DOCTOR OF SCIENCES

Video rate adaptation based on QoS and QoE information in wireless access networks

Toma, George

Award date:
2012

Awarding institution:
University of Namur

Facultés Universitaires Notre-Dame de la Paix
Faculté d'Informatique
Namur, Belgique

Video Rate Adaptation Based on QoS and QoE Information in Wireless Access Networks

George Toma

June 2012

Thèse présentée en vue de l'obtention du grade de Docteur en Sciences
(orientation Informatique)

Ph.D. Supervisor:
Professor Laurent Schumacher,
University of Namur, Belgium.

Ph.D. Committee:
Professor Christophe De Vleeschouwer,
Université Catholique de Louvain, Belgium.

Professor Jean-Noël Colin,
University of Namur, Belgium.

Opponents:
Professor Naji Habra,
University of Namur, Belgium.

Professor Jean-Paul Leclercq,
University of Namur, Belgium.

Dr. Danny De Vleeschauwer ,
Strategic Network Analyst for Corporate CTO, Alcatel-Lucent, Belgium.

Professor Raouf Hamzaoui,
De Montfort University, UK.

The public Ph.D. defense was held on the 7th June 2012 at the University of Namur.

Imprimé en Belgique
ISBN : 978-2-87037 -762-8
Dépôt légal: D / 2012 / 1881 / 24

Abstract

Streaming applications are becoming more and more popular in the mobile world, especially with the new developments in wireless data networks. Promises of maximum throughput speeds of a 100Mbps along with the appearance of multimedia tablets opened the path to a diverse range of streaming services. Signal degradation and user mobility make wireless networks a challenging environment for real-time applications like streaming, due to frequent bandwidth variations. Hence, to assure a continuous service and to maximise the user experience, it is necessary to automatically adapt the streaming rate of the session, based on the network state. This report presents a layered QoS and QoE based rate adaptation algorithm that tackles both delivery and playback problems. It uses measurements of the RTT variation, an original probing technique and playback quality reports to estimate user experience and react as fast and as accurate as possible. The adaptation strategy parameters have been tuned for optimal performance in different wireless environments, like WiFi, WiMAX and LTE.

Before anything else I would like to address the first thank you words to my promoter, Laurent Schumacher, who offered me the chance to discover the academic world as a researcher. He has the great merit for guiding and motivating me to put together the pieces that led to this Ph.D. thesis. Thank you!

I would also like to express my appreciation to Christophe De Vleeschouwer for his constructive ideas during my research period, as well to the members of the jury Raouf Hamzaoui, Jean Noël Colin, Danny De Vleeschauwer and Naji Habra for investing their time to read my work and for their interesting comments that greatly improved the quality of this material.

Special thanks I would like to address to Ivan Alen Fernandez who helped me with the implementation of my algorithm and to Stephen Rainey who offered to correct my rough English for only a couple of beers.

Si nu in ultimul rand, as dori sa multumesc familiei pentru sustinere si incurajari, colegilor de la facultate si prietenilor din Namur, "les bagabonts", care mi-au asigurat un mediu cultural elevat, citind impreuna multiple articole, acasa sau la biblioteca.

George, Iunie 2012

Contents

List of Figures

List of Tables

List of Acronyms

ADU Application Data Unit

AIMD Additive-Increase/Multiplicative-Decrease

AMP Adaptive Media Playout

APP Application specific packet

BCC Binomial Congestion Control

BER Bit Error Rate

bps bits per second

BSS Bit-Stream Switching

CDF Cumulative Distribution Function

CDMA2000 Code Division Multiple Access 2000

CDN Content Distribution Network

CNAME Canonical Name

DASH Dynamic Adaptive Streaming over HTTP

DCCP Datagram Congestion Control Protocol

DCH Dedicated Transport Channel

DCT Discrete Cosine Transform

DSIS Double-stimulus Impairment Scale

E2E End to End

ECN Early Congestion Notification

EDGE Enhanced Data rates for GSM Evolution

FDFU Fast bit-rate Decrease Fast bit-rate restore Up

FDSU Fast bit-rate Decrease Slow bit-rate restore Up

FSM Finite State Machine

FR-(VQM) Full Reference

GPRS General Packet Radio Service

GSM Global System for Mobile Communications, originally Groupe Spécial Mobile

HAS HTTP Adaptive Streaming

HSDPA High-Speed Downlink Packet Access

HTTP Hypertext Transfer Protocol

HVS Human Vision System

IANA Internet Assigned Numbers Authority

IIAD Inverse-Increase/Additive-Decrease

ITU International Telecommunication Union

LTE Long Term Evolution

MOS Mean Opinion Score

MPEG Moving Picture Experts Group

MSE Mean Squared Error

NADU Next Application Data Unit

NAL Network Abstraction Layer

NAT Network Address Translation

NetEm Network Emulator

NR No-Reference

NSN Next Sequence Number

NUN Next Unit Number

OS Operating System

OSI Open Systems Interconnection

PEVQ Perceptual Evaluation of Video Quality

PC Personal Computer

PSNR Peak Signal-to-Noise Ratio

PSQA Pseudo-Subjective Quality Assessment

QoC Quality of Content

QoE Quality of Experience

QoS Quality of Service

RAN Radio Access Network

RFC Request For Comments

RLE Run Length Encoding

ROI Regions Of Interest

RR Receiver Report

RR-VQM Reduced-Reference

RTCP Real-Time Transport Control Protocol

RTP Real Time Protocol

RTSP Real Time Streaming Protocol

RTT Round Trip Time

SDES Source Description

SDP Session Description Protocol

SR Sender Report

SS Single-stimulus

SSIM Structural Similarity Index Method

SSMR Single-stimulus with Multiple Repetitions

SVC Scalable Video Coding

TCP Transmission Control Protocol

TFRC TCP Friendly Rate Control

TTL Time To Live

UDP User Datagram Protocol

UE User Equipment

UMTS Universal Mobile Telecommunications System

VCR Video Cassette Recording

VQM Video Quality Metric

VoIP Voice over IP

VSSIM Video Structural SIMilarity

WAP Wireless Application Protocol

WiMAX Worldwide Interoperability for Microwave Access

Part I

Introduction and Motivations

Chapter 1

Introduction

1.1 Media Streaming History

The notion of media streaming is related to the appearance of the packet switched networks and the Internet and basically refers to some type of visual (audio) content that is continuously received and viewed (listened to) by an end-user while being delivered by a content provider over a communication channel. The streaming process is similar to typical TV broadcasting solutions, in the sense that the received content is not saved on the client side and the watching (listening) process begins shortly after the data flow has reached the player. There are however notable differences like the transport channel which is usually shared with other types of communication and the best effort delivery model where data will be delivered to its destination as soon as possible, but with no commitment as to bandwidth or latency. Another difference from classical broadcasting systems is the fact that a media stream can be targeted to only a group of users, as in the case of multicast or only to a single user, as in the case of unicast, which later became very popular over the Internet with the appearance of websites like YouTube. A typical unicast stream is shown in Fig. 1.1.

Because the transport and the presentation of the media happen at the same time or with a very small delay between them, streaming can be regarded as real-time traffic, which best effort networks are not well suited for. This is why, each media player has a buffer (usually called jitter buffer or play-out buffer) where it stores a few seconds of the received video before displaying it to the viewer to accommodate late or possible lost frames without interrupting the playback.

In 1996 the Real Time Protocol (RTP) Request For Comments (RFC) was published in order to help transmitting real-time data over unicast

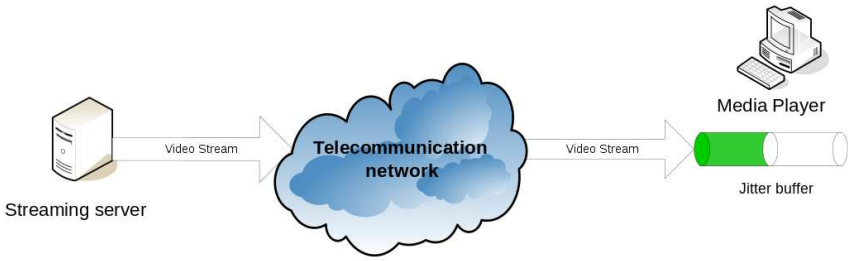

Figure 1.1: Typical streaming chain

or multicast network services. It soon became the standard application protocol for media streaming and video conferencing. It is independent of the underlying Open Systems Interconnection (OSI) layers, so it can be used on top of other transport protocols, like Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) [1].

In the last couple of years, a new technique that uses the Hypertext Transfer Protocol (HTTP) became more and more popular for video streaming. It is called progressive download since small parts of the video are downloaded piece by piece and stored on the client side and the playback can start before the whole file is saved. Although it is not a pure streaming method since the video file is stored at the client side in the end, it mimics well enough a streaming process. This method presents a couple of advantages over the classic RTP streaming:

- HTTP servers can be used instead of more expensive and elaborate streaming servers
- Media stream crosses all middleboxes (e.g firewalls), because the traffic is HTTP.
- There is no packet loss because HTTP is always used on top of TCP protocol
- It can use existing Internet infrastructures like Content Distribution Network (CDN), caches

However, because it uses TCP as the transport protocol, it is not very well suited for scenarios where some form of interactivity is needed (video conferences for example), where end-to-end delay is very important, or for channels that have variable throughput.

1.2 Mobile Communications Evolution

With the evolution and the increased popularity of the Internet, telecom operators tried to extend their portfolio beyond voice communications, starting with email access and a simplified form of web browsing that uses Wireless Application Protocol (WAP). Data transfer was available using the circuit-switched infrastructure but did not grow very popular among Global System for Mobile Communications, originally Groupe Spécial Mobile (GSM) users. With the appearance of the first packet-switched services for mobile phones, such as General Packet Radio Service (GPRS) and later Enhanced Data rates for GSM Evolution (EDGE), that can achieve a theoretical maximum throughput of 480kbps but an average of 180kbps as stated in [5], short browsing sessions were possible. But the new speeds were still not sufficient for real multimedia services. The upgrade to the 3rd generation networks like Universal Mobile Telecommunications System (UMTS) and Code Division Multiple Access 2000 (CDMA2000) brought an increase in the maximum available bandwidth up to 2Mbps and opened the road for new services like video streaming. Furthermore, with the introduction of High-Speed Downlink Packet Access (HSDPA) and HSDPA+ speeds up to 21Mbps can be achieved which are better suited for rich multimedia services.

However, with the latest access technologies that are just emerging, like Worldwide Interoperability for Microwave Access (WiMAX) and Long Term Evolution (LTE) with speeds of up to 100Mbps in certain conditions, combined with the huge number of internet tablets sold all over the world, multimedia traffic in mobile networks is expected to experience an impressive increase in the near future.

1.3 Wireless Networks Drawbacks

Along with the great opportunities brought by mobile communications, come new challenges that need to be addressed by operators and service providers. Compared to the wired medium, wireless transmissions suffer from signal degradation due to free path loss and propagation loss (diffraction, scattering, slow-fast fading) which eventually means lower achievable throughput for the user. On top of this, the mobility adds new problems like hand-overs and high variation of the throughput at high speeds.

Streaming sessions are the most affected by bandwidth variation since the media stream needs to be played back at a specific constant rate. If the network can not support that specific rate, video frames encapsulated

in network packets will be delayed and might miss their playback time. In such cases, the user will experience image degradation, jerkiness or video re-buffering when the jitter buffer gets empty and playback completely stops.

Of course, one solution would be to send from the beginning a video encoded at a lower bit-rate that will have a smaller chance of being affected by bandwidth reduction, but in such cases the video quality will be lower and it will not benefit from the higher bandwidth potentially available under better circumstances.

The solution would then be to send to the client a video adapted to his/her own current network conditions to maximize his/her experience. For that reason, the following items are needed:

- a way to measure or to estimate the quality of the video received at the client;
- a way to send this metric back to the server;
- take a decision on the server side.

1.4 Motivation

Bit-rate adaptation for video streaming is not a new research topic, but although a lot of solutions have been proposed, only a few simple ones have been implemented in the commercial streaming servers. This is because common media players only support the basic Real-Time Transport Control Protocol (RTCP) standard and most of the rate adaptation strategies present in the literature need additional reporting capabilities from the player. Typical adaptation schemes measure network parameters [6] or estimate the buffer occupancy of the player [7] to decide when to change the video quality delivered by the server. The problem is that in this case the adaptation mechanism can detect some of the network related issues, but cannot tell with accuracy how much playback may be affected. This is especially true in wireless networks where packet corruption or packet loss is common and cannot be predicted. On the other hand, small mobile devices have limited capabilities (low resolution, lower processing power) and the video quality can suffer even if the network parameters are optimal.

Consequently, it would be best to design a system that could detect both network related problems and the video quality received at the client side. So far we are not aware of a complete solution that proposes stream adaptation based on those parameters, which would greatly im-

prove the accuracy of the algorithm and therefore the research question of this thesis is:

"How to design and implement an adaptive streaming solution that takes into consideration network parameters and the video quality measured at the client? Once the algorithm is defined, which are the main parameters that need to be tuned for an efficient operation? What are the necessary feedback mechanisms that need to be used?"

1.5 Thesis Objectives

The objectives of this thesis are to design, implement and evaluate an adaptive streaming algorithm that uses network parameters and video quality measurements at the client side to take decisions regarding the quality of the video that will be streamed.

We propose to develop a layered adaptive streaming platform that can be used with:

- popular PC media-players like VLC, QuickTime
- a proprietary player that offers video quality reporting capability

The second contribution consists in the use of a new network probing mechanism, by sending in advance the video frames that would have to be transmitted anyway, creating an additional load on the network. By using RTP packets as probing data, the server will not send unnecessary information over the network and the media client will not need to be modified.

1.6 Structure of the Dissertation

The present dissertation is structured as follows:

Part II is divided in two chapters and describes the methods that are available for measuring network parameters and video quality. **Chapter 2** presents the parameters that are used to measure the network state during streaming sessions along with some modifications necessary for the algorithm later proposed in **Chapter 8**.

Chapter 3 shows the current techniques used for video quality measurement along with some experiments performed to test the usability of those methods in a streaming scenario.

Part III is split into three chapters and depicts the mechanisms that are available for transporting information from the player to the streaming server. **Chapter 4** and **Chapter 5** present the feedback

methods for network parameters and video parameters respectively, while **Chapter 6** motivates the choices made for the proposed solution.

Part IV contains four chapters and presents the proposed algorithm along with the tests performed to tune some of its parameters. In **Chapter 7** there is a discussion about the current adaptive streaming solutions present in the literature and commercial products, showing where the proposed algorithm is currently situated. **Chapter 8** describes the general characteristics of the layered adaptive streaming algorithm which is the focus of this dissertation. **Chapter 9** aims at determining the thresholds for the measured parameters used to reduce the video quality when a problem is detected, while **Chapter 10** presents the design choices of the probing technique along with the tests done to determine the limits for the parameters used to decide an increase in the video quality.

Part V evaluates the performance of the proposed algorithm and contains validation tests in three different wireless networks.

Finally, in **Part VI** some conclusions are drawn and future developments are discussed

1.7 Publications

The research work performed for this thesis has lead to the following peer reviewed publications:

- George Toma and Laurent Schumacher, *"Measuring the QoE of Streaming Sessions in Emulated UMTS Rel'99 Access Networks"* presented at SCVT2008. In this article we describe a method to measure video quality for streamed videos and show the results after testing three commercial streaming servers.

- George Toma, Laurent Schumacher and Christophe De Vleeschouwer, *"Offering Streaming Rate Adaptation to Common Media Players"*, presented at HotMD2011 workshop. The paper presents an adaptive streaming algorithm that uses only the RTCP Receiver Reports as feedback mechanism to determine the network state. In this way, the quality of the streamed video can be adapted even when common media players are used, like VLC or GStreamer.

- Laurent Schumacher, Gille Gomand and George Toma, *"Performance Evaluation of Indoor Internet Access over a Test LTE Mini-Network"* presented at WPMC 2011. In the article we present the results obtained from experiments performed in a mini LTE network.

- Ivan Alen Fernandez, Christophe De Vleeschouwer, George Toma and Laurent Schumacher, *"An Interactive Video Streaming Architecture Featuring Bit-rate Adaptation"* under revision to be published in the Journal of Communications(JCM, ISSN 1796-2021). The article presents an interactive streaming platform which integrates an adaptive streaming algorithm that improves both received video quality and the reactivity of the streaming system to user interactions.

Part II

Streaming Quality Evaluation

Chapter 2

Quality of Service

A general definition of Quality of Service (QoS) is given by International Telecommunication Union (ITU) in [8], as a "collective effect of service performances which determine the degree of satisfaction of a user of a service". When used in the context of packet switched telecommunication systems, QoS refers to the capability of the network to guarantee that certain parameters of a data flow respect the imposed level of performance, as suggested in [9] and [10]. This is especially needed when there are several concurrent flows on the same communication channel and the network capacity is insufficient.

QoS can be managed at different OSI levels:

- at data link - QoS management functions for UMTS bearer service in the control and user plane, or several service classes in WiMax;
- at network layer - DiffServ which essentially improves the performance of some data flows by classifying and shaping the traffic in different queues. By using queueing disciplines, the priority of specific packets can be raised or decreased so particular services will have access to more network resources;
- at application layer, where the application itself can regulate the data flow.

Such mechanisms have been implemented on a relatively limited scale, mostly inside corporate, academic or mobile cell networks. Since assuring End to End (E2E) QoS was impossible due to the implementation cost and scalability problems, the Internet is built mostly on equipment that operates on a best effort basis, without any other control mechanism. When possible, a certain level of QoS can be maintained at the application level by monitoring the transmission and adjusting

specific parameters in the application so they are better suited for the current transport channel.

2.1 QoS Parameters

Every packet-switched data flow is characterized by quality indicators that can be measured and have to be maintained in certain limits dictated by the type of application that uses that transport service. These factors may be accurately measured or just estimated with the error range depending on the application, the type of traffic, the protocols used or the architectural design of the network. The most relevant parameters are:

- **Throughput (transfer rate)** represents the amount of data transported by the channel during a fixed time period, usually measured in bits per second (bps). Maximum throughput, or bandwidth as often used in the literature, represents the maximum possible quantity of data that can be transmitted under ideal circumstances and in some cases this number is reported as equal to the channel capacity [11]. Usually the throughput depicts the total amount of transported data, including all protocol overheads. To define the amount of useful data transported by the network, the concept of goodput is introduced, which basically measures the amount of application data transferred. Considering the definitions given above, the available bandwidth at time `t` can be defined as the difference between the channel capacity at time `t` and current throughput or link load at time `t`. When the two have equal values, the available bandwidth will be zero, a situation similar to network congestion. The available bandwidth is time varying because it depends on the link load which is fluctuating on short time-scale, but the capacity can vary as well in wireless networks, especially in a fast moving scenario.

- **Transmission delay** - refers to the propagation time of a packet from the source to the destination and in this case it is called *one way delay*. It always has a minimal value, called the *real latency* [12] which is of physical nature and depends on the characteristics of the transport medium (wireless, copper, fibre) and the distance between the communicating entities. On top of that there is the *induced latency* [12] which is added in several ways:

  - Packet reassembly delay within network devices, so the more hops, the higher the delay

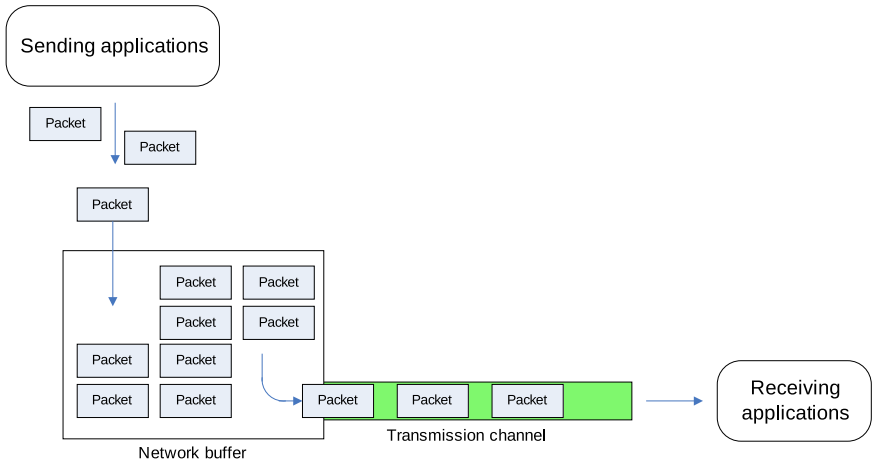

Figure 2.1: A typical network pipe

- Processing delay at end hosts and intermediate routers
- Queueing delay within the network devices

Queueing delay is the most important, first because its contribution to the overall latency value can be the most significant and second because it is the only one that can be reduced through queue management. This type of delay is related to the capacity or the available throughput at the moment it was measured. If we consider the communication channel between two entities as a pipe with a fixed capacity, every packet will spend a relative constant time to pass through the pipe. If the capacity has been reached, packets will have to wait (packets are queued) before entering the pipe so the amount of time to pass from one end to the other increases. This process is represented in Fig. 2.1. In many cases the Round Trip Time (RTT) is used instead of the one way delay because it is easier to measure, but it has two important weaknesses [13]:

- The returning path might be different from the sending one, so the estimated one way delay could have a significant error
- Even if the links are symmetric, different queueing mechanisms might be used for uplink and downlink traffic

- **Delay variation (Jitter)** in the context of computer networks, is defined as the variation of delay over a period of time. It can have

different sources, like different packet assembly times due to different packet sizes, variable propagation delay or varying load level of the network equipment. The latter can introduce a significant amount of jitter in the system, so high delay variation can be a sign of congestion in the network. Jitter is an important QoS metric to take into consideration when designing the play-out buffer for applications that need a constant flow of data like voice or video play-out. The buffer has to be large enough to accommodate maximum delay variation in order to avoid re-buffering. Another advantage of using this metric against absolute delay values is that the same variation can have the equivalent interpretation no matter what is the average delay.

- **Packet loss** results when one or more packets do not reach the destination. However, corrupted packets or fragmented packets that for some reason can not be reassembled at destination can be included in the same category. Another case is when a packet arrives at the destination with a large delay, it can be considered lost as well, especially in real-time applications. In TCP case for example, when the retransmission timer expires, the packet is considered lost.

  Packet loss can be caused by overloaded network equipment that drops packets when congestion is present, or by corrupted and lost packets in wireless environments due to signal degradation. A low level of packet loss may be acceptable in some applications (like video streaming or Voice over IP (VoIP)) but others require reliable delivery with zero losses.

2.2 QoS and Media Streaming

Since media streaming has particular requirements in terms of delay, jitter and losses, QoS plays an important role in assuring a high degree of user satisfaction. It is necessary then, to understand how the metrics presented in the previous section affect the streaming service and what their values for an acceptable performance level could be.

So, what are the conditions of a successful streaming session? First, a user wants to start experiencing the content as soon as possible, which means the play-out delay has to be kept small. This delay is influenced by the play-out buffer size of the player and the transmission delay [14]. Once the playback has started, it has to continue uninterrupted and without image degradation caused by the delivery process. High jitter

can affect the streaming activity while packet loss may deteriorate the displayed frames [15].To avoid re-buffering, the player needs a large buffer size to accommodate possible variations in available bandwidth. One can observe that there is a close connection between the buffer size and the quality of the streaming process, with a trade-off between responsiveness and robustness. Thus, the buffer can be considered a QoS metric for video streaming and the ability to maintain it filled between certain levels is critical for ensuring a continuous playback. For this reason, the throughput on the path between the media server and the client has to be below the total channel capacity, otherwise congestion will occur. This can be obtained through QoS mechanisms, overprovisioning of the network resources or by keeping the media encoding rate below the maximum achievable goodput for the whole duration of the streaming process.

