AMR-WB streaming with adaptation (Scenario 2). . . . .	99
8.10	Evaluation results for H263+ streaming without adaptation (Scenario 1). . . . .	101
8.11	Evaluation results for H263+ streaming without adaptation (Scenario 2). . . . .	102

8.12	Evaluation results for H263+ streaming with adaptation (Scenario 1). . . . .	104
8.13	Evaluation results for H263+ streaming with adaptation (Scenario 2). . . . .	105
8.14	Overall number of packets lost in the different scenarios. . . .	106
8.15	Example of four uniformly distributed bit rate and frame rate levels. . . . .	108
9.1	Screenshot of the developed application. . . . .	109
9.2	Class diagram of the class <code>GraphManager</code> . . . . .	111
9.3	Class hierarchy of <code>GraphManager</code> and derived classes. . . . .	111
9.4	Graphs for streaming and receiving of AMR-NB and AMR-WB using a WAV file as input. . . . .	112
9.5	Graphs for streaming and receiving of H263+ video and MP3 audio using an AVI-file as input. . . . .	112
9.6	Graphs for streaming and receiving of H263+ video and MP3 audio using an MPG-file as input. . . . .	113



# Bibliography

- [1] *3rd Generation Partnership Project*.  
<http://www.3gpp.org/>.
- [2] *AMR Audio Decoder Stub - from FFMPEG project*.  
<http://svn.mplayerhq.hu/ffmpeg/trunk/libavcodec/amr.c?view=markup>.
- [3] P. BACCICHET, X. ZHU, and B. GIROD. *Network-Aware H.264/AVC Region-of-Interest Coding for a Multi-Camera Wireless Surveillance Network*. Picture Coding Symposium (PCS-06), Beijing, China, April 2006.
- [4] MASSIMO BERNASCHI, FILIPPO CACACE, RAFFAELE CLEMENTELLI, and LUCA VOLLERO. *Adaptive Streaming on Heterogeneous Networks*. In *WMuNeP '05: Proceedings of the 1st ACM Workshop on Wireless Multimedia Networking and Performance Modeling*, pages 16–23. ACM Press, Montreal, Quebec, Canada, 2005.
- [5] B. BESSETTE, R. SALAMI, R. LEFEBVRE, M. JELINEK, J. ROTOLA-PUKKILA, J. VAINIO, H. MIKKOLA, and K. JÄRVINEN. *The Adaptive Multi-Rate Wideband Speech Codec (AMR-WB)*. In *IEEE Transactions on Speech and Audio Processing*, volume 10, pages 620–636. November 2002.
- [6] STEPHEN BLOTT. *Multimedia Technology, Lecture Notes*, 2005.
- [7] J. BOM, P. MARQUES, M. CORREIA, and P. PINTO. *QoS Control: An Application Integrated Framework*. In *Proceedings of the IEEE Internationale Conference on ATM*. Colmar, France, June 1998.
- [8] JENS BRANDT and LARS WOLF. *A Gateway Architecture for Mobile Multimedia Streaming*. In *European Symposium on Mobile Media Delivery (EuMob06)*. Alghero, Italy, September 2006.
- [9] JOHN F. KOEGEL BUFORD, editor. *Multimedia Systems*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1994.

- [10] INGO BUSSE, BERND DEFFNER, and HENNING SCHULZRINNE. *Dynamic QoS Control of Multimedia Applications based on RTP*. In *Computer Communications*, 19(1):49–58, 1996.
- [11] MARK CARSON and DARRIN SANTAY. *NIST Net: A Linux-based Network Emulation Tool*. In *SIGCOMM Computer Communication Review*, 33(3):111–126, 2003.
- [12] Y. J. CHUNG, J. KIM, and C.-C. J. KUO. *Network-friendly Video Streaming via Adaptive LMS Bandwidth Control*. In A. G. TESCHER, editor, *Proc. SPIE Vol. 3460, p. 448-456, Applications of Digital Image Processing XXI, Andrew G. Tescher; Ed.*, pages 448–456. October 1998.
- [13] *NIST Net WAN Emulation Software Installation and Configuration Note*.  
[http://www.cisco.com/univercd/cc/td/doc/product/sn5000/file\\_eng/fe\\_511/%wafs30/nist.pdf](http://www.cisco.com/univercd/cc/td/doc/product/sn5000/file_eng/fe_511/%wafs30/nist.pdf).
- [14] L. CLINE ET AL. *RTP Payload Format for the 1998 Version of ITU-T Rec. H.263 Video (H.263+)*. RFC 2429 (Proposed Standard), October 1998.
- [15] *CORBA Basics*.  
<http://www.omg.org/gettingstarted/corbafaq.htm>.
- [16] NELE VAN DEN ENDE, PETER VAN DER STOK, REINDER HAAKMA, and MADDY D. JANSE). *Perceived Quality for Transported Video*. In *20th International Symposium on Human Factors in Telecommunication*. Sophia-Antipolis, France, March 2006.
- [17] NICK FEAMSTER and HARI BALAKRISHNAN. *Packet Loss Recovery for Streaming Video*. In *12th International Packet Video Workshop*. Pittsburgh, PA, April 2002.
- [18] *FFMPEG Multimedia System*.  
<http://ffmpeg.mplayerhq.hu/>.
- [19] ERICH GAMMA, RICHARD HELM, RALPH JOHNSON, and JOHN VLISIDES. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley, 2000.
- [20] *Gnuplot*.  
<http://www.gnuplot.info/>.
- [21] *GNU General Public License, 1991*.  
<http://www.gnu.org/copyleft/gpl.html>.

- [22] AUDIO-VIDEO TRANSPORT WORKING GROUP, H. SCHULZRINNE, S. CASNER, R. FREDERICK, and V. JACOBSON. *RTP: A Transport Protocol for Real-Time Applications*. RFC 1889 (Proposed Standard), January 1996. Obsoleted by RFC 3550.
- [23] M. HANDLEY, V. JACOBSON, and C. PERKINS. *SDP: Session Description Protocol*. RFC 4566 (Proposed Standard), July 2006.
- [24] *Iperf*.  
<http://dast.nlanr.net/Projects/Iperf/>.
- [25] *Terms and Definitions Related to Quality of Service and Network Performance Including Dependability*. ITU-T Recommendation E.800, August 1994.
- [26] *Video Coding for Low Bit Rate Communication*. ITU-T Recommendation H.263, January 2005.
- [27] *Jini Architecture Specification*.  
[http://www.jini.org/wiki/Jini\\_Architecture\\_Specification](http://www.jini.org/wiki/Jini_Architecture_Specification).
- [28] D. JOHNSON, C. PERKINS, and J. ARKKO. *Mobility Support in IPv6*. RFC 3775 (Proposed Standard), June 2004.
- [29] *Joost*.  
<http://www.joost.com/>.
- [30] CHARLES KRASIC, JONATHAN WALPOLE, and WU CHI FENG. *Quality-Adaptive Media Streaming by Priority Drop*. In *NOSSDAV '03: Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 112–121. ACM Press, New York, NY, USA, 2003.
- [31] JAMES F. KUROSE and KEITH W. ROSS. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.
- [32] KEVIN LAI and MARY BAKER. *Measuring Bandwidth*. In *Conference on Computer Communications (IEEE Infocom)*, pages 235–245. New York, USA, March 1999.
- [33] *GNU Lesser General Public License, 1999*.  
<http://www.gnu.org/copyleft/lgpl.html>.
- [34] *LIVE555 Streaming Media*.  
<http://www.live555.com/liveMedia/>.