2.3 QoS Parameters in Streaming Applications

When using RTP or HTTP streaming over TCP, congestion detection mechanisms implemented in the TCP protocol will take care of the flow control. But because RTP streaming sessions are usually performed on top of UDP, which does not have a built in congestion control mechanism and because QoS is not implemented on a large scale, it is important to be able to determine the network state at the application layer of the OSI stack. Of course, Datagram Congestion Control Protocol (DCCP) which has congestion control, could be used as the transport protocol as is better suited for real time applications due to the lack of flow control. But as is a relatively new protocol it still lacks large scale stable implementation [16]. However, by measuring the QoS parameters at the application level, the streaming solution will be independent on the lower protocols used for transport. Congestion detection techniques rely on the RTT and packet loss, so a similar approach can be used for streaming applications.

2.3.1 Round Trip Time

Latency, as a sole measurement, can not give reliable information about the state of the network. Nevertheless, delay evolution and especially delay variation can be used to detect a congestion situation. Because it is easier to measure, RTT can be used instead of the one-way-delay. Although some links are very asymmetric in terms of RTT, the variation of the delay will be reflected in the final value. This is important since a fast increase in the RTT suggests that congestion is about to take place

in the network. Because the variation nature of the instantaneous RTT is spiky, two variables will be used to characterize delay evolution, as specified in the computation of the TCP retransmission timer [17]: a smooth RTT and the RTT deviation. The formulas, as given in [17] are as follows:

$$\begin{aligned} &\text{Smooth}_{RTT} \\ &\quad = \quad (1-\alpha) * \text{Smooth}_{RTT} + \alpha * \text{Instant}_{RTT} \qquad (2.1) \\ &\text{Deviation} \\ &\quad = \quad (1-\beta) * \text{Deviation} + \beta * |\text{Instant}_{RTT} - \text{Smooth}_{RTT}| \qquad (2.2) \end{aligned}$$

where the recommended values for α and β are 0.125 and 0.25 respectively [18]. α and β are smoothing factors between 0 and 1. Lower values provide better smoothing and avoid sudden changes as a result of a very high or a very low RTT. Conversely, more measurements will be necessary before detecting a significant increase in the smooth RTT. Higher values (closer to 1) make the smooth RTT change more quickly in reaction to abrupt variations in measured RTT, but delay spikes will not be levelled.

Because it is interesting to know if the Deviation is positive or negative (positive Deviation suggests a possible congestion, while negative Deviation signals the end of congestion), the absolute value in (2.2) is replaced with the real value:

$$\begin{aligned} \text{Deviation} \quad = \quad & (1-\beta) * \text{Deviation} \\ & +\beta * (\text{Instant}_{RTT} - \text{Smooth}_{RTT}) \qquad (2.3) \end{aligned}$$

Fig 2.2 shows an example of the RTT evolution along with the *Deviation* and $Smooth_{RTT}$ during congestion period. The Network Emulator (NetEm) Linux module is used to reduce the network bandwidth to the streaming rate for a limited time period, while α and β have the standard values. The RTT values increase abruptly when a bandwidth reduction is applied and quickly decrease when the limitation is removed. RTT samples were collected every 5 seconds.

The convergence time of the $Smooth_{RTT}$ to the instant RTT varies with the values given to α, as can be seen in Fig 2.3.

Fig. 2.4 plots only the RTTs that were measured when the bandwidth limitation was removed. It can be seen that when α takes the standard value, the convergence time is almost twice as high compared to the case when $\alpha = 0.25$. In such cases, short-term RTT variations will not be reflected in the $Smooth_{RTT}$ evolution. On the other case, when α takes values closer to 1, the $Smooth_{RTT}$ follows closely the evolution of the

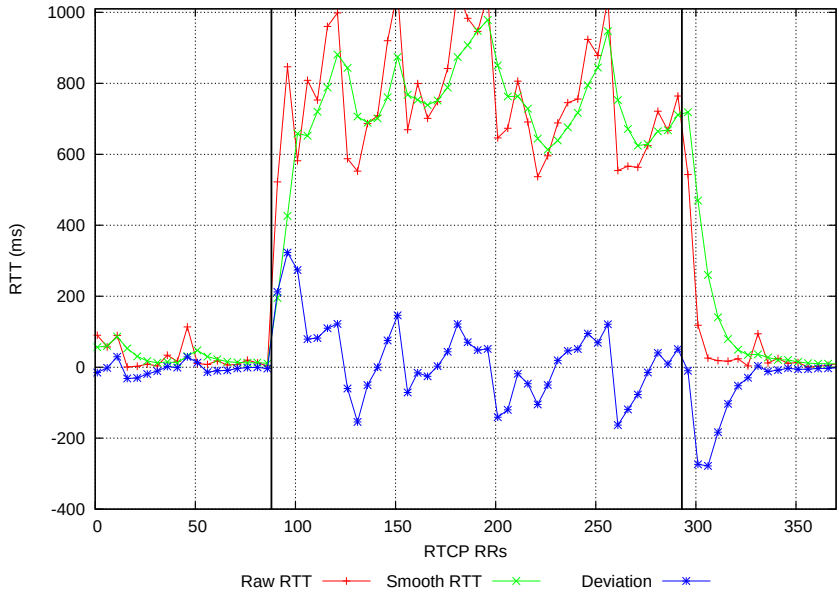

Figure 2.2: RTT evolution in a congested network. Bandwidth reduction applied at sample 88 and removed at sample 293, marked by vertical lines.

instant RTT which is not desired since random delay spikes should not be taken into consideration. In mobile networks fast bandwidth variations are possible, which could generate brief RTT increases, so an α of 0.25 should be used instead of the standard value.

2.3.2 Packet Loss Ratio

Packet loss is easy to determine in either HTTP or RTP based streaming approaches. TCP already contains the necessary components to detect if packets got lost through positive acknowledgement with retransmission techniques and sequence numbers. When RTP protocol is used, each RTP packet contains a sequence number for each RTP data packet sent and is to be used by the receiver to detect packet loss. Compared to TCP, RTP does not interfere in the transmission process when packets are lost; it is left to the application to decide whether it is necessary to take any action. Packet loss can be expressed either as a cumulative number since the beginning of the streaming session or as a percentage representing the ratio between the number of lost packets and the total number sent, but should be used what is best suited for a specific application.

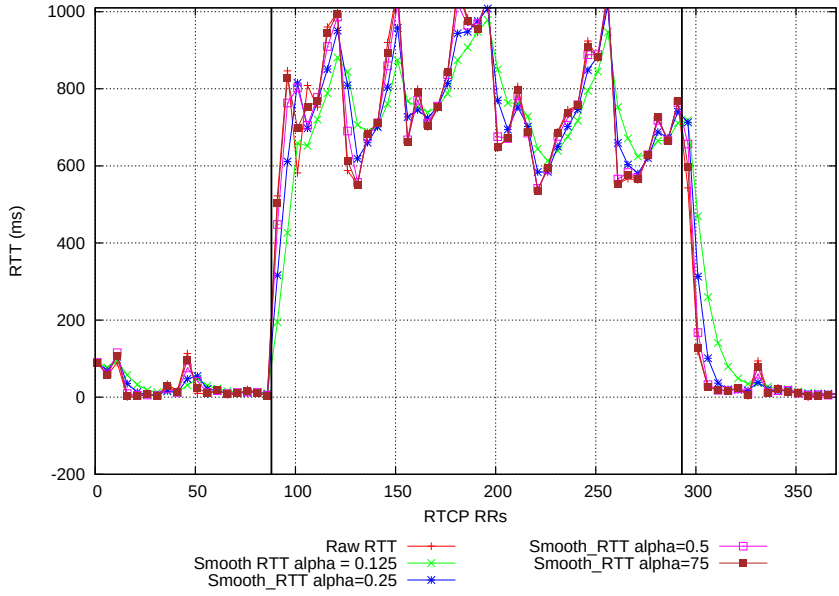

Figure 2.3: Smooth RTT evolution in a congested network with different values for α. Bandwidth reduction applied at sample 88 and removed at sample 293, marked by vertical lines.

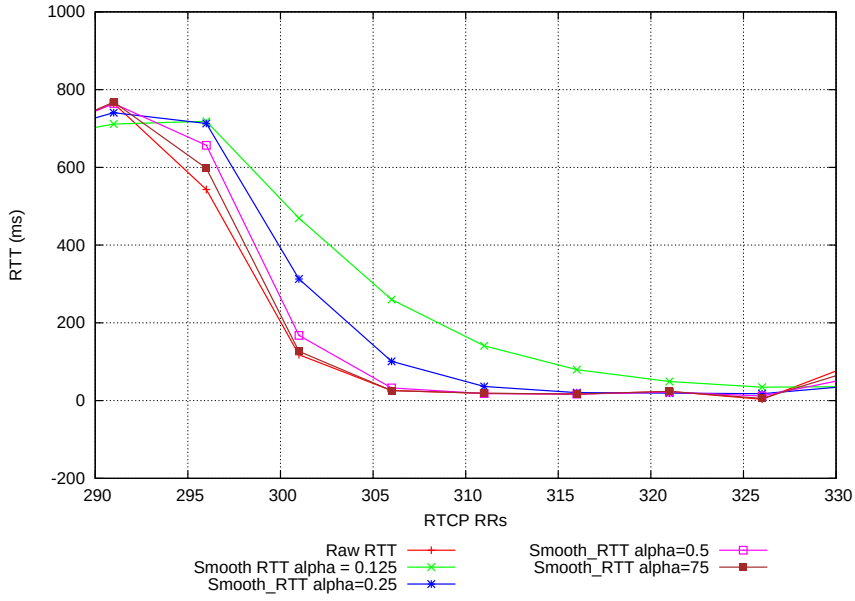

Figure 2.4: Zoom on the last RTT samples from Fig. 2.3

Chapter 3

Quality of Experience

Quality of Experience (QoE) as defined in [19] represents "the overall acceptability of an application or service, as perceived subjectively by the end-user". Used in the context of telecommunications, it includes end-to-end system effects introduced by the client, terminal, network, service infrastructure and can be influenced by the user's expectations and context. The environment (at home or on the move), the nature of content (movies, news), user's expectations (expensive or cheap service) all affect the experience. The viewer's emotional involvement can affect the experience in a positive or a negative way. For example somebody who enjoys the content might be more tolerant to quality degradations, while an uninterested and bored viewer can get easily annoyed, even if the content is the same. However, the opposite could equally well be true: a sufficiently interested viewer might be more sensitive to disruptions and other quality problems because he/she is anxious not to miss any of the content. Even more, the device used for viewing the streamed media has an important effect over the perceived quality: on a Personal Computer (PC) one may expect to watch movies in HD while on a small device like a smart phone, lower resolutions will produce the same amount of satisfaction. So, measuring the experience of a user as a whole is clearly challenging since it depends on very subjective factors, but work has been performed to obtain an estimation of user satisfaction.

In video streaming, QoE can be seen as a function of two factors [20]:

- Quality of Content (QoC) is a user's subjective, often unconscious, appreciation of the attractiveness or relevance of a piece of content, like an important sport event or a fresh new episode of a TV series. It is also determined by how well the received media reproduces reality, given by the technical properties of the transferred video: display resolution, video frame-rate, frame compression.

- QoS, as discussed in Chapter 2, characterises the transport part of the streaming process. It is sometimes used directly as a measure of the whole experience because QoC is of little value unless delivered intact to the user. If users experience freezing in video playback, color blurring, significant delays for start-up or other transmission errors, they may abandon the service, even if the QoC is high.

So, QoE in streaming sessions is mostly determined by the quality of the video that arrives at the viewer, making the two components QoC and QoS strongly connected. A higher quality video has a larger size that has to be transported over the network, so a better QoS is needed, otherwise the video will arrive with errors at the viewer.

3.1 Video Quality Measurement

As defined in [21] video compression "refers to a process in which the amount of data used to represent image and video is reduced to meet a bit rate requirement (below or at most equal to the maximum available bit rate), while the quality of the reconstructed image or video satisfies a requirement for a certain application and the complexity of computation involved is affordable for the application". But besides that, videos are subject to distortions during acquisition, transmission, processing, and reproduction. Examples of these impairments include tiling, error blocks, smearing, jerkiness, edge blurriness, and object retention [22], so it is important to be able to identify and quantify video quality degradations.

3.1.1 Subjective Video Quality Measurement

Since human beings are the ultimate receivers in most image-processing applications, the most accurate way of assessing the quality of video is by subjective evaluation. This type of measurement is performed by observers who watch a series of videos and express their perceived quality by giving a mark on a scale from 1 to 5. Specifically, the subjects are asked to rate the pictures by giving some measure of picture quality or they are requested to provide some measure of impairment to the pictures. The average result gathered from all the subjects is called the Mean Opinion Score (MOS), where 1 is the lowest perceived video quality, and 5 is the highest perceived video quality measurement, as shown in Table 3.1.

The ITU has developed a standard in [23] where it describes how to present the videos to be evaluated and how to collect and interpret the

MOS	Quality scale	Impairement scale
5	Excellent	Imperceptible
4	Good	Perceptible but not annoying
3	Fair	Slightly annoying
2	Poor	Annoying
1	Bad	Very annoying

Table 3.1: MOS score and its meaning

results. Several methods are proposed, but two main classes based on whether a reference video sequence is present or not can be identified.

Double-stimulus Impairment Scale (DSIS) method

This method is recommended when one needs to measure the robustness of systems (i.e. failure characteristics, effects of transmission path impairments) a situation that is similar to the streaming case.

The observer is first presented with a reference sequence that does not contain impairments, then with the same sequence impaired. Following this, he/she has to note how annoying the image artefacts he/she experienced were, using the impairment scale (Table 3.1) and keeping in mind the reference.

At the beginning of each session, the observers should receive an explanation about the type of assessment along with a sample of the impairments they would see during the test. The worst quality observed should not necessarily be graded with the lowest score, but the samples should be chosen in such a way that they cover the whole grading scale.

Single-stimulus (SS) methods

This technique of assessing subjective video quality is used when there is no reference sequence to be compared to the tested one. Again, it is a typical case encountered in streaming situations, when the only available content is the one that arrives (with possible image degradation) over the transport channel. The test conditions should be similar to the DSIS ones and the voting scale remains the same. The testing sequence can be presented once or many times to the observer, in which case the method is called Single-stimulus with Multiple Repetitions (SSMR).

Analysis and presentation of results

No matter the evaluation method, the results will be gathered in the form of an average score from all subjects, along with an associated confidence

interval which is derived from the standard deviation and the number of observers. In [23] a 95% confidence interval is proposed, given by:

$$[\bar{u} - \delta, \bar{u} + \delta] \tag{3.1}$$

where: $\bar{u}$ is the average score over all subjects, or the MOS. δ is defined as:

$$\delta = 1.96\frac{S}{\sqrt{N}} \tag{3.2}$$

where N is the number of observers and

$$S = \sqrt{\sum_{i=1}^{N} \frac{(\bar{u} - u)^2}{N - 1}} \tag{3.3}$$

3.1.2 Objective Video Quality Measurement

Although subjective tests are the most precise methods to evaluate video quality, they are expensive and usually too slow to be useful in real-world applications. This is why mathematical models are used instead of human observers to approximate results of subjective quality assessment. These are based on specific criteria and metrics that can be measured and evaluated objectively by a computer program. There are several criteria on which objective measurements can be classified, but the most common is the one based on the quantity of reference information needed for the evaluation.

- Full Reference (FR-(VQM)) quality metrics - assume that a reference, distortion free video exists, the algorithm comparing it frame by frame with the sequence that needs to be evaluated. Due to the frame by frame comparison, the two videos must be spatially and temporally aligned to obtain relevant results. Temporal synchronisation in particular is quite a strong impediment and can be very difficult to achieve in practice, because of frame drops, repeats, or variable delay introduced by the system under test in the other clip. In spite of that, these methods are the most popular, starting with the common Peak Signal-to-Noise Ratio (PSNR) and Mean Squared Error (MSE) metrics and ending with more advanced ones based on Human Vision System (HVS).

- No-Reference (NR) quality metrics have access only to the impaired video so they determine its quality by analysing particular properties of the received signal. These methods are more practical for streaming situations since in such situations the original video is not available and there is no need of synchronization. The downside though is that usually they are less correlated with subjective tests than FR methods.

- Reduced-Reference (RR-VQM) quality metrics are a combination between NR and FR methods. They need only some information about the original video to estimate video quality. Their advantage is that they are more precise in estimating subjective MOS than NR methods and they require less information than the FR ones.

3.2 Video Quality Experiments

3.2.1 Subjective video quality evaluation for streaming sessions in wireless environment

To determine the effects of transmission errors over streaming sessions in a wireless access network, two subjective tests were conducted, one using the DSIS method and another using the SS procedure. Similar quality evaluation work has been performed in [24], [25], but usually tests like these are conducted in a controlled environment with high resolution screens where image imperfections can be clearly observed by the subject. Although the experiments described in this section followed the DSIS and SS guidelines, the equipment used was a regular laptop with a matte screen and a mobile phone. The idea behind these experiments was to see how the video impairments are perceived when the video stream is viewed on a mobile device, with a small screen.

The experimental setup is presented in Fig. 3.1. For both tests, the devices were connected to a wireless access point, further connected to the streaming server through an emulated UMTS Rel 99 Dedicated Transport Channel (DCH) [26]. The emulator can change the allocated bandwidth and the Bit Error Rate (BER) to simulate packet loss and packet corruption which affect the image quality. Neither mobility nor handover has been emulated, but typical varying impairments of the radio access network are duly reproduced. This means that packets in transit suffer from stochastic bit error or complete loss, with the help of Linux NetEm module. The fate of each packet is determined by a Gilbert-Eliot, two-state Markovian time-correlated error model. In the impairment stage, bits are flipped according to a Weibull model whose

parameters are tuned according to measurements from a real life UMTS network. During the simulations, the RLC Transparent Mode (suited for video streaming) was considered, with a Spreading Factor of 8, which limits the bandwidth to 240 kbps.

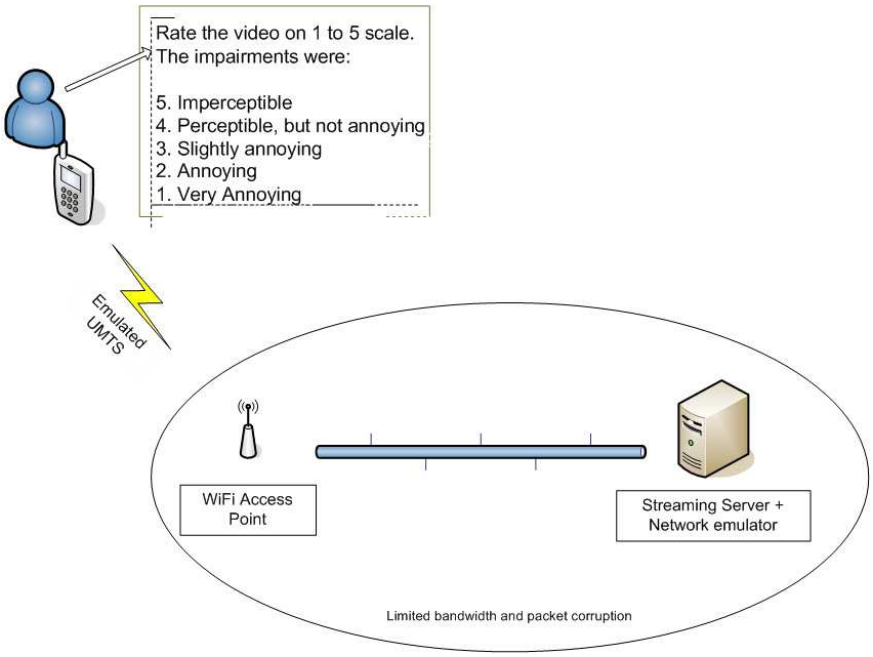

Figure 3.1: Subjective VQM

DSIS experiment

This test has been performed using the notebook display on which 2 versions of the same video were consecutively presented to the user. The video sequence was chosen from a popular TV series and encoded using the latest baseline profile of the H.264 codec. The baseline profile was needed for the compatibility with the Nokia N95 terminal used in the SS tests. The video characteristics are summarized in Table 3.2 and were chosen to match the channel bit-rate available in a typical UMTS connection.

The reference and impaired clips were streamed and played back in their original size while the subject was seated in front of the computer display.

At the transmission stage, two cases were considered: with and without transmission errors. For the first run, the video was streamed without transmission errors and it was considered to be the reference sequence.

Video Length	60s
Video resolution	391x256
Video frame rate	15fps
Video bitrate	165kbps
Codec	H264 baseline profile
Container	MP4

Table 3.2: Subjective quality assessment video characteristics

The second run was with the same video, but this time random packet loss and packet corruption was emulated. For each different observer, the seed for the random generation was initialised with the same value, so the test conditions were similar for each participant. After viewing the reference and impaired sequence, they had to rate the perceived quality, using the standard impairment scale presented in Table 3.1.

A total of 16 subjects were asked to participate in this experiment, with one more than the number of observers suggested by ITU in [23]. The majority were university staff or students, so it is possible that their common background influenced the scoring process. A wider spread of social categories should have been used for a better statistical relevance.

SS experiment

This test followed the DSIS one, when the subjects were already accustomed with the procedure and after they have already given their first vote. The video sequence was different from the one used in the previous experiment, so the evaluators were not influenced by the results of the first test. The encoding settings were the same as the ones presented in Table 3.2 so the results from the two tests can be compared. This time, the device used for watching the clip was a Nokia N95 smartphone, connected to the same network setup. The experiment consisted in one streaming session, with transmission errors enabled and with the device hand held by the observers to their own convenience.

Subjective quality evaluation results

The outcome from the two experiments has been summarized in Table 3.3.

The MOS obtained from the SS experiment is slightly better than the one obtained from the DSIS test. This can be caused by the small size of the smartphone display where image impairments may be less visible than the ones viewed on the computer screen. Also, observers

	DSIS experiment	SS experiment
MOS	3.1	3.5
Standard deviation	0.71	0.51
95% confidence interval	$MOS \pm 0.34$	$MOS \pm 0.24$

Table 3.3: MOS and standard deviation for the two experiments

might have had lower expectations when viewing content on the mobile phone and this could have increased their general score.

Fig. 3.2 shows the votes of each observer for the two experiments.