- [35] MARCO LOHSE. *Developing Plug-ins for NMM, April 2003.*  
<http://graphics.cs.uni-sb.de/NMM/current/Docs/plugins.pdf>.
- [36] MARCO LOHSE. *Format Programming for NMM, April 2003.*  
<http://graphics.cs.uni-sb.de/NMM/current/Docs/format.pdf>.
- [37] MARCO LOHSE. *The NMM Interface Definition Language, April 2003.*  
<http://graphics.cs.uni-sb.de/NMM/current/Docs/idl.pdf>.
- [38] MARCO LOHSE. *Network-Integrated Multimedia Middleware, Services, and Applications.* Ph.D. thesis, Department of Computer Science, Saarland University, Germany, June 2005.
- [39] MARCO LOHSE, PHILIPP SLUSALLEK, and PATRICK WAMBACH. *Extended Format Definition and Quality-driven Format Negotiation in Multimedia Systems.* In *Multimedia 2001 – Proceedings of the Eurographics Workshop*, pages 65–74. Springer, 2001.
- [40] XIAOXIANG LU, RAMON ORLANDO MORANDO, and MAGDA EL ZARKI. *Understanding Video Quality and its use in Feedback Control.* Packet Video 2002, Pittsburgh, PA, USA, April 2002.
- [41] JOAN L. MITCHELL, WILLIAM B. PENNEBAKER, CHAD E. FOGG, and DIDIER J. LEGALL, editors. *MPEG Video Compression Standard.* Chapman & Hall, Ltd., London, UK, 1996.
- [42] *Motama GMBH.*  
<http://www.motama.com/>.
- [43] *MPlayer - A Open Source Movie Player.*  
<http://www.mplayerhq.hu/>.
- [44] DIEU THANH NGUYEN and JÖRN OSTERMANN. *Streaming and Congestion Control using H.264/AVC Scalable Video Coding.* In *15th International Packet Video Workshop*, 7(5):749–754, May 2006.
- [45] *NIST Net.*  
<http://www-x.antd.nist.gov/nistnet/>.
- [46] *NIST Net Mailinglist Archive - Patch for the bug related to ports (accessible after registration only).*  
<http://www-x.antd.nist.gov/mailman/private/nistnet/attachments/20060404%/6b3fdc93/kernel-proto.bin>.
- [47] ANDY ORAM. *A Nice Way to Get Network Quality of Service?* Oreilynet, June 2002.

- [48] V. PAXSON and W. STEVENS. *TCP Congestion Control*. RFC 2581 (Standard), April 1999.
- [49] ERIC PETERS. *Design und Implementierung eines verteilten Multimedia Streaming Servers auf Basis der Network-Integrated Multimedia Middleware (NMM)*. Diplomarbeit, Department of Computer Science, Saarland University, Saarbrücken, Germany, 2005.
- [50] J. POSTEL. *User Datagram Protocol*. RFC 768 (Standard), August 1980.
- [51] K. RAMAKRISHNAN, S. FLOYD, and D. BLACK. *The Addition of Explicit Congestion Notification (ECN) to IP*. RFC 3168 (Proposed Standard), September 2001.
- [52] MICHAEL REPPLINGER. *Eine Architektur zur Anbindung und Kontrolle von im Netz verteilten Multimedia-Geräten*. Diplomarbeit, Department of Computer Science, Saarland University, Saarbrücken, Germany, 2003.
- [53] MICHAEL REPPLINGER. *Object Serialization in NMM, May 2003*. <http://graphics.cs.uni-sb.de/NMM/current/Docs/serialization/index.html>.
- [54] MICHAEL REPPLINGER and MARCO LOHSE. *Clic - An Application for Setting up NMM Multimedia Flow Graphs, July 2004*. <http://graphics.cs.uni-sb.de/NMM/current/Docs/clic.pdf>.
- [55] MICHAEL REPPLINGER, FLORIAN WINTER, MARCO LOHSE, and PHILIPP SLUSALLEK. *Parallel Bindings in Distributed Multimedia Systems*. Technical report, Computer Graphics Group, Saarland University, November 2004.
- [56] MICHAEL REPPLINGER, FLORIAN WINTER, MARCO LOHSE, and PHILIPP SLUSALLEK. *Parallel Bindings in Distributed Multimedia Systems*. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCS 2005)*, pages 714–720. IEEE Computer Society, 2005.
- [57] LARISA RIZVANOVIC and GERHARD FOHLER. *The MATRIX - A Framework for Real-time Resource Management for Video Streaming in Networks of Heterogenous Devices*. In *The International Conference on Consumer Electronics 2007*. Las Vegas, USA, January 2007.