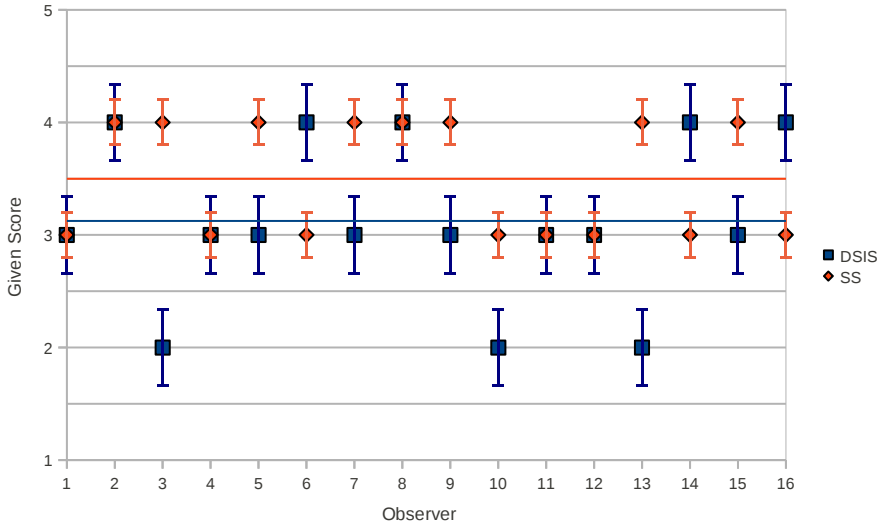

Figure 3.2: Score tests given by each participant, along with the 95% confidence interval

Conclusion

The two experiments have shown that streaming in a lossy, wireless environment has a considerable impact on the perceived video quality, but on a mobile device with a lower resolution display, the effect is less noticeable. The subjects considered that watching the impaired video produced a fair to good overall experience. Such evaluation procedure however is impractical to deploy in a real streaming scenario because of the resources it requires: available subjects to note the video quality, time to collect and process the data, or the lack of particular testing conditions required by the subjective evaluation methods.

3.2.2 QoE Evaluation of Video Streaming Through a FR Video Quality Metric

Since subjective tests are not practical for evaluating QoE for streaming sessions, an objective video quality evaluation experiment was conducted in order to determine user experience. However, at the time when the experiments were performed, the FR metrics available were optimised for the measurement of the performance of video codecs. The tests described in this section try to evaluate the behaviour of a FR quality metric when used in a streaming scenario. For this reason, we try to compare the QoE observed for three common streaming servers by using a publicly available FR video quality metric.

The experimental set-up combines:

- A dedicated streaming farm running CentOS, where several servers have been deployed, namely Apple's Darwin Server, Real Networks Helix and PVNS - a proprietary server. The streaming farm serves a limited set of H.264 encoded video sequences. The encoding has been performed so as to provide several versions of the same sequence, with various bit rates.

- A Linux-based emulator for a UMTS access link, similar to the one used for the subjective tests in Section 3.2.1

- A client computer hosting a variety of video players, namely Apple's QuickTime, RealPlayer and VLC. Media is delivered as RTP stream, but signalling with the server goes through RTSP. The received video sequences are recorded as played on the screen with the help of a screen-capture software, and stored for further analysis.

Tools for QoE evaluation

The main strength of the test-bed is the ability it offers to test the impact on the user experience of a variety of settings through the whole protocol stack. The drawback of this situation is the huge number of scenarios that would need to be tested to get a more or less complete view of the working of the system, hence the need to automate the evaluation process as much as possible. To evaluate the Quality of user Experience (QoE), the MSU Quality Measurement Tool [27] was used and the Video Quality Metric (VQM) was chosen over other metrics like Structural Similarity Index Method (SSIM) and Perceptual Evaluation of Video Quality (PEVQ) since it was supposed to offer the best correlation

with subjective quality tests at the time of experiment. It relies on Discrete Cosine Transform (DCT) comparison of the reference and processed frames and has originally been designed to compare the performance of encoders. The output of the algorithm is a score between 0 and 100, the lower the VQM score, the better the QoE. However, as it can be seen during the tests, the scale is not linear as the observed picture quality is acceptable with a score of 3.5, while a very distorted image (to the point where one would not tell what is in that picture) gets a score just above 10. In the experimental set-up, the screen capture of the received streamed sequence is compared to the original sequence. However, when the Gilbert-Eliot model switches to the impaired state, bit flips and losses can get so numerous that they trigger re-buffering at the player. Due to this re-buffering, the captured sequence loses synchronization with the original one. Since the metric is computed by comparing the videos frame by frame, a lack of synchronization biases the computation of the VQM. Even a small delay, hardly perceivable by a human being will lead the metric computation software to compare different frames for the two videos (source and received video). If the frames have different contents, the result will be a misleadingly high VQM. For example, the VQM score computed for two versions of the same sequence, the second one being delayed by a single frame, reaches 4.5 instead of 0. To avoid such a bias, the recorded sequence has been chopped in several parts whenever delay and consequent freezing or re-buffering were noticed. The frozen part has been trashed and the individual parts have been brought back to synchronization with the original video sequence, such that the VQM could be computed. Typically, for a recorded video sample of duration, say, 60s, 5s of unsynchronized material was removed. The VQM was then computed for the remaining 55s of the recorded video sequence. The QoE estimated with this VQM is optimistic, but still realistic, as long as the frequency and the duration of re-buffering events remain low. A similar issue should have affected the authors of [28], but this is not discussed in their paper.

VQM results

First tests were performed using one of the official H.264 video samples delivered with Darwin streaming server (known as Darwin sample). It contained one video stream encoded at 175kbps and one audio stream encoded at 48kbps, so a total bit-rate of 220kbps. As shown in Table 3.4, the results obtained were good, with a low VQM score for each streaming server. This can be explained by the very basic graphical content of the Darwin test sequence. Indeed it just shows an animated Quick-

Server	Darwin	Helix	PVNS
Average MOS score	3.52	2.98	4.3

Table 3.4: VQM results for Darwin sample sequence

Server	Darwin	Helix	PVNS
Average MOS score	12.1	11.4	11.9

Table 3.5: VQM results for the Leopold sample sequence

Time logo, with the background always remaining white. Therefore, to run deeper analysis, a locally created H.264 video sequence was chosen (known as Leopold sample), which contained more motion, encoded at 200kbps. Figures 3.3, 3.4 and 3.5 present three received frames from the second streaming sequence, with increasing VQM values, and hence degraded user experience. The samples were taken from the tests performed with Darwin streaming server and watched with VLC. Table 3.5 summarises the VQM results for the tests with Leopold video sequence. The computed score is around 11-12, which is fairly bad. In the worst case, scores of 3-4 are acceptable for a short period of time, as it can be seen from Fig 3.4. Clearly, the QoE is quite unsatisfactory.

Figure 3.3: Sample Frame with VQM = 0.6

Fig. 3.6 shows the evolution of the VQM score for the entire Leopold sequence. Its 991 frames were transmitted with 5,105 RTP packets.

The main source of errors in the video is the packet corruption at the NetEm module emulating the behaviour of a real UMTS network.

Figure 3.4: Sample Frame with VQM = 4.3

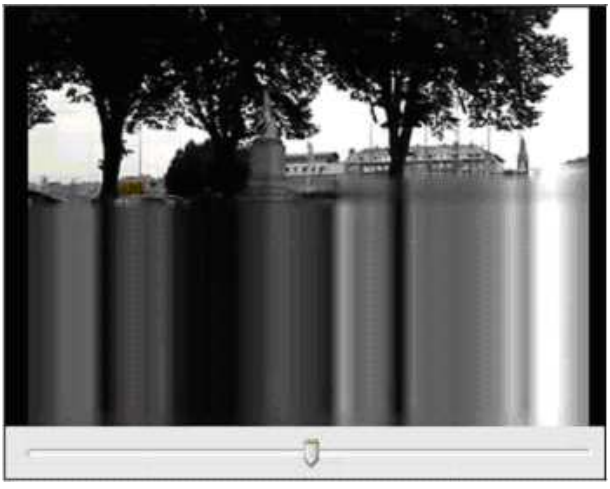

Figure 3.5: Sample Frame with VQM = 10.8

Although the network conditions are the same for each stream, each server behaves in its own way, which results in different VQM evolution patterns. In Fig. 3.6, the disruptions where the VQM fluctuates abruptly are related to state switches in the Gilbert-Eliot model. This behaviour emulates the burstiness of wireless bit errors [26]. However, the same behaviour is encountered when the corrupted packet is part of an I frame. This frame can not be decoded, nor the P and B frames that depend on that I frame. Since one frame is transported in several RTP packets, the chances that an I frame gets affected can be significant. To the best

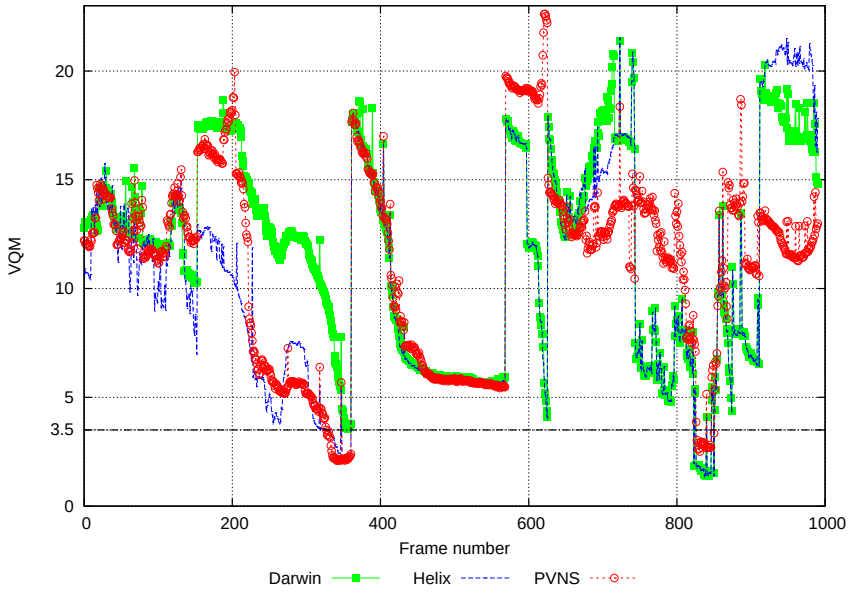

Figure 3.6: Evolution of the VQM score

knowledge of the authors, there is no threshold value on the VQM for the streaming scenario. Hence, we considered that a VQM smaller or equal to 3.5 is acceptable, which would correspond to a subjective MOS value of about 3 (Fair video quality). This threshold is illustrated on Fig. 3.6, where it appears that the VQM is most of the time above that threshold.

Fig. 3.7 is derived from Fig. 3.6 and it plots the cumulative histogram of the VQM data. The bins span from VQM = 0 (ideal case) to VQM = 24 (worst case observed), with a bin width of 3.5. The threshold and the mean VQM are also marked on Fig. 3.7, with vertical bars. From the cumulative histogram, Helix appears as a better performing server than the two other ones, as it exhibits a higher number of frames with good video quality (e.g. VQM below threshold) and a lower number of frames of poor quality (above threshold).

Traffic analysis

To understand why the quality fluctuated so much, a detailed traffic analysis was performed. Packet sniffer Wireshark was used to capture the RTP stream at the client. With the data contained in the RTP protocol it is possible to monitor the packet loss for the whole stream, or the delay and the jitter for each packet. In Fig. 3.8 the pattern of the

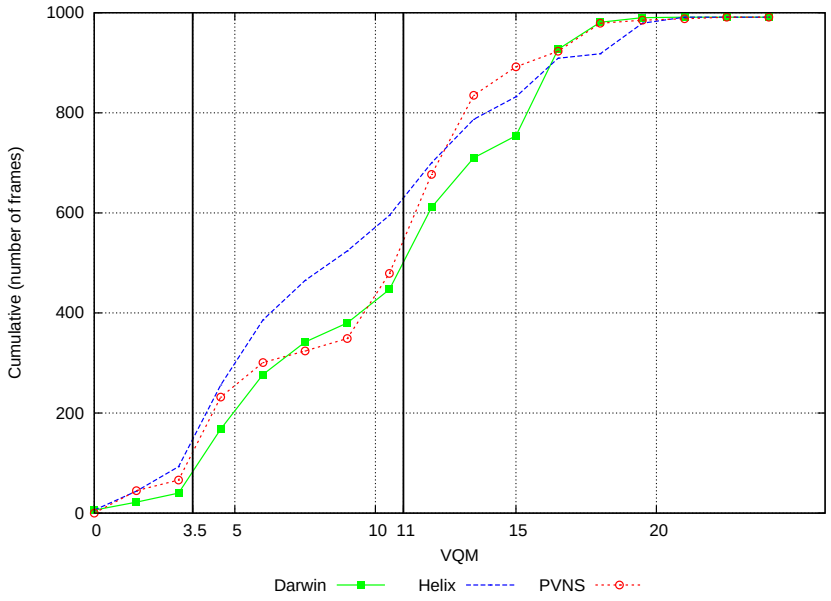

Figure 3.7: Cumulative histogram of the VQM scores

delay for each streaming server can be observed. Because the sequence is streamed in UMTS Rel'99 Transparent Mode, the network impairments have no impact on the delay, as there is no retransmission triggered. The delay exhibits a periodic pattern as shown on Fig. 3.8, which zooms on the beginning of a streaming session.

Fig. 3.8 shows that each server has a different delay pattern. PVNS has the lowest average delay (27.4ms), while Darwin and Helix have an average delay of about 32ms, as computed by Wireshark. However, the VQM score does not seem to be correlated to the delay pattern.

Conclusion of the VQM measurements

The Quality of Experience observed in this experiment was rather bad, which was confirmed by the VQM results. Although designed for codec comparison, the VQM metric can be used successfully to assess the video quality of streamed content. With a FR objective quality evaluation tool, the service provider could run several off-line tests to choose the optimal encoding parameters for a video to maximise the QoE over a specific type of access network. On the other hand, the synchronisation issues and the fact that the original video is needed for quality evaluation, does not make it a reasonable solution for real time measurements.

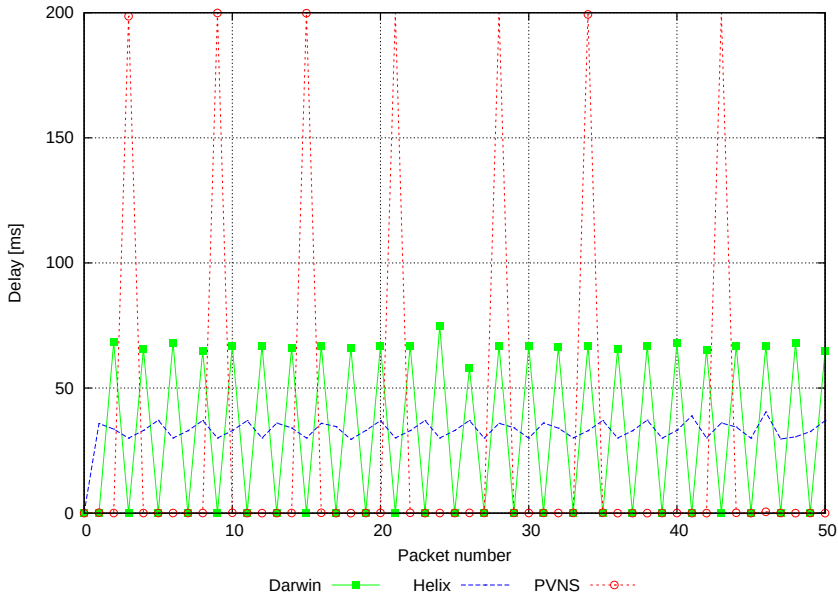

Figure 3.8: Delay evolution (zoom on the first 50 RTP packets)

3.2.3 Real time QoE evaluation using a NR-VQM algorithm

As described in the beginning of Section 3.1.2, the best method suited for evaluating QoE of a streaming session in real time is to use a NR quality metric. There are several NR-VQM algorithms proposed in the literature that take into consideration different aspects of video degradation to compute the MOS, but the most common impairments having a significant impact on the perceived video quality are those that affect temporal aspects of the video, like jitter and jerkiness [29]. In [30] and [31] the authors propose an algorithm that is based on the movement vectors of the video, in [32], [33] and [34] motion discontinuities and clearness-sharpness degradations are used to estimate the MOS, while in [35] the authors propose a model based on blurring, block distortion and jerkiness/jitter. When choosing the right method for evaluating QoE in a streaming session over a wireless channel, several aspects have to be considered [4]:

- the model must have a good correlation to the subjective MOS
- it has to be suited for use in wireless conditions characterised by frequent packet loss and low bandwidth

- it has to perform well even on small resolutions typical of mobile phone displays
- the computational complexity of the algorithm has to be as low as possible so it can to be implemented on mobile devices

In [4] several NR-VQM models, including the ones mentioned above, were analysed taking into consideration the characteristics of streaming sessions over wireless networks. The optimal solution seemed to be offered by the model based on motion discontinuities, proposed in [33], which was implemented as a plug-in in the GStreamer framework. Modified in this way, the media player analyses the image quality of the received stream every 400ms and gives a score that ranges from 10 to 100, 100 representing excellent quality. From the tests done with the enhanced media player in [4], some observations can be made about the behaviour of the NR-VQM implementation:

- there is a delay of 2-4s between the time when losses or high jitter appear in the network and the moment image degradation is detected. This delay depends on the buffer size of the GStreamer media player.
- there is a larger delay (between 10 and 20 seconds) between the moments the network perturbations stop and the video quality score goes back to the maximum value. The delay is proportional to the level of image degradation, as detected by the algorithm.

3.3 Concluding remarks

Part II presented the elements that define the quality of a streaming session and some of the methods used to determine them. The QoC can not be measured since it is purely subjective and depends on the preferences of the user. The content however has to be transported over a telecommunication network to the viewer and to successfully accomplish this, the network has to guarantee a certain level of QoS. Throughput, packet loss rate and latency are the main parameters that determine the QoS and Section 2.3 has shown that by measuring the variations of RTT and packet loss, one can detect if the QoS level assures the correct delivery of the media stream. When there are losses or high jitter present in the network, the video quality can be degraded which affects the overall experience of the user. Picture quality is therefore an important element of the streaming experience and it is necessary to be able to measure it. The subjective quality evaluation experiment showed

that watching impaired content on a small device may hide certain image artefacts, making the viewing experience better from this point of view when compared to a large display. The second experiment described how a Full-Reference metric, originally created to compare video codecs, could be used to measure the impact of network degradation over the video quality. But, as pointed out in the two experiments, both methods are not suitable to determine QoE in real time, so a No- Reference video quality metric should be used for this purpose. The work performed in [4] and [29] shows that motion based video quality metric is the most suitable for a streaming scenario and the algorithm presented in [33] and [34] has been implemented as a plug-in in the GStreamer framework.

Part III

Reporting Mechanisms

Chapter 4

Reporting QoS parameters

Typically, a streaming session is a one way process, with the media flow being sent from the server to the receiver. As discussed in the previous chapter though, there are a lot of elements that can be evaluated to determine streaming quality, but most of them have to be computed at the client side, so it is necessary to have a way to send those parameters back to the server. This is why in practice there is some type of message exchange between the two entities mostly to provide feedback on the quality of the data distribution.

4.1 RTCP protocol suite

The RTCP protocol suite is part of the RTP/RTCP standard [1] and is the default method for sending feedback information from the player to the server in a streaming scenario. It was designed to help the streaming server perform the following actions:

- to monitor and diagnose network problems
- to identify different media flows and to perform synchronization between flows if necessary (for example between video and audio), by using a unique identifier for each media stream: Canonical Name (CNAME)
- to compute the sending frequency of the RTCP messages

The standard defines several packet types that carry different information from the sender to the receiver and vice-versa:

- Sender Report (SR) contains information that is sent by the server to the client(s). Its structure is depicted in Fig. 4.1

- Receiver Report (RR) contains information similar to SR, but is sent by the receiver to the sender
- Source Description (SDES) contains data about the source, including the CNAME
- BYE packet is sent to close a session, either by the client or the server
- Application specific packet (APP) contains information particular to certain applications. Only the format of the packet is standardised, the data contained being application specific.

From the packet types listed above, the SR and RR are important because they contain information about QoS parameters of the transmission: delay, jitter, packet loss. APP packets offer additional information but usually these are application specific for proprietary solutions, where this study focuses on open schemes. Typically a RTCP packet contains one or more report blocks, as shown in the structure presented in Fig. 4.1. The RR packet looks similar, without "the sender info".

From all the fields present in the SR and RR, the following ones contain information about the state of the network:

- `NTP timestamp` - This is a 64 bit value that represents the wallclock time when the current SR was sent. By storing this value, together with the arrival timestamp of the next RR and with the data in the DLSR field, the server can estimate the round trip propagation time (RTT) to the client. This can be further used to compute the $Smooth_{RTT}$ and RTT *Deviation* as presented in Chapter 2, Section 2.3.1.
- `fraction lost` - represents the ratio between the number of lost packets and expected number of packets since the emission of the last RR or SR.
- `cumulative number of packets lost` - represents the number of lost packets since the beginning of the session. Taking into consideration only fraction losses or only cumulative losses is not sufficient enough to determine if losses are significant or not. For example, if we have 4 RTP packets sent between 2 RTCP reports and 2 of those packets get lost, then the fraction field would report 50% loss, which is very high thinking in percentages, but is low as number of packets. On the other hand, cumulative loss reports the total number of packets for the whole session in each RTCP report. Hence,

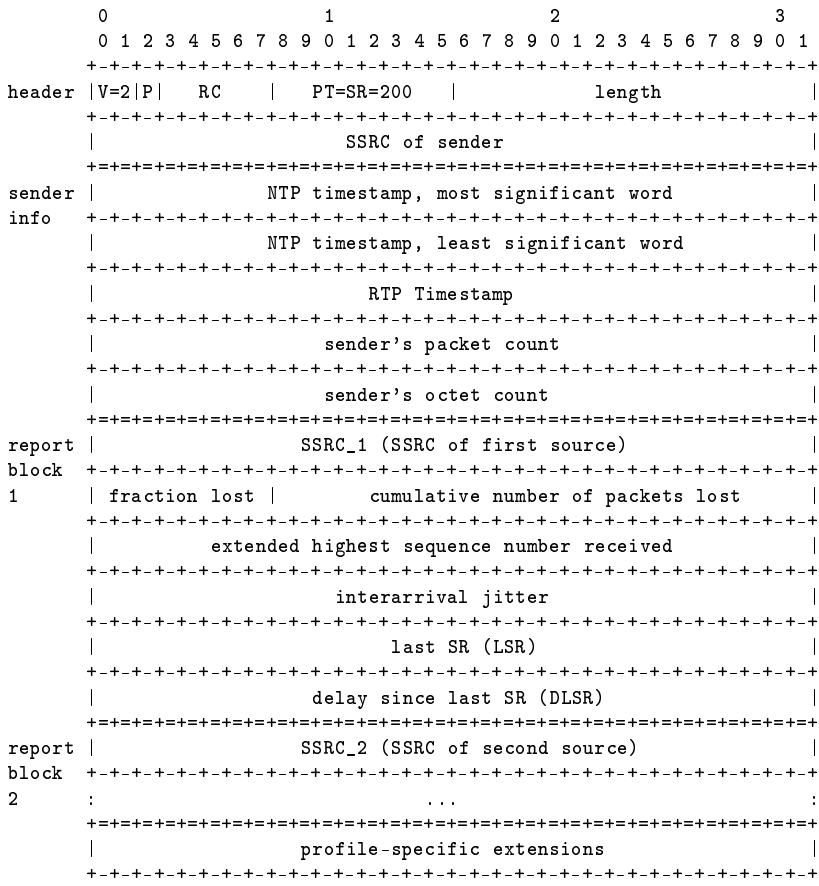

Figure 4.1: RTCP Sender Report packet structure [1]

with the cumulative loss from two consecutive RTCP reports the number of lost packets between those consecutive reports can be computed, which can be used together with the fraction lost.

- `inter-arrival jitter` - "*is an estimate of the statistical variance of the RTP data packet inter-arrival time, measured in timestamp units*" [1]. Because it is measured in timestamp units, the measurement can introduce noise in case of some codecs so instead of this metric, the variation proposed in **Part II, Section 1.3.1** is used.

- `last SR timestamp (LSR)` - represents the timestamp (only the middle 32 bits) of the most recent RTCP sender report (SR) packet received.

- `Delay since last SR timestamp (DLSR)` - represents the "processing" delay: between the arrival of the last SR packet and the sending of the current report block.

Based on the information presented above, a few observations should be made: the RTT and jitter are instant values, reflecting the state of the network only at that precise moment when the report was sent, while loss information represents a cumulative value, for the whole session or for the interval between 2 consecutive RTCP reports. Therefore, the accuracy in estimating the network state depends on the feedback frequency. With more frequent messages the sampling interval will be smaller so the precision will increase. The standard proposes a method based on the session bandwidth to compute the sending interval and it recommends to allocate 5% from the total session bandwidth to the RTCP feedback. However, most commercial media players that implement the RTP/RTCP protocol use only the recommended value for a fixed sending interval, which is 5 seconds. This reporting period can be quite long, especially in high mobility scenarios. For example, a mobile device moving at a highway speed of 120km/h covers a distance of 200m within 5 seconds.

4.2 3GPP Rel.6 RTCP extensions

As shown in Section 2.2, buffer capacity and buffer filling are important QoS parameters in a streaming session to ensure continuous playback. This is why the 3GPP consortium has proposed in [2] an extension of the current RTCP protocol, using the RTCP APP framework, to allow the player to send feedback about its buffer status.

The specific application packet proposed contains information about the Next Application Data Unit (NADU) to be decoded and the report block has the structure presented in Fig. 4.2.

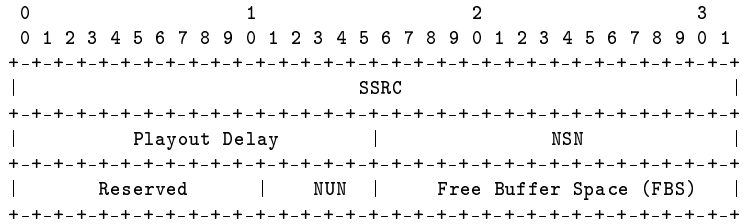

Figure 4.2: Data block for RTCP NADU APP packet [2]

The metrics contained in the NADU report block are defined below, as they appear in [2]:

- `Playout delay` - is the time interval between the scheduled playout time of the next Application Data Unit (ADU) to be decoded and the time of sending the NADU APP packet, as measured by the media playout clock, expressed in milliseconds. The playout delay allows the server to have a more precise estimate of the amount of time before the client will underflow.

- `Next Sequence Number (NSN)` - represents the RTP sequence number of the next ADU to be decoded

- `Next Unit Number (NUN)` - designates the next frame to be decoded. In the case of H.264 codec, it represents the next Network Abstraction Layer (NAL) unit to be decoded.

- `Free buffer Space` - reports the amount of free buffer space available in the client at the time of reporting.

From the metrics contained in a NADU packet, the streaming server can extract two useful information, among others: the number of packets stored in the client buffer and decoded by the client and the amount of free buffer space. Fig. 4.3 clearly illustrates these parameters:

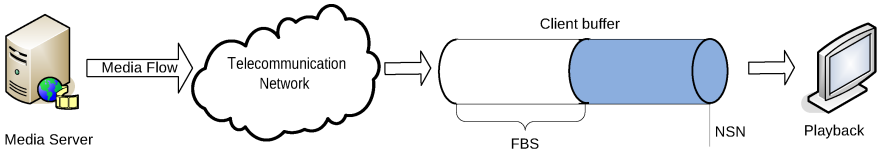

Figure 4.3: Buffer parameters

4.3 RTCP-XR extensions

For some specific applications like VoIP or multicast sessions, the metrics present in the standard RTCP reports were not enough, so a new RTCP packet type able to transport more detailed information was standardized in [36]. XR packets have a similar structure to the standard RTCP packet, being composed of a header and one or multiple report blocks. The RFC3611 standard defines seven additional report blocks which fall in three categories, but new report blocks can be defined in the future using the same framework. Below are the three classes of extended report blocks with some of the metrics they transport:

1. Packet-by-Packet group contains detailed statistics upon packet receipt and loss events. In this category are three report blocks which are aimed to help in the discovery of the network topology tree in multicast sessions, but can also be used in unicast sessions:

   1.1. `Loss Run Length Encoding (RLE) Report Block` - contains information about which packets from the RTP flow were lost.

   1.2. `Duplicate RLE Report Block` - contains statistics about the reception of duplicate RTP packets.

   1.3. `Packet Receipt Times Report Block` - includes the arrival times for each RTP packet.

2. Reference time related blocks - contain statistics about the wall-clock times, similar to the ones present in the RTCP SR report, extending in this way the capability of the clients (receivers) to compute the RTT.

   2.1. `Receiver Reference Time Report Block` - contains the timestamp when this report block was sent

   2.2. `DLRR Report Block` - carries the delay since the last Receiver Reference Time Report Block was received, similar to the DLSR metric in RTCP RR, allowing the client to compute the RTT.

3. Summary metric block types: consists of two report blocks that give a broader range of information, without being extremely detailed.

   3.1. `Statistics Summary Report Block` - Offers information about some QoS parameters that are also present in the standard RTCP reports, but this time these are not instant values, but an average over the RTCP interval. These metrics are:

      3.1.1. Minimum, maximum, average and standard deviation values for the jitter

      3.1.2. Minimum, maximum and average values for the Time To Live (TTL) or hop limit for the RTP packets

      3.1.3. Number of lost and duplicated packets in the interval between two RTCP reports

   3.2. `VoIP Metrics Report Block` - contains some general QoS parameters that are not present in other RTCP or RTCP-XR reports and metrics specifically designed for VoIP communications. Albeit aimed at VoIP calls, some of the fields could be used in a streaming scenario as well:

      3.2.1. `Burst density` - The fraction of RTP data packets within burst periods since the beginning of reception that were either lost or discarded. (A burst is a period during which a high proportion of packets are either lost or discarded due to late arrival). This information is useful since usually the signal degradation in a wireless network affects several adjacent packets, therefore the burst name. If the burst density is high and the burst duration is long enough, we might draw the conclusion that the degradation of the network is for a longer term.

      3.2.2. `Gap density` - The fraction of RTP data packets within gaps since the beginning of reception that were either lost or discarded. (A gap is the period of time between two bursts and a low number of packets can be lost in a gap).

      3.2.3. `Burst duration` - The mean duration, expressed in milliseconds, of the burst periods that have occurred since the beginning of reception

      3.2.4. `Gap duration` - The mean duration, expressed in milliseconds, of the gap periods that have occurred since the beginning of reception.

      3.2.5. `MOS-LQ` - an estimation of the MOS for Listening Quality

3.2.6. `MOS-CQ` - an estimation of the MOS for Conversational Quality

4.4 RTSP protocol

The Real Time Streaming Protocol (RTSP) [37] is an application-level protocol designed to establish, negotiate and control audio-video streaming sessions. Typically it is used at the beginning of the session to initiate the communication between the server and the client and to act as an interface between the two, allowing Video Cassette Recording (VCR) like commands as PLAY, PAUSE, STOP, FFWD, etc. It is a stateful, text-based protocol as illustrated in Fig. 4.4.

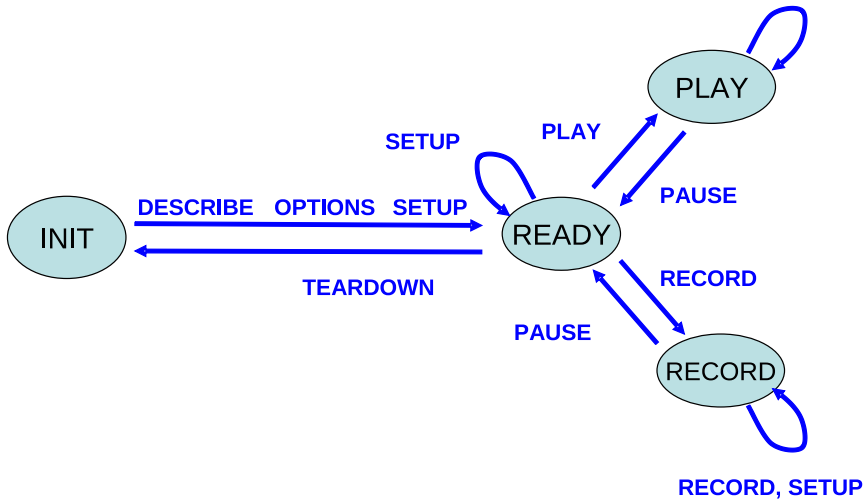

Figure 4.4: RTSP states and messages

The RTSP protocol is based on a request/response approach, similar to HTTP, where the client or the server sends a message and the other replies with an appropriate response. The response and the message itself contain several standard header fields that are mandatory for a standard RTSP implementation, with the possibility to define new headers. The main messages are:

- DESCRIBE - asks for the description of the media object identified by the request URL from a server (this corresponds to the initialisation of the streaming session)
- SETUP - used to negotiate the transport mechanism for the media flow

- PLAY - tells the server to start sending the data
- PAUSE - tells the server to temporarily stop sending data
- TEARDOWN - ends the playback and frees resources allocated to the current session
- GET_PARAMETER - the server can ask the client to send specific information as a response to this message
- SET_PARAMETER - the client can ask the server to change the value of some parameter

The purpose of the RTSP protocol is not to send regular feedback like in the RTCP case, but it is rather event-driven since messages can be generated by random events and can be sent at any moment in time. Unfortunately, the standard defines only one header field related to QoS parameters that is an optional implementation. This is the `bandwidth` field that contains information about the estimated bandwidth available at the client at the beginning of the session. The method used for bandwidth estimation is chosen by the application developer.

4.4.1 3GPP Rel.6 RTSP header extensions

The 3GPP organisation proposes in [2] some extension headers for the RTSP protocol to be used when streaming in mobile networks.

3GPP-Link-Char header

This extension is aimed to complement the `bandwidth` field by enabling clients to report the link characteristics of the radio interface to the media server, especially when there are QoS reservation mechanisms involved. This enables the server to set some basic assumptions about the possible bit-rates and link response. Three parameters are defined that can be included in the header, and future extensions are possible to define. The three parameters are:

1. `GBW`: the link's guaranteed bit-rate in kilobits per second
2. `MBW: the` link's maximum bit-rate in kilobits per second
3. `MTD: the` link's maximum transfer delay, in milliseconds.

The 3GPP standard recommends that the `3GPP-Link-Char` header should be included in a SETUP or PLAY request by the client, to give

the initial values for the link characteristics. A SET_PARAMETER or OPTIONS request can be used to update the metric values in a session currently playing, but SET_PARAMETER produces less overhead both in bandwidth and server processing.

3GPP-Adaptation header

To avoid buffer overflows, which will determine the client to discard useful data, this extension header, beyond other metrics, reports information about the client's buffer. Together with the 3GPP RTCP NADU APP packet described in Section 4.2, this information can be used to monitor buffer level. This allows the server to closely analyse the buffering situation on the client side and to do what it is capable in order to avoid client buffer overflow. The client specifies how much buffer space the server can utilize and the minimum target level of protection the client perceives necessary to provide interrupt-free playback. The fields that contain buffer information are:

- `buffer-size-def` - represents the total buffer size of the player, including reception, dejittering, and, if used, pre-decoder buffers and deinterleaving buffers for complete ADUs.
- `target-time-def` - represents the target protection time or preroll buffer, representing the minimum amount of buffering (in ms) that the client perceives necessary for interrupt-free playback.

This header can be used in the following RTSP messages: SETUP, PLAY, OPTIONS and SET_PARAMETER. It can be signalled before the playback begins since buffer characteristics usually remain constant for the whole duration of the session.

Chapter 5

Reporting QoE parameters

As seen in Chapter 4, the standards offer several options to send the QoS parameters from the client back to the server. For the QoE feedback unfortunately, at this moment there is no standard defined, but several proposals or extensions to current protocols that will be analysed in this chapter.

5.1 RTCP APP defined packet

The RTP/RTCP RFC specifies the RTCP APP packet structure that can be used to send application specific data. If the application becomes widely available, the APP packet could be registered at Internet Assigned Numbers Authority (IANA) to become a stand-alone RTCP packet type. The structure of the APP packet is presented in Fig. 5.1 where the field
`application-dependent` data can be used to send QoE specific parameters, like the MOS value. The `name` field is assigned to differentiate between other APP packets that might be used in the same application.

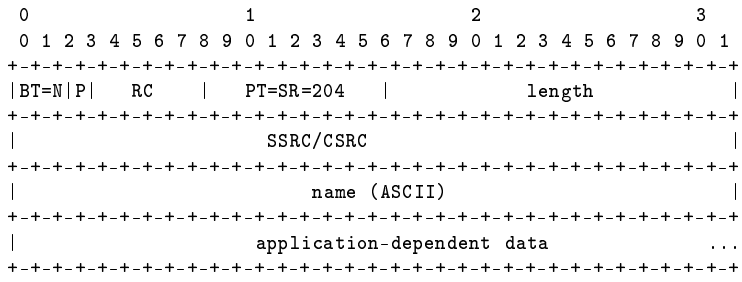

Figure 5.1: RTCP APP packet framework [1]

5.2 RTCP-XR extensions for QoE

The RTCP-XR standard does not propose specific blocks for video streaming and the only reference to parameters related to QoE are the MOS-LQ and MOS-CQ fields in the VoIP Metrics Report Block. Being designed especially for voice conferencing, the other metrics in that report block are not useful in a video streaming scenario. In [3] the authors proposed a new report block specifically designed to transport the QoE parameters in multimedia applications. The structure of this report bloc is presented in Fig. 5.2:

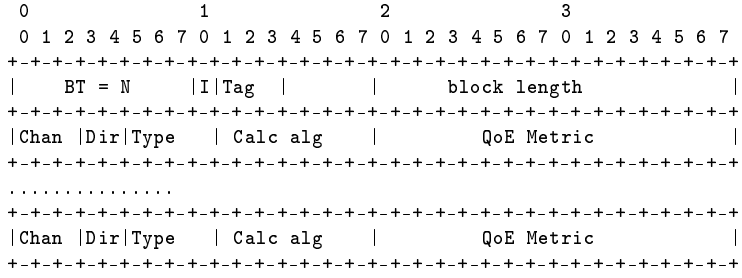

Figure 5.2: RTCP XR packet framework with QoE report block [3]

The main fields of this report block are:

- `BT` - The report block type to be registered with IANA if the draft becomes a standard.
- `I` - Interval Metric flag - When set to zero, this field indicates that the reported QoE parameters represent measurements for the current RTCP interval; when set to one, the measurement is cumulative for multiple RTCP intervals.

- `Tag` - used with the Measurement Identifier block proposed in [38] to define the duration of the measurement interval.
- `Type` - the type of MOS present in the QoE Metric field. For example, it can be MOS-V for video quality, MOS-A for audio quality, PSNR, etc
- `Calc alg` - This is the calculation algorithm used to determine the MOS, with 3 algorithms defined and with an option to supply new algorithms through Session Description Protocol (SDP).
- `QoE Metric` - the MOS, stored on 16 bits as a 8:8 integer scaled representation. It can take values in the range 0.0 to 255.996

This RTCP-XR extension is, so far, the most comprehensive method to send QoE related information in a streaming session, but unfortunately this did not become a standard since the draft expired in April 2008. Nevertheless, it can still be used as it is, so the author in [4] has selected it to encapsulate the MOS computed at the client site to send this value back to the server.

5.3 3GPP Rel.6 RTSP QoE headers

As was the case with QoS metrics, there are several extensions to the RTSP protocol proposed by 3GPP in "Transparent end-to-end Packet-switched Streaming Service (PSS)" technical document [2] to help with the feedback of QoE parameters.

3GPP-QoE-Metrics header

Although called QoE headers, these fields do not contain information about a global quality indicator like the MOS, but they include metrics directly related to playback impairments like re-bufferings or changes in frame-rate. In this standard, 3GPP proposes a reporting mechanism based on RTSP messages, with a constant feedback interval similar to the RTCP protocol. The `3GPP-QoE-Metrics` header is defined to enable the media server and client to negotiate which QoE metrics the media client should send, how often they should be sent and how to turn the metrics transmission on and off. This header can be included in the following RTSP messages: SETUP, SET_PARAMETER, OPTIONS and PLAY with the most important fields listed below:

- `Metrics` - includes the list of metrics that will be reported using the `3GPP-QoE-Feedback` header

- `Sending-Rate` - represents the maximum time period in seconds between two successive reports. When the `Sending-Rate` is set to 0, the reporting is done only when a specific event occurs at the client, otherwise the interval is set to the precise value present in this field. The minimum duration is 1 second, while the maximum is not specified. The value "End" means that only one report is sent at the end of the session, in a TEARDOWN message.

- `Measure-Range` - specifies the time interval in the current stream for which the metrics will be reported.

3GPP-QoE-Feedback header

This header is used to send the QoE parameters that were set with the `3GPP-QoE-Metrics` extension. It is recommended to include it in a SET_PARAMETER, PAUSE or TEARDOWN message, depending on the time when feedback is sent. The metrics that can be delivered with this RTSP extension are defined in the same technical specification [2], as follows:

- `Corruption duration metric` - represents the time period from the last good frame before the corruption, to the time of the first subsequent good frame or the end of the reporting period. A corrupted frame is defined as an entirely lost frame, or a media frame that has quality degradation. However, the quality degradation detected at the codec layer is not interpreted so the effect on the visual perception is not determined as is the case of objective video quality measurements.

- `Re-buffering duration metric` - depicts the duration of any stall in playback due to a buffer underflow event.

- `Initial buffering duration metric` - is the time from receiving the first RTP packet until playing starts.

- `Successive loss of RTP packets` - designates the number of consecutive RTP packets that were lost during transmission.

- `Frame rate deviation` - represents the difference between a reference frame rate, and the current playback frame rate, expressed in fps. The pre-defined frame-rate is signalled by the server in the same 3GPP-QoE-Feedback header through the `FR` parameter.

- `Jitter duration` - Playback jitter is expressed in seconds and appears when the absolute difference between the actual playback

time and the expected playback time is larger than a predefined value, which is 100 milliseconds.

Some of the events listed above can happen more than once during the reporting interval, in which case every incident will be present in the QoE report.

Chapter 6

Feedback methods implementation

The previous chapters presented several ways of sending QoS and QoE parameters, which included standardized and non-standard methods. Therefore it is important to know which feedback mechanisms have been adopted by the most popular media servers and players to determine the compatibility between them. Such an extensive task needs to take into consideration not only the players and servers themselves, but also other aspects of the streaming chain, like the access network, device, Operating System (OS), so there is a large variety of settings and scenarios that could be investigated. Fig. 6.1 shows all the test combinations performed to verify the implementation status of the feedback messages.

The central component of the testing set-up was the streaming server farm, consisting in four of the most common products, namely HELIX, DARWIN, PVNS and LIVE555. Multiple platforms and OSs were used and as can be observed in Fig. 6.1, some client applications were multi platform with "lighter" versions for mobile devices. The videos were encoded with the latest version of H.264 codec and two types of containers were used: mp4 and a multi encoding 3gp file. A packet capture software was used on the server side to analyse all the RTCP and RTSP message exchange, the results being summarized in Fig. 6.2

It can easily be observed that the RTCP-XR standard is not supported at all and in fact there is no support whatsoever for a QoE metric. 3GPP NADU APP packet is recognized by the mobile version of RealPlayer and PVNS, which also implement some of the RTSP extensions proposed by 3GPP. It must be mentioned though, that the 3GPP extensions were signalized only when the 3gp files were streamed, but that should not be a problem since all the players were able to recognize

this type of file. When Darwin streaming server is used in combination with Apple's media player QuickTime, there is a new RTCP packet that is sent, the QTSS application packet. This is in fact part of the so called reliable RTP, a proprietary protocol developed by Apple to improve the delivery of streaming media. A similar behaviour is present when RealPlayer is used in combination with Helix server, the RNWK RTCP APP packet being sent by the client. However, information about this packet type is not publicly available.

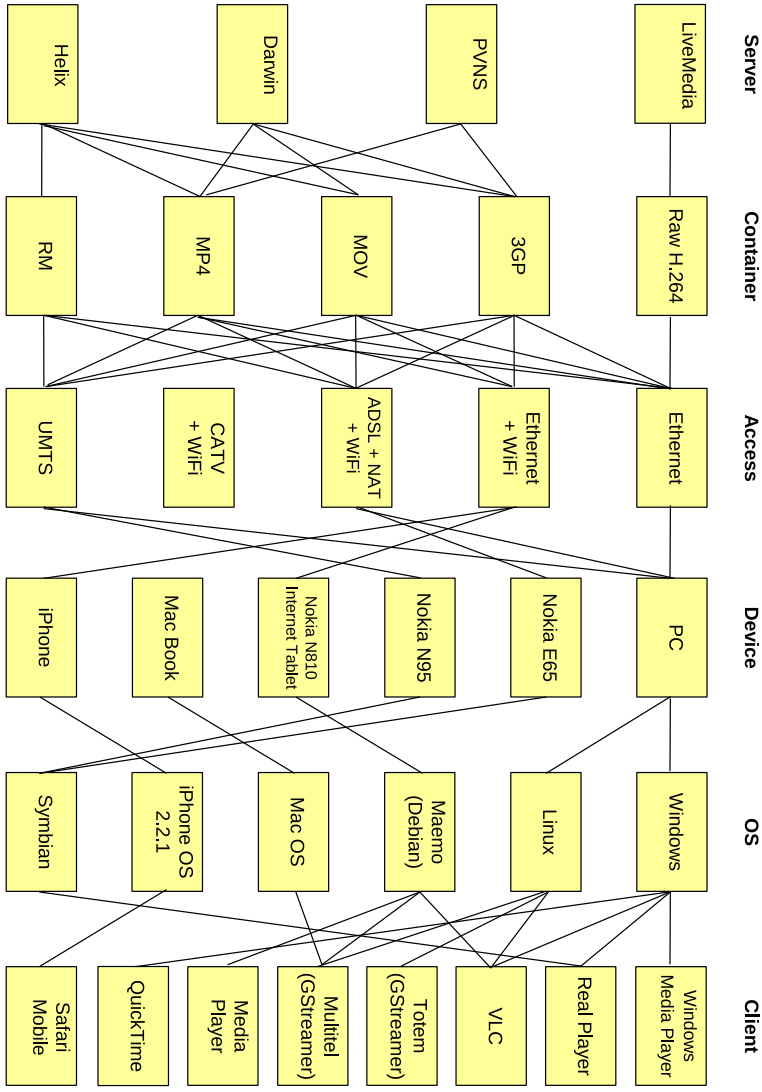

Figure 6.1: Testing Combinations

	OS	Application	Basic RTCP	RTCP XR (RFC 3611) +extensions	RTCP NADU APP (3GPP)	Basic RTSP	3GPP headers in RTSP
Server	CentOS	Darwin Helix	SR+SD, BYE, APP(qtsi) SR+SD, BYE, APP(RNWK)	NO NO	NO Yes Mobile Ext.	Yes Yes	NO Yes Mobile Ext.
		LiveMedia PVNS	SR+SD SR+SD, BYE	NO NO	NO Yes	Yes Yes	NO Yes
Client	Windows XP	QuickTime	RR+SD, BYE, APP(qtss,ack)	NO	NO	Yes	NO
		Real Player	RR+SD, BYE, APP(qtss)	NO	NO	Yes	NO
		VLC	SR+SD, RR+SD, BYE	NO	NO	Yes	NO
		Windows Media Player	Yes	NO	NO	Yes	NO
	Windows Vista	QuickTime 7.5	Yes	NO	NO	Yes	NO
		Real Player 11.1	Yes	NO	NO	Yes	NO
		VLC 0.8.6e	Yes	NO	NO	Yes	NO
		Windows Media Player	Yes	NO	NO	Yes	NO
	Windows 7	QuickTime	RR+SD, BYE	NO	NO	Yes	NO
		Real Player	Yes	NO	NO	Yes	NO
		VLC	SR+SD, RR+SD, BYE	NO	NO	Yes	NO
		Windows Media Player	Yes	NO	NO	Yes	NO
	Linux Ubuntu	Mulitel	Yes	NO	NO	Yes	NO
		Totem 2.24.3	Yes	NO	NO	Yes	NO
		GStreamer NR-VQM plugin	Yes	Yes (QoE draft)	NO	Yes	NO
		VLC 0.9.4	SR+SD, RR+SD, BYE	NO	NO	Yes	NO
	Linux Maemo	MediaPlayer 1.2-23	RR+SD	NO	NO	Yes	NO
		VLC 0.9.0	Yes	NO	NO	Yes	NO
	Mac OS	VLC 0.8.6e	Yes	NO	NO	Yes	NO
	iPhone OS	Safari Mobile	NO	NO	NO	NO	NO
	Symbian OS	Real Player s60.30.45.01.M	RR+SD	NO	Yes	Yes	Yes

Figure 6.2: Status of the current RTCP/RTSP standards implemented in common clients and servers

6.1 Summary

In this part several possibilities to send QoS and QoE parameters from the player back to the server were presented. The standards propose many methods and metrics for QoS feedback but few QoE possibilities. Even more, the implementation status in the industry resumes to the basic RTP/RTCP standard, while the 3GPP extensions still lack wide support. Table 6.3 summarizes the relation between the different reporting fields and the QoS or QoE elements discussed in **Part II**. It should be noted that in most of the cases the QoS/QoE metric it is not reported itself, but elements that help estimate it, or other particular information that give a more precise estimation of that QoE/QoS element.