- [58] PEDRO M. RUIZ and EMILIO GARCIA. *Adaptive Multimedia Applications to Improve User-Perceived QoS in Multihop Wireless Ad-hoc Networks*. Joint Conference: IEEE International Conference on Wireless LANs and Home Networks (ICWLHN 2002) and IEEE International Conference on Networking (ICN 2002), Atlanta, Georgia, USA, August 2002.
- [59] TOMAS SCHIERL and KATHRIN RÜMMLER. *Adaptives H.264 Internet Streaming*. Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institute Image Processing Dept., June 2004.
- [60] DOUGLAS SCHMIDT, MICHAEL STAL, HANS ROHNERT, and FRANK BUSCHMANN. *Pattern-Oriented Software Architecture*, volume 2, Patterns for Concurrent and Networked Objects. John Wiley and Sons, New York, 2000.
- [61] H. SCHULZRINNE and S. CASNER. *RTP Profile for Audio and Video Conferences with Minimal Control*. RFC 3551 (Standard), July 2003.
- [62] H. SCHULZRINNE, S. CASNER, R. FREDERICK, and V. JACOBSON. *RTP: A Transport Protocol for Real-Time Applications*. RFC 3550 (Standard), July 2003.
- [63] H. SCHULZRINNE, A. RAO, and R. LANPHIER. *SDP: Session Description Protocol*. RFC 2326 (Proposed Standard), April 1998.
- [64] DORGHAM SISALEM and HENNING SCHULZRINNE. *The Direct Adjustment Algorithm: A TCP-Friendly Adaptation Scheme*. In JON CROWCROFT, JAMES ROBERTS, and MIKHAIL I. SMIRNOV, editors, *Quality of Future Internet Services (QofIS)*, volume 1922 of *Lecture Notes in Computer Science*, pages 68–79. Springer, Berlin, Germany, 2000.
- [65] RALF STEINMETZ and KLARA NAHRSTEDT. *Multimedia: Computing, Communications and Applications*. Prentice Hall, Upper Saddle River, 1995.
- [66] T-MOBILE. *Press Release*.  
[http://www.t-mobile.de/downloads/unternehmen/pressemitteilungen/06-10-24\\_amr-wideband.pdf](http://www.t-mobile.de/downloads/unternehmen/pressemitteilungen/06-10-24_amr-wideband.pdf).
- [67] *Vudu*.  
<http://www.vudulabs.com/>.

- [68] PATRICK WAMBACH. *Formatdefinition und Formatverhandlung von Multimedia-Geräten*. Diplomarbeit, Department of Computer Science, Saarland University, Saarbrücken, Germany, 2001.
- [69] *Wikipedia, the free encyclopedia - Adaptive multi-rate compression*.  
[http://en.wikipedia.org/wiki/Adaptive\\_multi-rate\\_compression](http://en.wikipedia.org/wiki/Adaptive_multi-rate_compression).
- [70] *Wikipedia, the free encyclopedia - Distributed Component Object Model (DCOM)*.  
[http://en.wikipedia.org/wiki/Distributed\\_Component\\_Object\\_Model](http://en.wikipedia.org/wiki/Distributed_Component_Object_Model).
- [71] *Wikipedia, the free encyclopedia - Digital Audio*.  
[http://en.wikipedia.org/wiki/Digital\\_Audio](http://en.wikipedia.org/wiki/Digital_Audio).
- [72] *Wikipedia, the free encyclopedia - Nyquist-Shannon sampling theorem*.  
<http://de.wikipedia.org/wiki/Nyquist-Shannon-Abtasttheorem>.
- [73] *Wikipedia, the free encyclopedia - OSI model*.  
[http://en.wikipedia.org/wiki/OSI\\_model](http://en.wikipedia.org/wiki/OSI_model).
- [74] *Wikipedia, the free encyclopedia - Timbre*.  
<http://en.wikipedia.org/wiki/Timbre>.
- [75] FLORIAN WINTER. *Development and Evaluation of Communication Strategies for the Network-Integrated Multimedia Middleware (NMM)*. Bachelor's Thesis, Department of Computer Science, Saarland University, Saarbrücken, Germany, 2006.
- [76] D. WU, Y.T. HOU, W. ZHU, Y.Q. ZHANG, and J.M. PEHA. *Streaming Video over the Internet: Approaches and Directions*. In *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):282–300, March 2001.
- [77] *xine - A Free Video Player*.  
<http://xinehq.de/>.
- [78] MEI YANG, YAN HUANG, JAIME BAE KIM, MEEJEONG LEE, TATSUYA SUDA, and DAISUKE MATSUBARA. *An End-to-End QoS Framework with On-Demand Bandwidth Reconfiguration*. In *Conference on Computer Communications (IEEE Infocom)*. Hong Kong, March 2004.
- [79] Q. ZHANG, Q. GUO, Q. NI, W. ZHU, and Y.Q. ZHANG. *Sender-Adaptive and Receiver-Driven Layered Multicast for Scalable Video Over the Internet*. In *IEEE Transactions on Circuits and Systems for Video Technology*, 15(4):482–495, April 2005.



# Appendix A

## Acronyms

<b>3GPP</b>	3rd Generation Partnership Project
<b>ACELP</b>	Algebraic Code Excited Linear Prediction
<b>AIMD</b>	Additive Increase Multiplicative Decrease
<b>AMR-NB</b>	Adaptive-Multi-Rate Narrowband
<b>AMR-WB</b>	Adaptive-Multi-Rate Wideband
<b>API</b>	Application Programming Interface
<b>CD</b>	Compact Disc
<b>CNG</b>	Comfort Noise Generation
<b>COS</b>	Class of Service
<b>DCOM</b>	Distributed Component Object Model
<b>DTX</b>	Discontinuous Transmission
<b>DVD</b>	Digital Versatile Disc
<b>ECN</b>	Explicit Congestion Notification
<b>ETSI</b>	European Telecommunications Standards Institute
<b>FIFO</b>	First In, First Out
<b>FTP</b>	File Transfer Protocol
<b>GOP</b>	group of pictures
<b>GPL</b>	General Public License

<b>GSM</b>	Global System for Mobile Communications
<b>IESG</b>	Internet Engineering Steering Group
<b>IPTV</b>	Internet Protocol Television
<b>ISDN</b>	Integrated Services Digital Network
<b>ISO</b>	International Organization for Standardization
<b>ITU-T</b>	International Telecommunication Union Telecommunication Standardization Sector
<b>LAN</b>	Local Area Network
<b>LGPL</b>	Lesser General Public License
<b>NMM</b>	Network-Integrated Multimedia Middleware
<b>MMS</b>	Multimedia Messaging Service
<b>MPEG</b>	Moving Picture Experts Group
<b>MP3</b>	MPEG-1 Audio Layer 3
<b>NTP</b>	Network Time Protocol
<b>QoS</b>	Quality of Service
<b>PCM</b>	Puls-Code-Modulation
<b>RFC</b>	Request for Comments
<b>RSVP</b>	Resource Reservation Protocol
<b>RTP</b>	Real-Time Transport Protocol
<b>RTCP</b>	Real-Time Transport Control Protocol
<b>RTSP</b>	Real-Time Streaming Protocol
<b>SDP</b>	Session Description Protocol
<b>SNR</b>	Signal-to-noise ratio
<b>SSID</b>	Service Set Identifier
<b>SVN</b>	Subversion
<b>TCP</b>	Transmission Control Protocol

---

<b>UDP</b>	User Datagram Protocol
<b>UMTS</b>	Universal Mobile Telecommunication System
<b>URL</b>	Uniform Resource Locator
<b>VAD</b>	Voice Activity Detection
<b>WLAN</b>	Wireless Local Area Network
<b>VoIP</b>	Voice over Internet Protocol
<b>WWW</b>	World Wide Web