Feedback mechanism			Reported QoS/QoE parameters					
Protocol	**RTCP Report Block/ RTSP Header**	**Field**	**Throughput**	**Transm. Delay**	**Delay Variation**	**Packet Loss**	**Buffer Charact.**	**QoE**
RTCP	SR	NTP Timestamp		√	√			
	RR/SR	Fraction lost	√			√		
		Cumulative lost	√			√		
		Inter arrival Jitter			√			
		LSR		√	√			
		DLSR		√	√			
	NADU	Playout delay					√	
		NSN					√	
		NUN					√	
		Free buffer space					√	
RTCP XR	RLE	bit_chunk				√		
	Duplicate RLE	bit_chunk				√		
	Packet Receipt Times	Receipt Time of RTP packet		√				
	Receiver Ref. Time	NTP Timestamp		√				
	DLRR	DLRR		√	√			
	Statistics Summary	Min,Max,Avg jitter			√			
		Lost packets				√		
		Duplicated Packets				√		
	VoIP metrics	Burst density				√		
		Gap Density				√		
		MOS LQ						√
		MOS CQ						√
	QoE metrics	QoE metric						√
RTSP	3GPP link char	GBW	√					
		MBW	√					
		MTD		√				
	3GPP Adaptation	Buffer size def					√	
		Target time def					√	
	3GPP QoE Feedback	Corruption Duration						√
		Re buffering duration					√	√
		Initial buffering duration					√	
		Successive loss of RTP packets	√			√		
		Frame rate deviation						√
		Playback jitter duration						√

Figure 6.3: Summary of the feedback mechanisms analysed in **Part III** and the reported QoS/QoE metrics discussed in **Part II**

Part IV

Adaptive Streaming

Chapter 7

State of the Art

As seen in the previous chapters of this thesis, the QoE of a streaming session is mostly determined by the quality of the video displayed on the user's screen, which is highly influenced by the state of the transport channel. The main concern regarding the communication network remains frequent bandwidth variation which can be limited to some extent through QoS mechanisms by prioritizing delay sensitive traffic. Even in those cases though, it can be possible that the encoding rate of the media flow surpasses the maximum throughput, which can overflow network buffers and thereby lead to packet loss. This chain of events will ultimately generate image degradation or re-bufferings.

In the beginning of Internet streaming, the user had the option to select the appropriate video encoding rate from a list and the server would stream the video with the selected parameters for the entire session. For example, a content provider had to create separate versions for users of 28Kbps and 56Kbps modem connections, ISDN lines, etc., but this solution had many obvious problems. First, it was based on the assumption that the actual bandwidth of the channel between server and client is bounded only by the last link in the chain (i.e., client's connection to the ISP), which is not always true. And, the most important, it did not address the possibility of dynamic changes in channel bandwidth and loss statistics. This approach is acceptable when the user has a wired connection where the network characteristics do not change much over time, but in a wireless environment, where the viewer is usually moving, the transmission rate has to match the bandwidth variation pattern. To achieve this without emptying or overflowing the player's buffer, the server has to automatically adapt the encoding bit-rate of the video, based on the instantaneous experience of the viewer.

A typical adaptive streaming system is presented in Fig. 7.1. In order

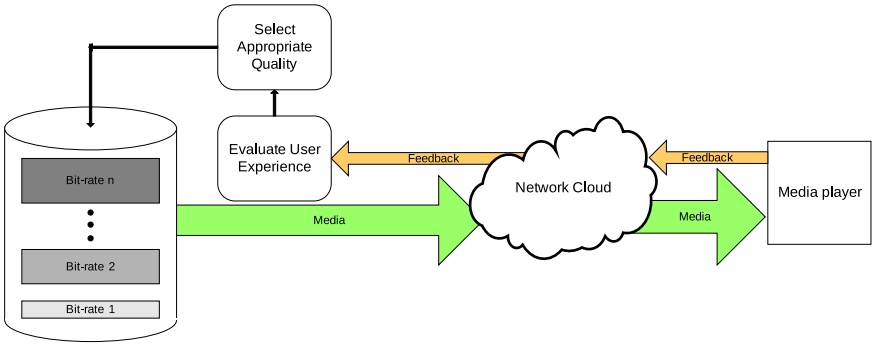

Figure 7.1: Overview of a typical adaptive streaming chain

to achieve rate adaptation, there are three main actions that have to be performed:

1. Estimate the user experience
2. Inform the server that a change in quality is needed, or forward the QoE status and let the server decide whether to change the video rate or not
3. Increase or decrease the bit-rate of the streamed video

The first item has been discussed in detail in **Part I**. Step 2 of the adaptation process depends on which entity decides that a change in bit-rate is required. Two cases can be distinguished:

- Server Centric - In this case the server takes the decision on when to perform rate adaptation according to the feedback received from the client. The advantage of this strategy is that the service provider has the control over the content that is sent (e.g. can apply different charging policies for higher quality) and it helps with scalability if the server gets too many requests. The downside is that the speed with which the server reacts to network throughput variations depends on the frequency of the feedback. Feedback possibilities have been discussed in detail in **Part III, Reporting Mechanisms**.
- Client Centric - In client side adaptation, the receiver of the stream is the one that selects the content quality it wants to receive. The advantage is that the decision of quality switch can be made instantaneously in the moment when the client detects some problems in the streaming quality. The disadvantage is that this type of behaviour is not yet standardized and the server needs to understand

the request sent by the client. Usually, this adaptation strategy is adopted when HTTP streaming is deployed.

7.1 Bit-rate variation methods

7.1.1 Video coding basics

First, video was captured, stored, and transmitted in analogue form. But with the digital revolution that started over two decades ago, it had to be converted in a format that could be used with the new equipment. When converted from analogue to digital, the quantity of information to represent the new video is huge, thus the necessity for video coding, to reduce the quantity of information required to store or transmit a video sequence. It can achieve this by exploiting redundancy present in video signals, like psycho-visual, coding and statistical redundancy [39]. Psycho-visual takes advantage of the particularities of the HVS, while the code symbols that can be avoided form the coding redundancy. Statistical redundancy refers to the temporal (inter-frame coding) and spatial (intra-frame coding) similarities between adjacent frames.

Intra-frame coding reduces the size of a single picture (I-frame), while inter-frame compression reduces the spatial correlation between multiple frames. At specific intervals the large I-frames are replaced with smaller predicted frames (P frames) and bidirectional frames (B frames) that are generated from a single I-frame. This structure has the disadvantage that if an I-frame is lost or corrupted during the transport process the other linked P and B pictures will not be correctly decoded. For further details about video coding, the reader can check reference [39].

With the adjustment of the encoding rate, the quality of the video can be modified, thus when the bit-rate is reduced, so is the user's experience. However, as shown in [40] it is better to have a lower quality video, than a higher quality one with many lost frames or with re-buffering periods.

The bit-rate can be changed by taking into consideration spatial or temporal characteristics of the video or using scalability properties of some codecs.

7.1.2 Frame skipping

The easiest and one of the first solutions to reduce the bit-rate was to simply drop some of the Predicted frames (P-frames) and Bidirectional frames (B-frames) which would not affect the correct decoding of other pictures in the stream. Such method was used in Darwin streaming server version 6.0.3 and by the authors in [6]. In [41], the authors propose a

more intelligent way of frame dropping by analysing the content and skipping frames from low motion scenes. However, reducing the frame-rate introduces jerkiness and jitter in the video that have an important impact on the perceived video quality [29]. Thus, although this solution is better than allowing congestion in the network, exploiting other characteristics of the video is the preferred way to change the bit-rate since it is less noticeable by the viewer.

This can be achieved by having multiple quality versions of the same video and switching between them, a technique called Bit-Stream Switching (BSS) [6].

7.1.3 Bit-Stream Switching

In this approach, a video sequence is compressed into several quality levels at different bit rates. Therefore, channel bandwidth variation is adapted by dynamically switching among these bit-streams. Because of the temporal prediction, switching at a P or B frame would result in different references at the encoder and the decoder, which would bring the so called drifting error that would propagate to subsequent frames until the prediction chain is cut, for example, by an I frame. So, in order to achieve drift-free switching, some special frames, known as key frames are used to provide the access points to accomplish the switching between versions. Of course, the switch can be made between different quality versions of the video or even between different contents, for example to insert advertising. Fig. 7.2 illustrates this concept.

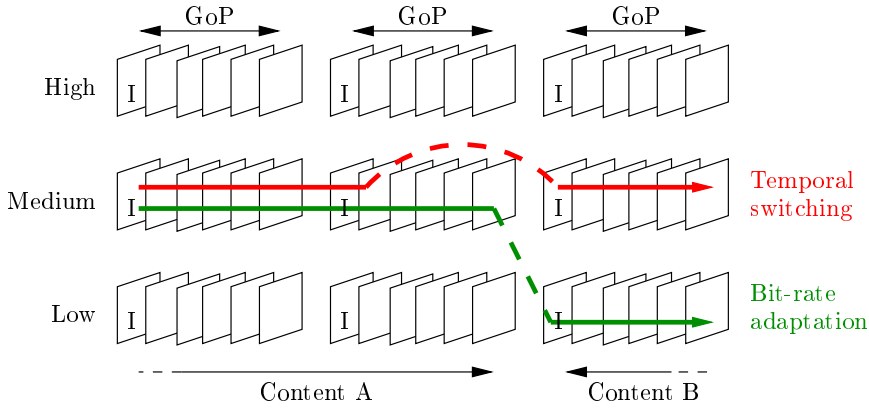

Figure 7.2: Temporal switching and bit-rate adaptation

However, this method usually imposes a trade-off between coding efficiency and switching flexibility, since adding more key frames increases

the video bit-rate but reduces the delay until the switch can be performed. To enable seamless and drift-free changes between quality versions, the switch cannot be accomplished until the arrival of a key frame, which can be an I frame, or SP frame [42] in the case of H.264 codec. When a change in channel bandwidth is detected, the required switching cannot be accomplished until the arrival of a key frame to avoid frame-drops and to enable drift-free switching.

This type of bit-rate variation is widely adopted by the HTTP Adaptive Streaming (HAS) implementations where each quality version is further split into smaller pieces, called chunks. It is required that every video piece must be addressable individually. This can be accomplished by having individual files for every chunk, but it is not entirely required. It is possible to group all chunks in one single file and the accessing mechanisms depend on the implementation solution. For example, Microsoft in its "Smooth Streaming Service" uses one contiguous MP4 file for each quality version to store all the chunks and seeks to the appropriate byte range when a switch request arrives at the server [43]. The authors in [44] use a single 3GPP file format (3GP) to store several pre-transcoded files of different bit-rates, with padding inserted between each encoding. Since the padding size and the storing order is known, the server can seek that file to switch to a different quality.

7.1.4 Scalable Video Coding

Another way to vary the bit-rate while streaming is to use Scalable Video Coding (SVC) techniques which allow to encode content only once with multiple "layers" to deliver video with different quality levels to receiving devices. In a single step and into a single bitstream, quality layers are encoded with different resolutions (Spatial scalability), different picture quality levels or different frame rates (Temporal scalability) and combinations of these features [45], [46]. So, the transmission of some layers can be skipped based on the capabilities of the decoding device or on the bandwidth restrictions of the delivery network. The advantage of this solution is the avoidance of the management and access problems imposed by the manipulations of several videos and saves storage space. The compromise is on coding efficiency since for the same video quality, SVC typically requires 10%-20% more bits compared to a single-quality encoding, because of the intermediate steps. However, the authors of [47] propose a new coding technique to eliminate this overhead.

7.2 Adaptive Streaming based on QoS

This section will review some adaptive streaming solutions that rely only on QoS parameters to estimate the user experience, highlighting the techniques used to detect network state.

7.2.1 Equation-based Rate Control

In [6] and [48] a Binomial Congestion Control (BCC) [49] rate control algorithm is used to deliver the media to the client. Similar to TCP Friendly Rate Control (TFRC), a single formula is used to estimate the suitable sending rate in a TCP friendly manner, while trying to limit TCP inconveniences with video streaming. TCP is not well-suited for real-time audio and video because its reliability and ordering semantics increase end-to-end delays and delay variations [49]. Furthermore, TCP uses the principle of Additive-Increase/Multiplicative-Decrease (AIMD) [50] which means that a TCP connection probes for extra bandwidth by increasing its congestion window linearly with time and it is reducing its window multiplicatively by a factor of two when congestion is detected. This behaviour determines abrupt reductions in transmission rate which do not suit streaming applications. So, instead of using AIMD approach, the authors in [6] use an Inverse-Increase/Additive-Decrease (IIAD) algorithm [49] that increases the transmission rate inversely proportional to the current one and decreases the sending rate in a linear way, when congestion is detected. This means that IIAD is less aggressive than AIMD when increasing or decreasing the data speed which should help in a streaming scenario by avoiding severe transmission variations. Using TCP as the transport protocol has the advantage that it is already "TCP friendly" to other TCP flows. But TCP is "link-fair and not application-fair" [51], so it offers throughput without taking into account the application's needs. By implementing a TFRC or similar rate control at the application level and using UDP as the transport protocol, the "friendliness" can be controlled in a way that would help the streaming application, in the detriment of other TCP flows.

However, this type of approach poses a few inconveniences. First, in RTP streaming, usually done over UDP, the algorithm relies on RTCP feedback to get information about packet loss. Usually, the RTCP sending interval is one packet every 5 seconds, which can be considered quite long. Even more, packet losses suggest that the network is already congested so a less aggressive behaviour of the rate control algorithm worsens the situation. If the algorithm relies on packet loss to indicate network congestion, when packets get lost or corrupted in a wireless environ-

ment, the sending rate is automatically reduced, although this might not be necessary, decreasing its efficiency over noisy channels. To avoid this behaviour, the authors in [52] have proposed to use Early Congestion Notification (ECN) signalling instead of loss rate in the bandwidth formula. To improve the TCP behaviour over wireless networks, service providers can implement different technologies to make the protocol more robust, as presented in [53].

Another issue is related to the media player's buffer occupancy. If the computed rate is lower or higher than the video bit-rate, the buffer at the player side may underflow or overrun in the absence of a reporting mechanism for the client buffer status. For this reason, the solution proposed in [6] needs a media player which sends buffer information through RTSP messages, while [48] needs 3GPP Rel.6 compatible players which send the NADU APP packet described in 4.2.

As bit-rate variation techniques, [6] uses a combination of BSS and frame skipping, while [48] uses BSS for its RTP adaptive streaming solution.

7.2.2 Adaptive streaming based on buffer estimation

Compared to the solutions presented in Section 7.2.1 where a TFRC or similar algorithm is used, in the examples that follow, only the buffers of the client or the network are monitored to adjust the sending rate. In [7] the authors use RTCP feedback together with 3GPP RTCP extensions to monitor the media player buffer level and to estimate the occupancy of the network buffer. The server stores the latest sequence number sent and it knows the latest sequence number received by the client from the RR. Knowing the cumulative size of the RTP packets sent, the server can estimate the fill level of the network buffer as the cumulative size of all RTP packets sent but not received according to the last RR. Using information from the 3GPP extensions, the server can estimate the media player's buffer level. Besides these calculations, the server needs to know the maximum size of the buffers to avoid overfilling them. The 3GPP provides RTSP extensions to send the client's buffer size, but the network buffer size is usually not known a priori. A simpler buffer management technique is the Adaptive Media Playout (AMP) [54] in which the client varies the playback rate, decreasing its speed when the buffer is under a certain threshold and returning to the original speed when the buffer returns above the critical level. This approach may experience problems if the buffer decreases for a long time, since it is not possible to vary the playout speed beyond a certain limit without introducing excessive perceptual distortion.

7.2.3 Adaptive Streaming based on active probing techniques

Another approach is to use tools that compute available bandwidth by explicitly probing the network, sending a few probe packets to the destination and exploiting techniques such as packet pair or packet train dispersion. Such developed tools are Abing [55], Pathchirp [56], Wbest [57] the latter being used by the authors in [58]. The disadvantage of these tools is that they also need a client side that has to analyse the probing packets so a specific media player would be needed. On top of this, the fraction of inaccurate approximations in a mobile network (bandwidth estimation values that lie outside the interval [-15%,+15%]) is higher than 60%, as shown in [59]. The last aspect is that they send additional traffic into the network that is used just for bandwidth estimation.

7.2.4 Early congestion determination

Since packet loss and buffer outage usually occur due to congestion present in the network, some techniques try to discover a congestion situation as soon as possible. In [5] the author tries to determine the access network based on RTT variation, knowing this way which could be the average and maximum throughput. Congestion or signal degradation is detected by measuring the RTT, so there is no need for a particular media player with buffer monitoring. In [60] the author proposes the use of ECN technique on top of UDP so congestion can be addressed before it occurs in the network. Unfortunately the use of ECN depends on the implementation status along the path and since there is no standard for ECN support for UDP, specific signalling has to be defined.

7.2.5 HTTP Adaptive Streaming (HAS)

Since the HTTP protocol is simple, scalable, and not affected by firewalls or Network Address Translation (NAT) traversal issues, progressive download has become the main media transport protocol over the Internet. Apple, Microsoft and Adobe offer proprietary solutions for HAS, while Moving Picture Experts Group (MPEG) and 3GPP organizations work on standardizing Dynamic Adaptive Streaming over HTTP (DASH) which eliminates most of the issues with proprietary techniques.

When HAS/DASH is used to deliver media to the clients, the video is usually divided in consecutive chunks, and each chunk is encoded at various bit rates. Because the media is delivered as fast as the network allows, the transfer rate is easily computed by measuring the time it

Feature	RTP	HTTP
Can be used on top of UDP	YES	NO
Suitable for interactive systems	YES	NO
Trick-play modes (VCR-like controls)	YES	YES (Only in DASH standard)
Possibility to prioritize traffic	YES	NO (HTTP traffic)
Easy firewall and NAT traversal	NO	YES
Can use existing Internet infrastructures (CDNs, cashes)	NO	YES

Table 7.1: Comparison of RTP and HAS features

takes to deliver a video chunk, while the rate control is done by the TCP protocol. This is different from the RTP streaming where using the throughput value to determine the available bandwidth is not feasible since packets are not sent at the maximum network speed, but at a rate close to the encoding speed. Due to this behaviour, in HAS, bit-rate adaptation consists in choosing the video chunk with the encoding rate close to the measured transfer rate obtained for the previous one. Therefore, most of the research work has been done to optimise the storing and access methods of the video chunks, leading to techniques like "chained chunking", "virtual chunks", or "unchunked byte ranges" [61].

HTTP adaptive streaming solves most of the issues that come with progressive download like wasted bandwidth if the user decides to stop watching the content after progressive download has started or live media services. However, being HTTP traffic, HAS suffers from the known TCP issues with media transport, discussed in the beginning of Section 7.2.1. Because the content is stored on the client side, the media player needs a very large buffer, in the order of tens of seconds. In consequence, there is a large delay between the video information being sent and the playback. This works well for video distribution, but it is less suitable for interactive or live video streaming [51].

Thus, for scenarios where the user might frequently interact with the streaming system or where the media is delivered over transport channels that present regular bandwidth variation, we think that RTP adaptive streaming is still the optimal solution. Table 7.1 briefly resumes the advantages and drawbacks of RTP streaming versus HTTP adaptive streaming.

7.3 Adaptive Streaming based on QoE

The methods described in Section 7.2 use different network parameters to estimate the experience of the user. But they do not guarantee that this is exactly the case, because there are several perceptual factors that are involved in a streaming session. For example an algorithm that tries to optimize buffer level, might not detect packet loss which creates image artefacts, or maybe the lost packets affect only some P frames in a GOP and so the degradation in image quality does not require the video bit-rate to be reduced. So, using directly QoE feedback, it would be possible to have a more precise measure of what the viewer is seeing. The QoE can be estimated automatically using VQM algorithms, as discussed in **Part II, Chapter "Quality of Experience"**. Although there is a lot of research work to design new NR-VQM metrics, there is limited study on how to use those metrics to adapt the video rate. For example, in [62] the authors use a Pseudo-Subjective Quality Assessment (PSQA) metric to control best-effort background traffic in order to satisfy the service requirement of real-time video streams. In [63] a Reduced Reference form of the Video Structural SIMilarity (VSSIM) metric is used to efficiently allocate the network resources for video delivery in LTE mobile networks. Instead of trying to maximise the QoE for each user, the objective function aims to maximize the average perceived quality of all users by jointly optimizing the application layer and the lower layers of the radio communication system.

However, an adaptive system based only on QoE measurements will only be good in detecting a deterioration of the service, but it will not be able to tell if the network allows to increase the quality of the streamed video, without interfering with the user's perception.

Chapter 8

Proposed Solution

8.1 Problem statement and contribution

As discussed in Chapter 7, stream adaptation is not a new research topic, several solutions have been proposed, some of which already became standards. Nevertheless, only a few simple adaptation mechanisms were implemented on commercial streaming servers due to the lack of extended RTCP support in the common media players. However, even if most of the content providers adopted HTTP as their delivery technique, there is still a need for efficient RTP adaptive streaming, especially for interactive or live content. On the other hand, a true estimation of the user experience can only be obtained by using video quality measurements at the client side, but at this moment such standards do not exist so a specific implementation is needed. So the research question stated in Section 1.4 can be reformulated as:

"How to design and implement an adaptive streaming solution that improves the QoS and the QoE in an RTP streaming session? Once the algorithm is defined, which are the main parameters that need to be tuned for an efficient operation in wireless environments? What feedback mechanisms need to be used to report all the necessary elements while keeping compatibility with a large number of media players?"

The solution proposed to answer this question is to develop a layered adaptive streaming algorithm that can be used with any popular media-player like VLC, QuickTime, and with a proprietary player that offers QoE reporting capability. The combination of the two types of metrics would increase the speed and the accuracy of the adaptation algorithm: QoS based adaptation only takes into consideration delivery problems but could detect in advance a possible situation that would lead to image degradation; QoE based adaptation offers more accurate measurements,

but would detect problems when the image quality is already affected and it is not well suited to discover whether the network allows an increase in quality.

One of the latest trends in multimedia streaming is to offer interactive services where the user selects on the fly from multiple angles of view or multiple Regions Of Interest (ROI). Interaction delay is then critical for the user experience. So, to keep delay to a minimum, RTP streaming should be used. Since the RTP standard specifies feedback messages through RTCP reports, the adaptation strategy needs to be server centric. The main idea for QoS adaptation is to assess whether the network conditions support the current streaming rate or a higher rate, using only the standard RTCP feedback received from the client. Based on the estimated network conditions, the server would increase or decrease the video bit-rate. Although this approach is similar to the ones published so far, its main advantage is that it does not need information about the buffer state, network condition being estimated from the RTT deviation and packet loss, as described in Sections 2.3.1 and 2.3.2. The first adaptation layer would take into consideration the QoS network parameters, as reported by the client through RTCP messages and will be active all the time. Since the basic RTP/RTCP standard is supported by a large number of media players, this will enable stream adaptation for many applications. The second adaptation layer would take in consideration the MOS value obtained through a NR-VQM method, sent by the proprietary player, so it will be active only when a compatible client is used. The architecture is presented in Fig. 8.1

One original contribution of this thesis is the design of a complete solution, which combines QoS and QoE measurements to offer rate adaptation. The second contribution consists in the use of a new network probing mechanism, that sends in advance the video frames and analyses the delay variation that results from the additional load induced in the network.

Part of the material presented in Chapters 8-12 has already been published in [64], together with Laurent Schumacher and Christophe de Vleeschouwer.

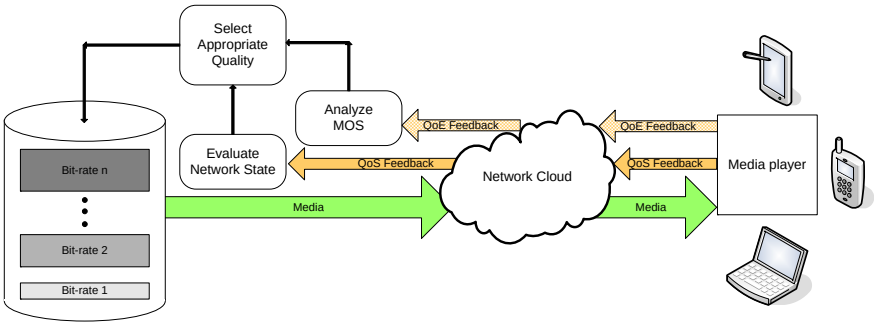

Figure 8.1: Overview of the adaptive streaming chain with the QoS and QoE components

8.2 Adaptation State Machine

Depending on the action it takes, the behaviour of the server can be best described as a Finite State Machine (FSM), as depicted in Fig. 8.2. Several *Normal* states and two additional phases can be observed: *Initialisation* and *Probing*.

8.2.1 Initialisation State

This is the first phase of the algorithm, it includes the RTSP negotiation and network discovery, when the server collects statistics about the current state of the network. The first two RTCP reports are used for the initialisation of $Smooth_{RTT}$ and $Deviation_{RTT}$. There are two possibilities regarding the initial quality of the delivered media:

- The user has the choice of selecting the appropriate video quality suited for his/her network capabilities. Even if the initial client's selection is not optimal, the proposed adaptation scheme will help the server to eventually adjust and send the video encoded at a rate that best matches the available network bandwidth.

- To be on the safe side, the server can start streaming at the lowest quality rate, then increase it until congestion is detected. This is similar to Slow Start in TCP and has the advantage of lower delay until the playback begins.

8.2.2 Normal X_k State

In this state the server sends the media at a constant rate of X_k kbps, where $k \in \{1..N\}$ and N is the number of supported bit-rate versions.

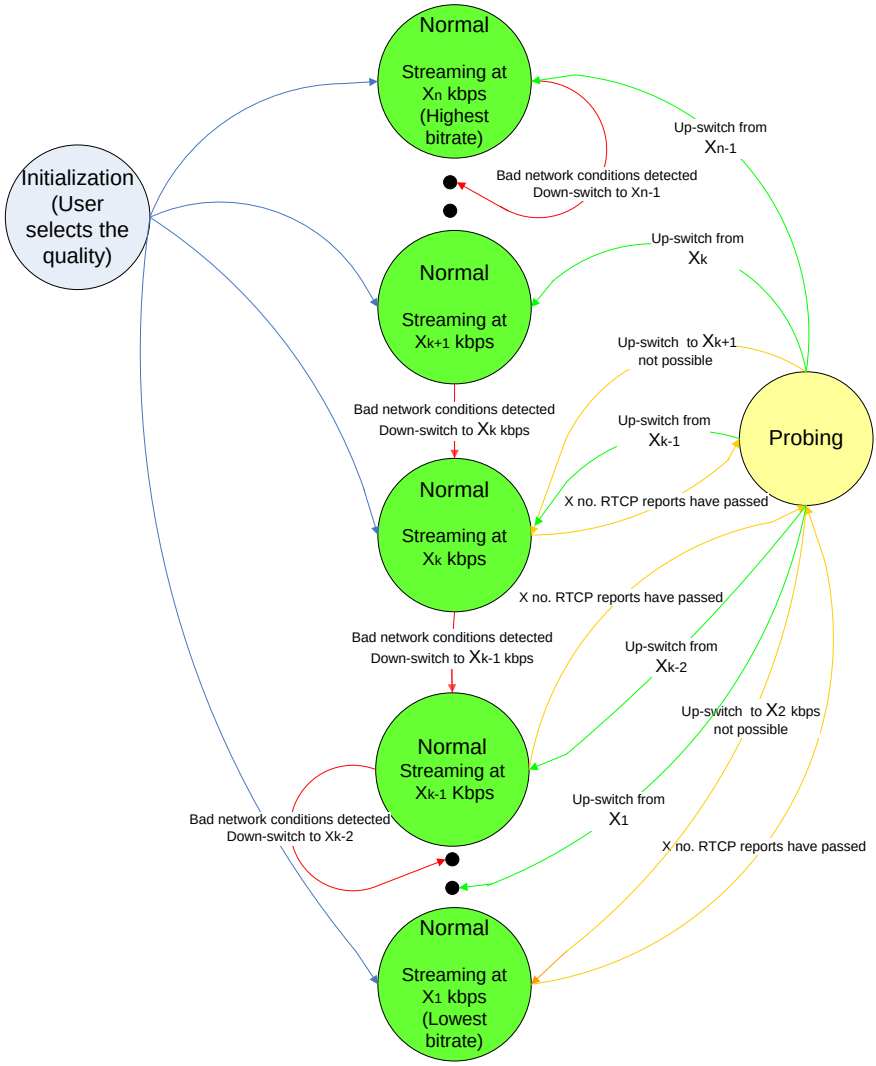

Figure 8.2: Adaptation algorithm

The streaming quality is monitored by analysing the RTCP reports. If the QoS or QoE measurements reach their lower level thresholds, the server will immediately start to stream a lower encoded version of the content, going into Normal X_{k-1} state. Therefore a "down-switch" is defined as the reduction of the encoding quality. If the network is stable the server will go into the *Probing* state where it tries to determine if the available bandwidth is high enough to sustain the streaming of a higher bit-rate. If the *Probing* state indicates that it is possible to stream a higher encoding, the server will move to X_{k+1} state (up-switch), streaming the next available quality level. Consequently, an "up-switch" is defined as the increase of the encoding quality. To resume, two types of transitions can be noticed:

- Down-switch - this is when the server goes from X_k to X_{k-1} state, which is equivalent to a reduction in the video bit-rate.
- Up-switch - takes place when the server goes from X_k to X_{k+1} state, after passing through the Probing state. When an up-switch occurs, the bit-rate of the streamed video is increased.

To define possible bit-rates for the X_k states, a Fast bit-rate Decrease Slow bit-rate restore Up (FDSU) approach can be used. According to [60], this method is the most suitable way to assess network congestion. As the name suggests, when congestion is detected, the server should abruptly decrease the video quality to approx. 60% of the current encoding. Although high oscillations in bit-rate are not recommended because the decrease in quality is easily observed by the user, this approach limits the number of future packet losses due to congestion, which would have greater impact on video quality. This technique implies that false positives should be kept to a minimum, otherwise the user experience can be heavily affected. However, a slow bit-rate increase implies producing many quality versions of the same content, which puts a burden on the post processing and storage of several versions. Consequently we will rather use a Fast bit-rate Decrease Fast bit-rate restore Up (FDFU) approach. The total number of states, N, can be unlimited in theory. However, from a practical point of view, it can not be very large, depending on the bit-rate variation method used. For example, if SVC is chosen, the number of states represent the number of scalable levels. If real-time transcoding is implemented, then, in theory, an endless number of quality steps could be obtained, but in reality this is limited by the processing power required when several sessions are streamed in parallel. If several pre-encoded versions for the same content are used, the production and management of different files becomes important. Typically, we would

recommend to switch between three *Normal* states. For example in a cellular environment, each *Normal* state would be defined to encompass the average rate of a cellular generation, e.g. EDGE (140 kbps), UMTS (200 kbps) and HSPA (350 kbps), as measured in [5].

8.2.3 Probing State

This is an auxiliary state, since the video content keeps on being streamed to the client at the reference rate X_k kbps. However, in this case silent gaps and bursts of RTP packets alternate in order to estimate the available network bandwidth. The main idea behind this technique is to temporarily send the video frames at a higher rate (burst) to put the network under stress. If the bandwidth limit is close to the current bit-rate, the packets sent at a higher rate will queue in the network buffers and the RTCP reports will feed-back high RTT values at the server. Consequently, from those RTT values, it can assess whether the available network bandwidth is high enough to switch to a higher bit-rate. If this is the case, then the server goes from X_k state to X_{k+1} state. If the probing result indicates that there is not enough available bandwidth to increase the video quality, it will resume regular streaming in the X_k state.

Chapter 9

Down-switch conditions

The system goes from one state to another based on the values of the parameters received from the media player through the RTCP reports. It is necessary to determine which are the thresholds that trigger a change of state.

The value of these parameters influences the behaviour of the system, which can vary from an aggressive to a more relaxed one. In the first case, the server responds too fast to the slightest sign of congestion or image degradation. This seems to be the desired action, but the bit-rate could be reduced even when it is not necessary, affecting user's experience. In the other extreme case, the system waits too long before taking an action, which could lead to severe image degradation.

9.1 QoS Down-switch thresholds

When congestion is about to appear on the transport path, the delay will begin to increase, as the packets are held in network buffers. An example of a congestion situation in a streaming session is shown in Fig. 9.1 where the formulas in Eq. (2.1) and Eq. (2.3) are plotted, along with the instantaneous RTT. The NetEm Linux module was used to reduce the network bandwidth approx. 20% under the streaming rate, for a limited time period. The instantaneous RTT increases rapidly, along with the Deviation_{RTT} and the Smooth_{RTT}, returning to their regular values when congestion is over. It can be seen that the deviation can take negative values as well, which means that the delay is currently decreasing.

To see how congestion affects the deviation values, several tests were made in a controlled environment, where the bandwidth was reduced using NetEM in Linux to different levels: from 1.2 times higher than the

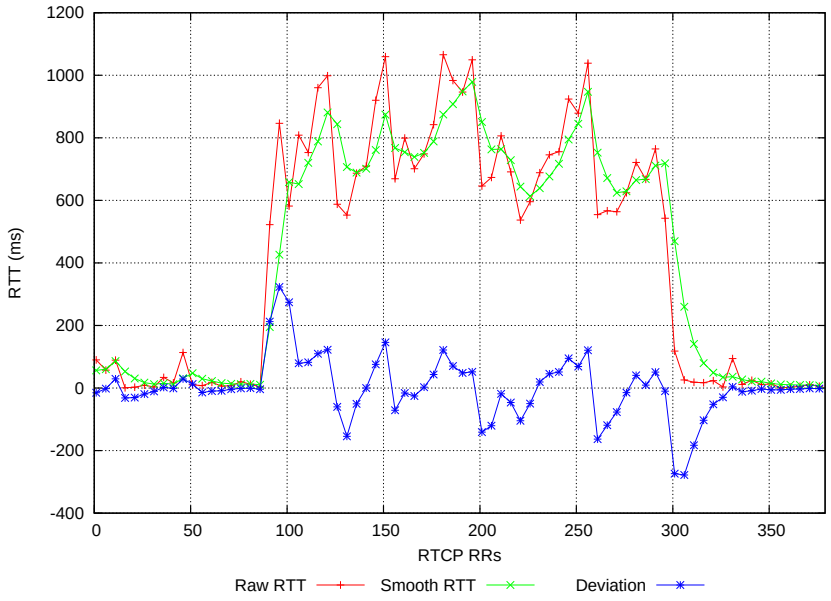

Figure 9.1: RTT evolution in an artificially congested network. Bandwidth reduction applied after approximately 80 s.

current rate to 90 percent of the current streaming rate. From the Cumulative Distribution Function (CDF) plotted in Fig. 9.2 it can be observed that when streaming close to the bandwidth limit, up to 1.2 times the current rate, the RTT deviation is affected, although it is still possible to receive the media without perturbation. When the bandwidth is lower than this limit, the deviation is very high, meaning the delay increases with time, as the packets are queued in the network buffers. Fig. 9.3 shows the evolution of deviation when different levels of bandwidth reduction have been applied. In the cases of 1.2 times the current rate, or higher bandwidth limit, the deviation increases for the first 2 RTCP reports, but then decreases to a value close to zero. This means that the RTT does not continue to increase with a high rate, so the current video quality is still supported by the network and therefore a down-switch is not necessary. In the other cases, the deviation increases rapidly meaning that congestion is persistent and the video bit-rate should be reduced to avoid packet loss.

To determine the events that cause a down-switch, it is also important to know what is the RTT deviation in different networks under normal conditions and under congestion. To this end, measurements in live 3G LTE networks [65, 66, 67, 68] were used to feed the formulas in (2.1)

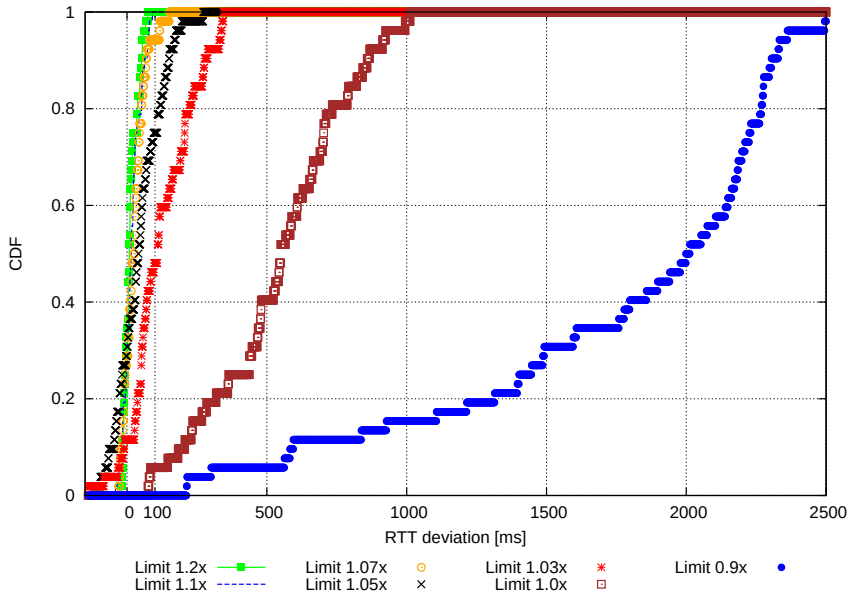

Figure 9.2: CDF of the RTT deviation for different congestion levels.

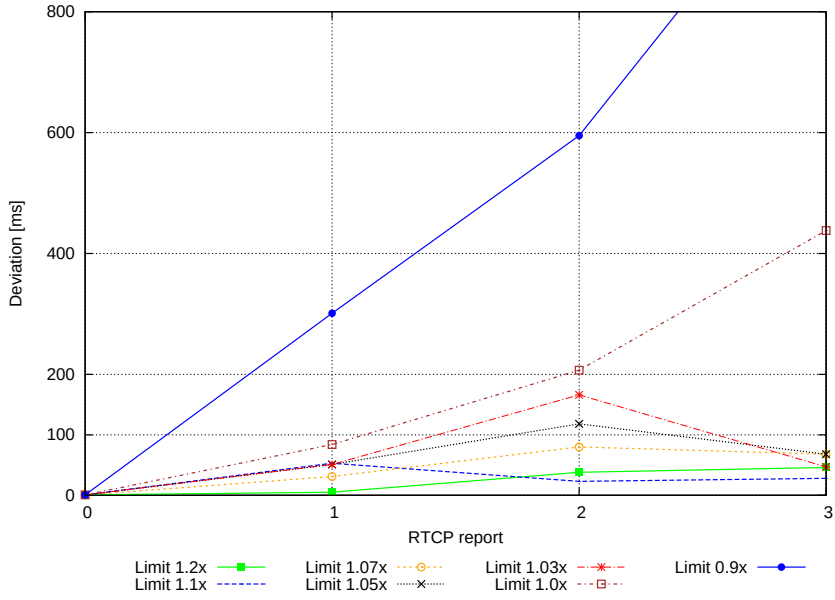

Figure 9.3: Deviation evolution after bandwidth limitation

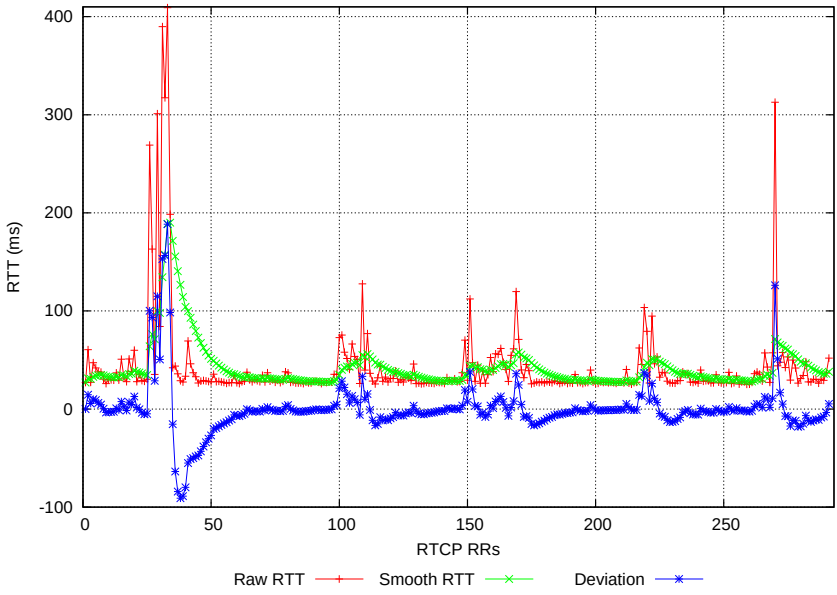

Figure 9.4: RTT evolution in a LTE network.

and (2.2). Fig. 9.4 shows the evolution of the RTT and deviation in a streaming session performed in a LTE network. The first spike that can be observed, where the deviation is well above 100ms, is produced by a short period of congestion. The next delay spikes on the graph are produced by hand-overs, as the user moved from one base station to another [68]. In those cases the Deviation_{RTT} never goes above 100ms.

It can be shown that the Deviation_{RTT} only goes above 90-100ms when the current transmission rate is close to the maximum available bandwidth, but it remains under this value in absence of congestion, even in the case of a GPRS connection, whereas the jitter is higher than in other mobile networks. Also, the authors of [69] have reported that in 90% of the cases, the jitter was smaller than 100 ms in their measurements.

Consequently, if the absolute Deviation_{RTT} value is higher than 100 ms, this should be interpreted as a sign of congestion. Since it is not desired to down-switch the bit-rate too early, it is recommended to wait for the next RTCP report to see if the deviation value is still greater than 100ms, which would confirm that the congestion is persistent. For the sake of accuracy the number of RTCP RRs taken into account should be higher. However, when the media client supports a standard implementation of the RTP/RTCP protocol (like VLC or QuickTime) it sends RTCP re-

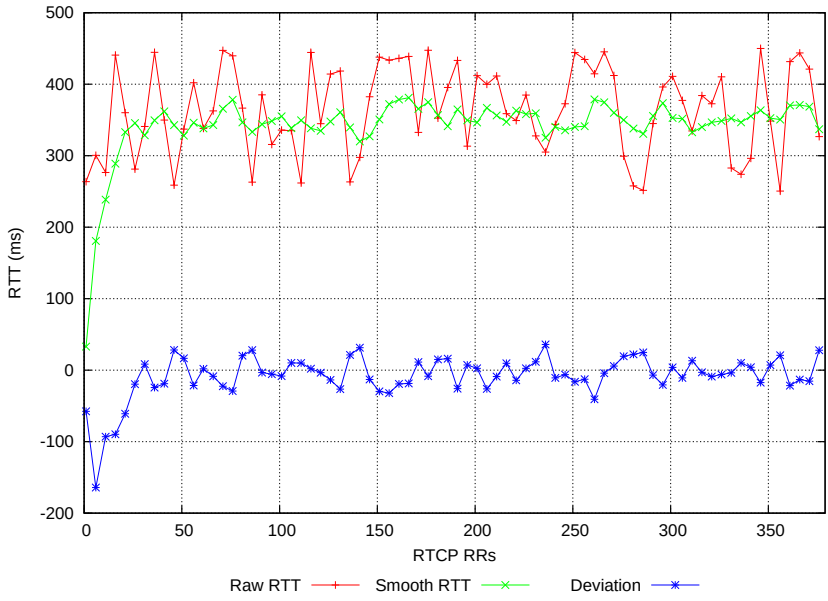

Figure 9.5: RTT evolution in GPRS network.

ports every 5 seconds. Waiting for more than two reports would therefore lead to a reaction time longer than 10 s, which is not acceptable.

There are though two particular cases when the 100ms limit for the deviation is not optimal.

- Severe congestion - In this case, waiting for 2 or more consecutive RTCP reports is not feasible, a faster response is needed to avoid packet loss or re-buffering. Severe congestion is reflected in very high RTTs, resulting in a high deviation, as it can be seen in Fig. 9.3. If the deviation is higher than 300ms, the server immediately switches to a lower bit-rate.

- After down-switch - According to the FDFU approach, the bit-rate is reduced to approx. 60% of the current encoding. After this event, the reported RTTs will still indicate a high deviation since packets are still queued in network buffers and might trigger a new down-switch, although this would not be necessary. The other case is when the current bandwidth still can not support the reduced rate, and a new down-switch is needed. To distinguish between these cases, several tests have been performed to observe the evolution of the RTT deviation after the bit-rate reduction. Again, the Linux NetEm module was used to limit the bandwidth

Bandwidth Limitation	0.8x	0.7x	0.65x	0.6x	0.5x
Average $\left(\frac{Dev_1}{Dev_{Down-Switch}}\right)$	1.73	1.71	2.82	2.26	2.06
σ (standard deviation)	0.2	0.33	0.75	0.77	0.48

Table 9.1: The ratio between first RTT deviation after down-switch and the RTT deviation that caused the down-switch

Bandwidth Limitation	0.8x	0.7x	0.65x	0.6x	0.5x
Average $\left(\frac{Dev_2}{Dev_1}\right)$	0.65	1	1	1.53	1.53
σ (standard deviation)	0.17	0.11	0.12	0.31	0.16

Table 9.2: The ratio between the second RTT deviation and the first RTT deviation after the down-switch

to 0.8, 0.7, 0.65, 0.6 and 0.5 times the current streaming rate. To quantify the evolution of the deviation after down-switch, the fraction between the current deviation and the previous one is used. Table 9.1 gives the average of the ratio between the first RTT deviation after the bit-rate reduction and the deviation that triggered the down-switch. It can be seen that when the bandwidth is reduced to 0.8 or 0.7 times the current rate, the average increase factor is approx. 1.7, with a not so large σ. However, for 0.65x, 0.6x and 0.5x, although the average ratio is around 2, the σ is quite large. Therefore, looking just at the first RTT deviation after the down-switch does not enable to say with confidence if the newly reduced bit-rate is supported by the network.

Table 9.2 shows the increase ratio of the second RTT deviation, compared to the first value after the down-switch. When the bandwidth is reduced to 0.8 times the current rate, the deviation decreases, while in the 0.7 and 0.6 it remains constant. It keeps growing when the network can not support the newly reduced rate, in the case of 0.6x and 0.5x limit. So, these limits can be used to assess if the current rate can be kept, or a new reduction is necessary. Therefore, the video bit-rate will be reduced one more time if the inequality (9.1) is true. More important is that σ is low enough in all cases to be confident in these intervals.

$$\frac{\text{Second Deviation}}{\text{First Deviation}} \geq 1 \tag{9.1}$$

Besides the RTT deviation, the server takes into consideration the packet loss ratio as well. Since the wireless environment is a lossy one, it is possible to experience a small amount of losses even if the network can

sustain the current streaming rate. Based on the experiments performed in **Part V**, the loss rate limit between RTCP reports has been set to 10% but is taken into consideration only if the total number of lost packets is higher than 10. We experienced satisfactory behaviour with these values in a wide range of wireless access networks, from WiFi to LTE.

9.2 QoE Down-switch thresholds

When the QoE adaptation layer is enabled, the streaming server will periodically receive MOS reports which show the quality of the received video. As explained in Section 3.2.3, the QoE adaptation strategy is based on the NR-VQM algorithm implemented by R.Henkes in [4]. The values vary from 10 to 100, 10 meaning "Bad" video quality while 100 being "Excellent" on a subjective measurement scale. Since the algorithm detects discontinuities in the decoded video, the MOS score does not depend on the encoding quality, so it will show a 100 score for all the bit-rates, as long as playback fluidity is not affected by network or other decoding issues. To better understand the reported MOS values, they can be mapped over a subjective measurement scale, bearing in mind that the correlation factor is 0.9 [4] (on a 0 to 1 scale). Table 9.3 shows the correlation between MOS scores generated by the NR-VQM algorithm and the subjective video quality scale.

The NR-VQM algorithm behaviour has been tested in different scenarios taking into consideration two QoS elements that can influence the user experience: bandwidth and packet loss. These tests were performed in [4] and some of the results are shown in Fig. 9.6 and Fig. 9.7. For both tests, the average encoding bit-rate of the streamed video was approx. 650 kbps, so looking at Fig. 9.6 it can be seen that even when the bandwidth limitation is much higher than the video rate, there are moments when the MOS score drops to a value of 70 for a short period. This means that there is some slight image degradation, but is not annoying for the viewer and a down-switch is not required. The same

Reported MOS	Subjective Scale	Video Quality	Impairment Perception
100	5	Excellent	Imperceptible
75	4	Good	Perceptible but not annoying
50	3	Fair	Slightly Annoying
25	2	Poor	Annoying
10	1	Bad	Very Annoying

Table 9.3: Corresponding video quality and impairment perception for the NR-VQM computed MOS score

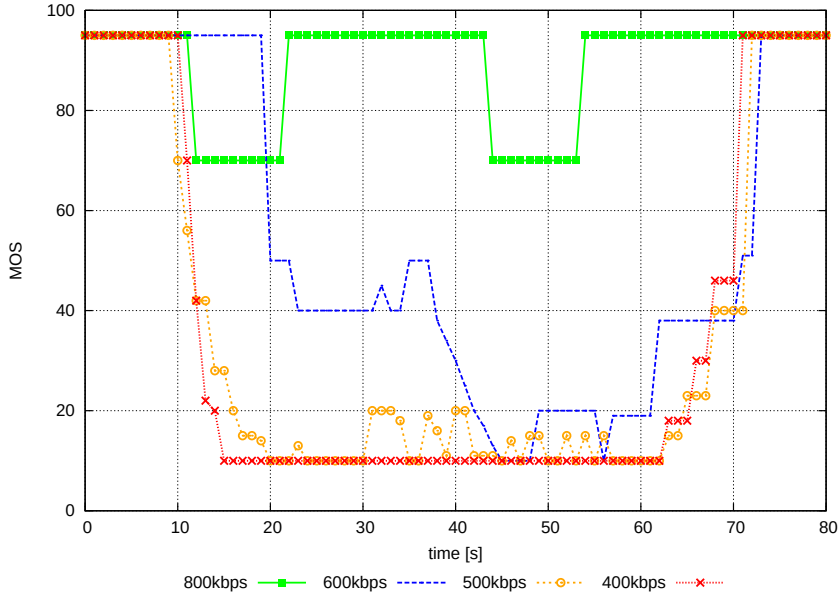

Figure 9.6: MOS scores for different bandwidth limitations [4]. Video bit-rate was 650kbps

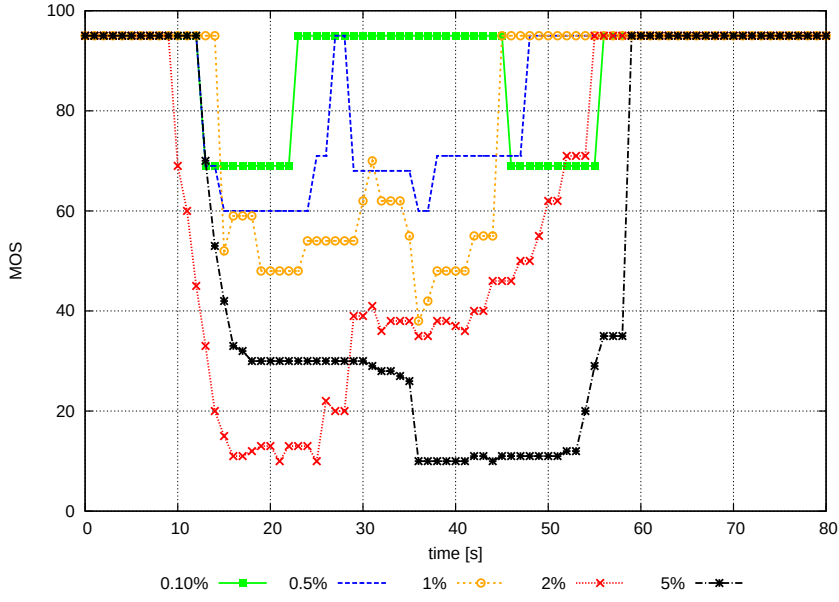

Figure 9.7: MOS scores for different packet loss ratios [4]

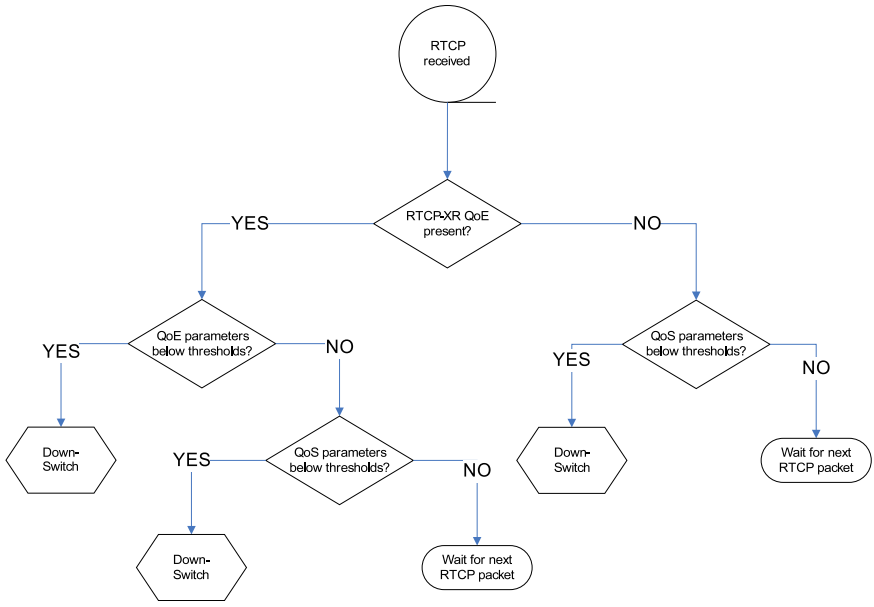

Figure 9.8: Server behaviour at the receipt of a RTCP packet

conclusion can be drawn from Fig. 9.7, case "a" and case "b" where the losses created some deterioration for a short period, but again a bit-rate reduction is not required. To avoid an aggressive adaptation behaviour, a down-switch should be performed only when the reported MOS is less than 50, which means lower than "Fair" video quality.

The QoE adaptation layer includes the effects of packet loss and network jitter, offering a precise estimation of the user experience. However, QoS adaptation is still active and can detect potential congestion before any image degradation occurs. Fig 9.8 illustrates the behaviour of the streaming server when a RTCP packet is received.

Chapter 10

Up-switch conditions

To experience the best video quality, the user has to receive the highest possible bit-rate allowed by current network conditions. Sections 9.1 and 9.2 discussed when to reduce the bit-rate when the network can not support the current streaming rate. But the adaptation process has to be performed in both directions, so this section will analyse in which circumstances the video quality can be raised.

The main challenge in finding the right moment to increase the bit-rate is that a simple analysis of the QoS parameters can not be used to assess whether the network supports a higher video rate, as shown in [5]. The easiest way would be just to raise the video quality and then if it is not supported by the network, the adaptation mechanism will reduce it back. However, frequent quality changes are disturbing for the viewer. Even more, this approach can have the opposite effect, it can reduce the user experience or can even cause the playback to stop if the quality increase happens when the streaming rate is close to the bandwidth limit.

To overcome this issue, a probing mechanism has to be used to send additional data to stress the network in a completely transparent way for the user. In return, the QoS parameters indicate whether the network supports the additional load. Specific tools that use active probing have been designed to determine the available bandwidth, but have certain disadvantages, as shown in Section 7.2.3. For that reason, a new probing technique is proposed, that sends in advance useful video packets from the current media stream and then analyses the effects of the probing on the RTT deviation. The advantage of this technique compared to the tools that compute the available network bandwidth by sending packet trains or packet chirps, (for instance Abing [55], Pathchirp [56] or Wbest [57]) is that it does not send extra data over the network, since

the RTP packets would have been sent anyway. In addition, it does not require the deployment of a dedicated client application to analyse the probing traffic.

10.1 Network probing design

The scope of the probing is therefore to determine if the network supports the streaming of a higher quality content. Ideally, it should give an estimation of the exact value for the available bandwidth as dedicated tools do, but this requires a client with a specific implementation which would limit the choice of media players. So, a different approach is needed. According to the FDFU concept, the next encoding level should be about 60% higher than the previous one, so basically an up-switch should be made only if the available bandwidth is greater or equal to 60% of the current streaming rate. However, since streaming closely to the bandwidth limit could lead to higher RTTs and packet loss ratios as shown in Section 9.2, we aim to up-switch only when the bandwidth limit is almost twice as high as the current streaming rate. Hence, frequent quality switches will be avoided. Even more important is to prevent switching to a higher bit-rate that is not supported by the network, as it can induce severe congestion, leading to a reduced user experience.

By sending data into the network at a faster rate, packets will get queued in buffers along the network path and the time spent in the queues will be reflected in the RTCP reports. The extra latency introduced by the probing mechanism will depend on the amount of available bandwidth, so analysing this delay can give an estimation of the available bandwidth.

The best practice would be to send packets at an increasing rate until the network limit has been reached. Unfortunately, the amount of data which can be sent at a faster rate is limited due to the risk of client buffer overflow. To overcome this issue, the burst of RTP packets has to be followed or preceded by a pause in the transmission, the so-called gap. This allows the data from the buffer to be consumed, or to refill the buffer to its average occupancy respectively. For example, the NR-VQM tests showed that VLC and GStreamer allow gaps shorter than 1 second without a decrease in the MOS score. Gaps higher than 1s translate into a reduction of the MOS value, suggesting that the buffer was emptied in the process. These results were consistent across different encoding bit-rates, ranging from 120Kbps to 1.5Mbps.

The respective positions of the burst and the gap depend on the risk one wants to mitigate. If one chooses to have the burst first, followed

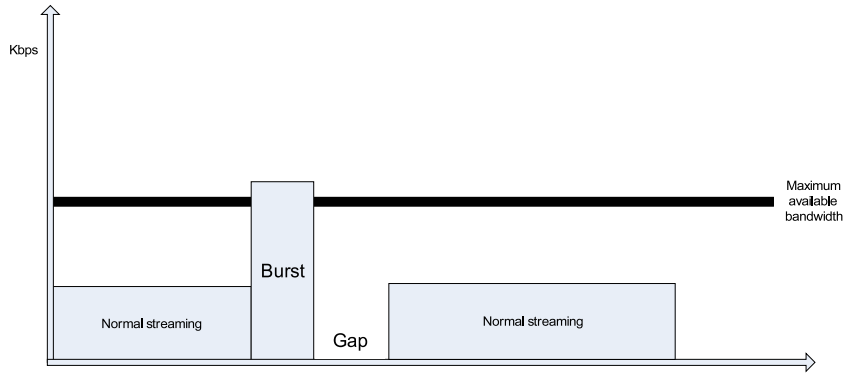

Figure 10.1: Probing design starting with a burst followed by a silent gap

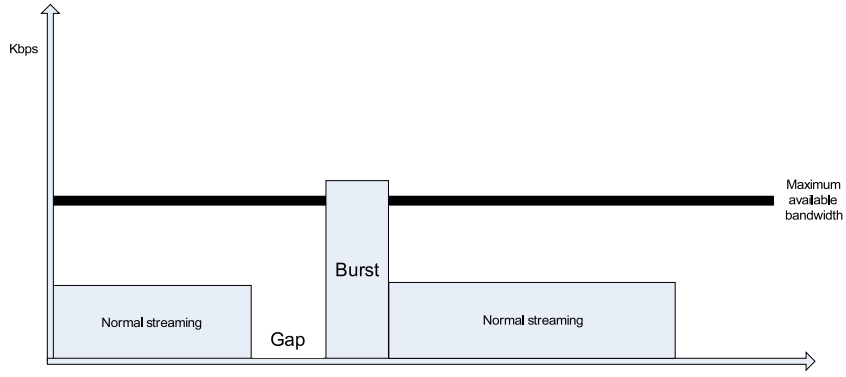

Figure 10.2: Probing design starting with a silent gap followed by packet burst

by the gap, we minimize the risk of starvation at the client. An empty buffer forces the playback to stop during re-buffering, which leads to a degraded user experience. On the other hand, a full buffer adds delay in an interactive streaming scenario. Indeed, in the absence of a mechanism to remotely flush the client buffer, the player will consume the old content received during a burst before switching to the display of the new requested content. The two possible designs are presented in Fig. 10.1 and Fig. 10.2

10.1.1 Probing options

To prevent buffer under-run or buffer overflow, the length of the gap is strictly related to the burst characteristics and is computed as:

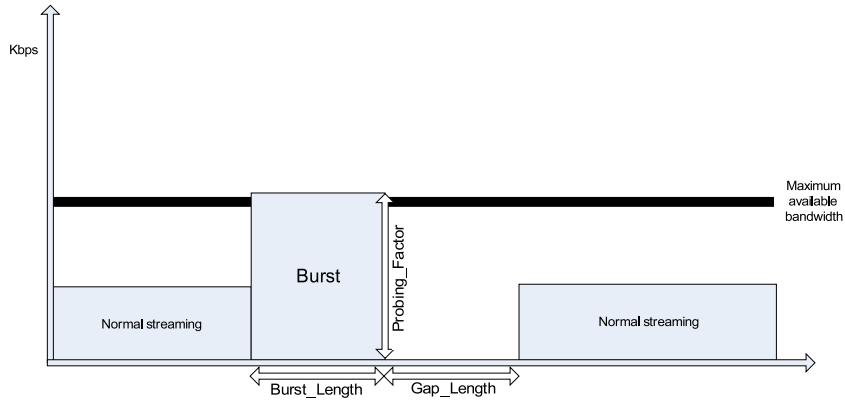

Figure 10.3: Burst and Gap parameters

$$\text{Gap\_Length} = \frac{\text{Burst\_Length}}{FPS} - \frac{\text{Burst\_Length} - 1}{FPS * \text{Probing\_Factor}} [seconds] \tag{10.1}$$

$$\text{Burst\_Length} = \frac{\text{Gap\_Length} * FPS * \text{Probing\_Factor} + 1}{FPS - 1} [frames] \tag{10.2}$$

where Burst_Length represents the probing duration expressed in number of frames, FPS is the video frame-rate expressed in fps and the Probing_Factor represents the frame rate increase (for example 2 times, 3 times the original value). From Eq. (10.1), the Burst_Length can be calculated as shown in Eq. (10.2). These parameters are illustrated in Fig. 10.3.

In a N fps video, one frame is displayed for $\frac{1}{N}$ seconds and the increase of the sending rate by a factor of Probing_Factor is equivalent to reducing the frame duration by the same factor. For this reason, the Burst_Length can be expressed in time units using the formula in (10.3)

$$\text{Burst\_Duration} = \frac{\text{Burst\_Length}}{FPS * \text{Probing\_Factor}} [seconds] \tag{10.3}$$

10.1.2 Parameter values

The elements that determine the level of induced congestion in the network are the Probing_Factor and the Burst_Length. As stated, an up-switch should be initiated when the available bandwidth is at least twice

as high the current rate, so when probing, the RTP packets should be sent two times faster. This means that in Eq. (10.2) the Probing_Factor will be equal to 2. The Burst_Length has to be as high as possible to obtain a noticeable effect, so the Gap_Length has to be as high as possible, but below the limit that would cause a buffer underflow.

Considering common media players like VLC or GStreamer, the Gap_Length has to be less than 1 second to avoid buffer problems. Hence, for a Probing_Factor of 2, a gap of 970ms and a typical 25fps video, the Burst_Length is 49 frames or 980ms. If the Probing_Factor is increased to 4, keeping the same gap, the Burst_Length becomes equal to 32 frames or 320ms. This probing technique has been implemented in an existing open source project, the Live555 Media Server. Using the Linux NetEm module, a bandwidth limit 1.9 times higher than the current rate was set and several streaming sessions were ran with the probing mechanism enabled. This 190% limit was set in detriment of the 200% because for a Probing_Factor of 2, due to a limited Burst_Length, the packets may not be delayed in the network buffers, so the RTT would not be affected. Two sets of tests were run, one with a Probing_Factor of 2 and the other with a Probing_Factor of 4, as the scope of these experiments was to investigate the relation between the Probing_Factor and the RTT deviation. It seems that using a larger value with shorter Burst_Length has a more visible impact on the network than a smaller increase factor, with a longer duration. Fig. 10.4 shows the network response in terms of RTT deviation to the probing action, higher deviation values being observed when a Probing_Factor of 4 is used. This behaviour is confirmed when looking at the CDF plotted in Fig. 10.5 where it can be seen that there is a larger probability to observe higher deviation when a Probing_Factor of 4 is used. The parameter values that will be used in the final adaptation scheme, are summarised in Table 10.1:

Gap_Length	Burst_Length	Probing_Factor
970 [ms]	$\frac{3.88*FPS+1}{FPS-1}[frames]$	4

Table 10.1: Probing parameters

The fact that not every burst and gap combination produces an increase in the RTT deviation has two possible causes. First, the RTT computed by the server from a RTCP RR message represents the time spent between the source and destination for that RTCP RR and the most recently sent RTCP SR packet. For a reporting interval of 5s it is possible that the RTCP SR packet is sent at the end of the gap period, so by that time the network buffers might have emptied, resulting a low

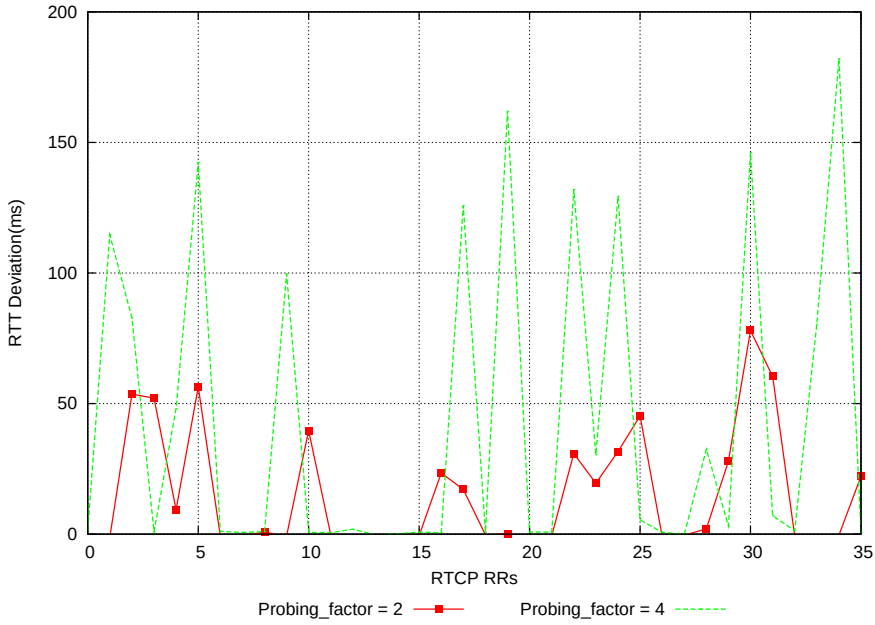

Figure 10.4: RTT deviation evolution for two different values of the Probing_Factor with a bandwidth limit of 1.9 times the nominal rate

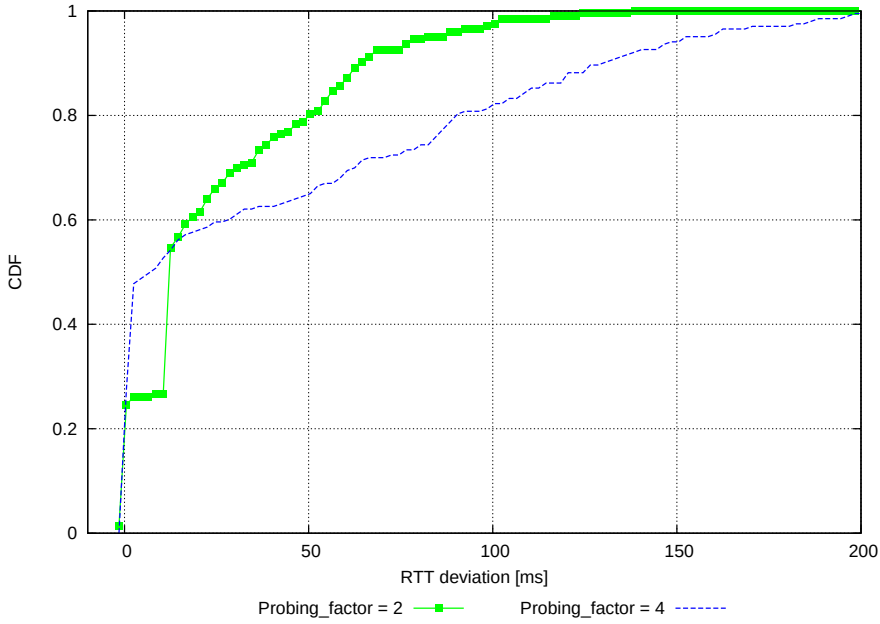

Figure 10.5: CDF of the RTT deviation for two different values of the Probing_Factor

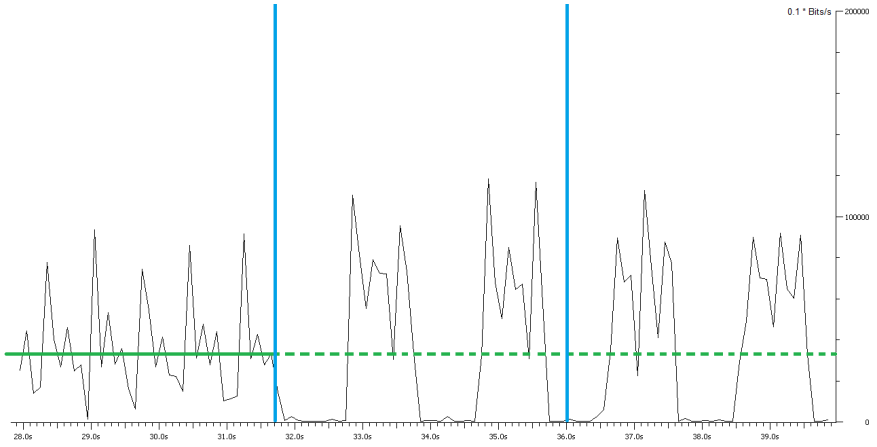

Figure 10.6: Throughput sample of a streaming session with probing enabled after 31.7 seconds. Probing_Factor = 2

RTT. Secondly, because RTP packets are used as probing data, the size of each packet is not constant, so the amount of data sent in each burst is not exactly the same each time. This can be observed in Fig. 10.6 and Fig. 10.7, where the burst shape is slightly different for each cycle. Marked with a horizontal line is the average throughput, while the vertical bars mark the arrival of the RTCP RRs.

10.1.3 Probing cycle

The probing cycle can be defined as the number of burst and gap combinations that are run before the probing ends. Considering the parameters discussed in the previous section, one burst-gap combination has a duration of about 1,290ms for a Probing_Factor of 4 and 1,950ms when the Probing_Factor is set to 2. This means that if the probing cycle is equal to 1, there would be approx. 3.7s or respectively 3s before the arrival of the next RTCP RR. By this time the network buffers may have cleared or the network bandwidth could have changed after the probing cycle. For this reason, the burst-gap combination should be repeated several times to increase the chance that a RTCP packet is queued in the network buffers. The probing cycle is determined therefore by the length of the gap-burst combination and the reporting interval for the RTCP feedback, as shown in Eq. (10.4).

$$\text{Probing\_Cycle} = \frac{\text{Reporting\_Interval}}{\text{Burst\_Duration} + \text{Gap\_Length}} [seconds] \quad (10.4)$$

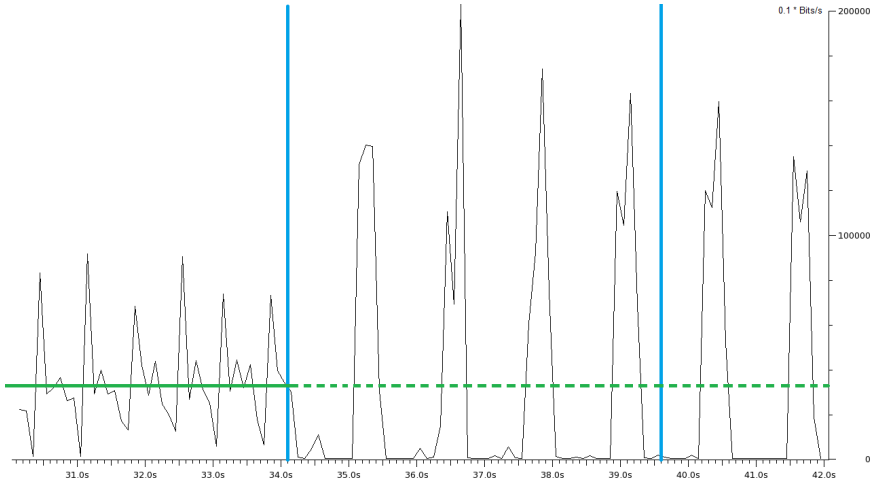

Figure 10.7: Throughput sample of a streaming session with probing enabled after 34.1 seconds. Probing_Factor = 4

For better accuracy, the probing cycle can be maintained for the duration of two or more reporting intervals, but this would increase the server's response time to bandwidth variations. A good compromise between speed and precision is achieved when two RTCP reports are taken into consideration, but if a more aggressive behaviour is desired, only one report should be used.

Due to the extra load put on the network and buffer limitations, special care should be taken when choosing the moment to start probing, especially if the available bandwidth is low. This is why if possible congestion or losses are detected, the server will continue to remain in the *Normal* state until better conditions are indicated by the RTCP reports.

10.2 Determining Up-switch threshold

After each probing cycle, the server would decide whether to up-switch or not, based on Deviation_{RTT} derived from the RTCP RRs.

As explained in Section 10.1 we aim to stream the next video quality level only when the available bandwidth is twice as high as the current encoding rate. Several tests have been performed in a QDisc limited Ethernet network to observe the RTT deviation when the probing is done under different bandwidth limits. Six scenarios have been considered where the bandwidth was respectively limited to 2, 1.85, 1.7, 1.5, 1.35 and 1.2 times the current streaming rate and the RTT deviation was

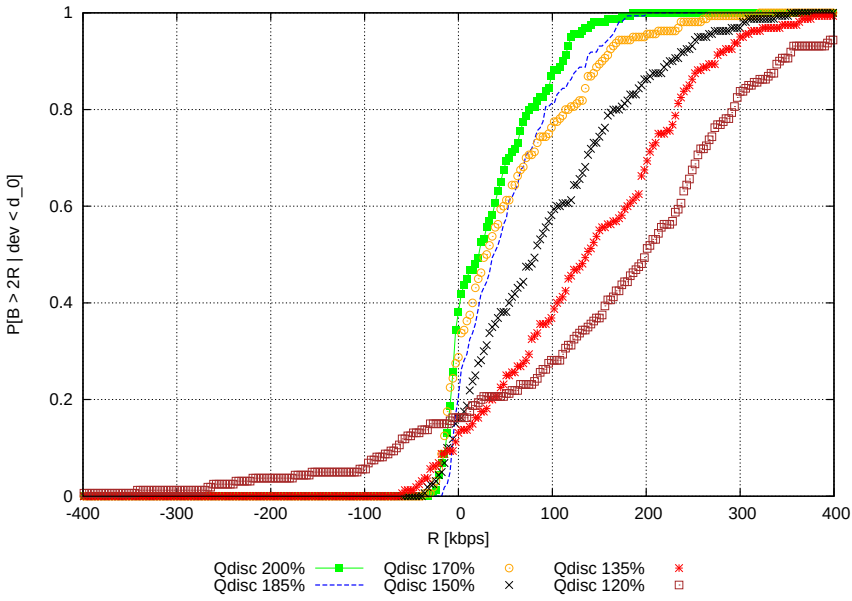

Figure 10.8: CDFs for the six scenarios

collected during each probing cycle. The CDF is plotted in Fig. 10.8 with data gathered from approximately one hour of streamed video for each case.

Looking at Fig. 10.8, the Deviation$_{RTT}$ can be seen as a function of the bandwidth margin. For instance, when probing under a limit two times higher than the current rate, the deviation is under 100 ms in 90% of the cases. Table 10.2 shows the cumulative probability to observe a given Deviation$_{RTT}$ considering the bandwidth margin:

Bandwidth Margin	Probability Dev = 50 ms	Probability Dev = 100 ms	Probability Dev = 200 ms
120%	.2	.3	.5
135%	.3	.4	.7
150%	.4	.6	.9
170%	.6	.75	.95
185%	.6	.8	1
200%	.7	.85	1

Table 10.2: Probability to observe a specific deviation in each bandwidth limited scenario from Fig. 10.8

Next, we derive the closed-form expressions of the probability that

the network bandwidth is indeed higher than twice the streaming rate given a Deviation$_{RTT}(dev)$.

Let T_0 be the outage probability that the available bandwidth B is greater or equal to twice the bit rate of the video sequence R. We would then look for the deviation threshold d_0 such that

$$P\left[B \geq 2\,R | dev \leq d_0\right] \geq T_0 \tag{10.5}$$

Conversely, one could set the deviation threshold d_0 and compute the outage probability T_0. From the definition of CDF we therefore know.

$$\begin{aligned} P\left[dev \leq d_0 | B = B_i\right] &= \int_{-\infty}^{d_0} T_{dev|B=B_i}\left(dev\right) ddev & (10.6)\\ &= cdf_{dev|B=B_i}\left(d_0\right) & (10.7)\end{aligned}$$

We can regard those six scenarios as a sampling of the frequency domain, so the average CDF can be written as:

$$\begin{aligned} &P\left[dev \leq d_0\right] \\ &= P\left[dev \leq d_0 | B = B_1\right] P\left[B < B_{1,2}\right] \\ &\quad + \sum_{i=2}^{5} P\left[dev \leq d_0 | B = B_i\right] \\ &\quad\quad P\left[B_{(i-1),i} \leq B < B_{i,(i+1)}\right] \\ &\quad + P\left[dev \leq d_0 | B = B_6\right] P\left[B \geq B_{5,6}\right] \end{aligned} \tag{10.8}$$

where

$$B_{i,j} = \frac{B_i + B_j}{2} \tag{10.9}$$

Returning to (10.5), we can write

$$P\left[B \geq 2\,R | dev \leq d_0\right] = 1 - P\left[B < 2\,R | dev \leq d_0\right] \tag{10.10}$$

This last conditional probability can be transformed thanks to Bayes formula into

$$\begin{aligned} &P\left[B < 2\,R | dev \leq d_0\right] \\ &= \frac{P\left[B < 2\,R\right]\, P\left[dev \leq d_0 | B < 2\,R\right]}{P\left[dev \leq d_0\right]} \end{aligned} \tag{10.11}$$

In (10.11), $P\left[B < 2\,R\right]$ depends on the wireless set-up under consideration. Extrapolating from downstream UDP throughput from [70], one could possibly model the available bandwidth from UDP streaming

as an exponential distribution parametrised to C, the nominal capacity of the wireless set-up, such that

$$P\left[B < 2\,R\right] = 1 - \exp\left[-\left(\frac{3}{C}\right)\,2\,R\right] \tag{10.12}$$

Based on relations (10.6-10.8) and on the bandwidth model (10.12), we would get

$$\begin{aligned}
& P\left[dev \leq d_0 | B < 2\,R\right] \\
&= P\left[dev \leq d_0 | B = B_1\right] P\left[B < B_{1,2}\right] \\
&\quad + \sum_{i=2}^{5} P\left[dev \leq d_0 | B = B_i\right] \\
&\quad\quad P\left[B_{(i-1),i} \leq B < B_{i,(i-1)}\right] \\
&\quad + P\left[dev \leq d_0 | B = B_6\right] P\left[B_{5,6} \leq B < 2\,R\right] \qquad (10.13) \\
&= \left\{1 - \exp\left[-\left(\tfrac{3}{C}\right)\,1.275\,R\right]\right\}\, cdf_{dev|B=B_1}\left(d_0\right) \\
&\quad + \left\{ \begin{array}{l} \exp\left[-\left(\frac{3}{C}\right)\,1.275\,R\right] \\ -\exp\left[-\left(\frac{3}{C}\right)\,1.425\,R\right] \end{array} \right\}\, cdf_{dev|B=B_2}\left(d_0\right) \\
&\quad + \left\{ \begin{array}{l} \exp\left[-\left(\frac{3}{C}\right)\,1.425\,R\right] \\ -\exp\left[-\left(\frac{3}{C}\right)\,1.6\,R\right] \end{array} \right\}\, cdf_{dev|B=B_3}\left(d_0\right) \\
&\quad + \left\{ \begin{array}{l} \exp\left[-\left(\frac{3}{C}\right)\,1.6\,R\right] \\ -\exp\left[-\left(\frac{3}{C}\right)\,1.775\,R\right] \end{array} \right\}\, cdf_{dev|B=B_4}\left(d_0\right) \\
&\quad + \left\{ \begin{array}{l} \exp\left[-\left(\frac{3}{C}\right)\,1.775\,R\right] \\ -\exp\left[-\left(\frac{3}{C}\right)\,1.925\,R\right] \end{array} \right\}\, cdf_{dev|B=B_5}\left(d_0\right) \\
&\quad + \left\{ \begin{array}{l} \exp\left[-\left(\frac{3}{C}\right)\,1.925\,R\right] \\ -\exp\left[-\left(\frac{3}{C}\right)\,2\,R\right] \end{array} \right\}\, cdf_{dev|B=B_6}\left(d_0\right) \qquad (10.14) \\
& P\left[dev \leq d_0\right] \\
&= P\left[dev \leq d_0 | B < 2\,R\right] \\
&\quad + cdf_{dev|B=B_6}\left(d_0\right) P\left[B \geq 2\,R\right] \qquad (10.15) \\
&= P\left[dev \leq d_0 | B < 2\,R\right] \\
&\quad + \exp\left[-\left(\frac{3}{C}\right)\,2\,R\right]\, cdf_{dev|B=B_6} \qquad (10.16)
\end{aligned}$$

We can plot the up-switch probability based on throughput distribution in different access networks in Fig. 10.9. For a deviation $d_0 = 50$ ms, an up-switch has a 90% success rate provided the streaming rate R is lower than 300 kbps in 3G networks, 500 kbps in WiMAX scenario, 1 Mbps in WiFi b and 5 Mbps in WiFi g. Considering a sequence at 1 Mbps streamed on a 3G network, the upswitch has a 30% success rate if the observed deviation is up to 200 ms, whereas this rate increases to 40% if the deviation is as low as 50 ms.

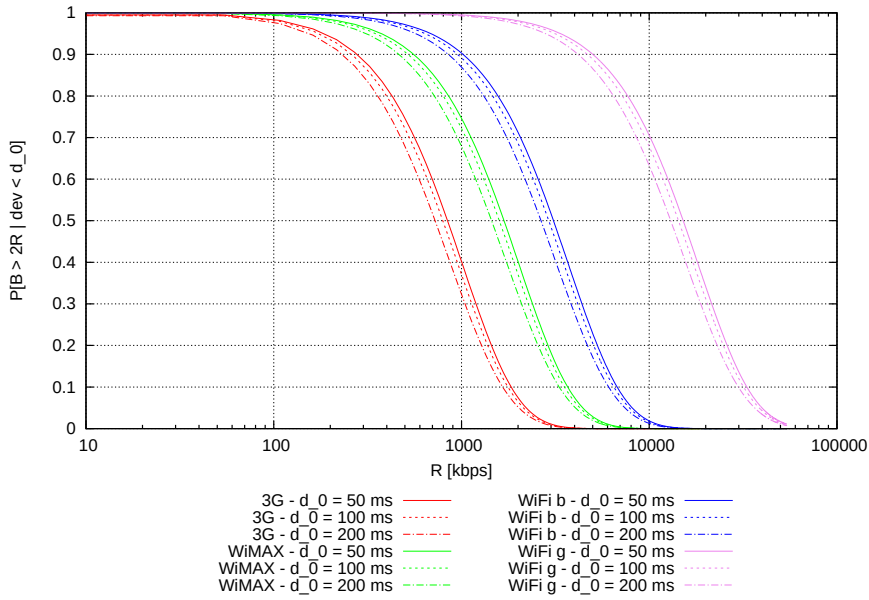

Figure 10.9: Probability that there is enough bandwidth to up-switch s.t. observed deviation in different access networks

When probing under a limit two times higher than the current rate, the deviation is under 100 ms in 90% of the cases, so if a higher deviation is observed, the bandwidth limit is below the 200% limit. For all our tests, we have considered the deviation up-switch threshold of 100 ms because it offers the best trade-off between a low percentage of false positives and a high success up-switch rate. It also matches the down-switch threshold.

10.3 Conclusion

Chapter 7 introduced adaptive streaming and the influencing factors of the design of such systems. Some reference rate adaptation solutions have been presented, emphasising the techniques used to evaluate user experience along with their possible limitations. A rate adaptation algorithm for RTP video streaming has been proposed in Chapter 8. It is based on a layered approach that takes into consideration both QoS and QoE parameters, with the goal of detecting bandwidth fluctuations as fast and as accurate as possible. For this reason, RTT deviation is the main parameter used in detecting network problems because when packets cannot be delivered at the desired rate, they will be delayed in

network buffers.

Several tests have been made to analyse the RTT and therefore the RTT deviation in different bandwidth limited, streaming scenarios. The obtained results along with measurements made in various wireless environments have led to the conclusion that an RTT deviation higher than 100ms indicates the presence of congestion. The use of this threshold determines a balanced adaptation strategy, between an aggressive behaviour and a more relaxed one in a wide range of wireless environments. This choice will be validated by the tests presented in Chapter 12. Unfortunately, detecting an increase in the available bandwidth is not as straightforward, so an original network probing mechanism has been proposed that works with the standard implementation of the RTP/RTCP protocol. This technique is based on the network's response in terms of RTT deviation to the additional load introduced by an RTP packet burst. To avoid overfilling or emptying the player's buffer, the burst has to be paired with a pause in the transmission, called gap. The burst and the gap are related and basically, the larger the client buffer, the higher is the load that can be put on the network. A higher load can be obtained either by increasing the transmission rate or by keeping the increased rate for a longer time. This would help the probing accuracy by increasing the chance that RTCP packets are queued together with the RTP ones, especially for long RTCP reporting intervals which can go up to 5 s.

The up-switch threshold for the RTT deviation has been determined in an empirical way, after running a set of experiments in several bandwidth limited scenarios. The 100 ms limit set in Section 10.2 is valid only when used in combination with the probing parameters selected in Section 10.1.2. Hence, if the Burst_Length and the Probing_Factor values are changed, the RTT deviation response will not be the same and a new up-switch threshold should be determined.

For better accuracy and to identify a broader range of problems that can affect user experience, a NR-VQM algorithm has been used to compute the video quality on the client side. The resulting MOS score was sent back to the server through RTCP-XR compound packets, to trigger a decision based on the MOS value. The QoS and the QoE layers are designed to work together, a higher priority being given to the video quality score: even if the QoS parameters indicate good network conditions, the server will make a down-switch if MOS values are below the threshold.

Part V

Performance and Evaluation of the Algorithm

Chapter 11

Performance evaluation

As shown in previous chapters of the thesis, rate adaptation is needed to counteract the effects of bandwidth variation in a streaming session. The main role of such algorithms is to follow as closely as possible the network throughput evolution, in a transparent manner for the viewer. Ideally, bandwidth fluctuations should be detected before they happen, while the changes in video quality should be done seamlessly with infinitesimal steps. In practice however, there is a delay between the moment a significant change in throughput occurs and the moment the system reacts to it. So the reaction time can be regarded as a performance indicator. Another performance criterion can be the accuracy with which bandwidth variation is followed. This is given mainly by the number of quality versions available for a video and the bit-rate spectrum they cover. In this case, the FDFU paradigm has been adopted, with 3 different encoding qualities for each video, as explained in Section 8.2.2. Because the proposed algorithm includes a stochastic component, the number of false positives and true negatives needs to be taken into consideration as a performance rule. Two other important aspects have also been taken into consideration, like the fairness between the users and detecting image quality degradations that are not network related. Table 11.1 presents a summary of the completed experiments, along with the section were they can be found.

The experimental setup contained four access networks and two kinds of tests:

- Network Stress - the effects of bandwidth variation and congestion are analysed
- CPU Stress - the media player shares processing resources with a CPU stress application

Access Network	Network Stress		CPU Stress	
	QoS data	QoE data	QoS data	QoE data
WiFi	Sec.12.3.1	Sec.12.2.2	Sec.11.3.3	Sec.11.3.3
WiMax	Sec.12.2.1	Sec.12.2.2	Sec.11.3.2	Sec.11.3.2
LTE	Sec.12.4	Sec.12.4	X	X
Wired (QDisc)	Sec.11.1, 11.2, 11.4	Sec.9.2	Sec.11.3.1	Sec.11.3.1

Table 11.1: Experiments summary

11.1 Down-switch delay

The first series of experiments tested the reactivity of the algorithm in the case of congestion detection (network stress), using the parameters presented in Chapter 9. The set-up consists two PCs, one hosting the modified version of a LiveMedia555 streaming server and the other hosting VLC media player, connected via 100 Mbps Ethernet interfaces. The available bandwidth between the two can be reduced using the Linux Traffic Control tool, to simulate congestion or signal degradation in wireless networks. During several streaming sessions the available bandwidth was reduced close to, or below the current streaming rate and the reaction time between bandwidth reduction and an actual down-switch was registered. The CDF for the delay is plotted in Fig. 11.1, where three scenarios have been considered:

1. When the bandwidth drops to a value closer to the current streaming rate, the down-switch delay is in average equal to 10s, which represents the duration of approximatively two RTCP reporting periods. This behaviour is to be expected since in this case, congestion is not serious and the RTT increases slowly. This slower reaction time is desired because if the congestion disappears, a reduction in video quality is avoided.

2. When the bandwidth drops to a value at least 20% lower than the current streaming rate, the system reacts faster, in about 5-6s, which represents the duration of about one RTCP reporting period. This can be associated to a severe congestion or a sudden drop in signal quality. The smaller reaction time is explained by the fact that the RTTs are increasing much faster, so the deviation threshold of 300ms is reached, resulting in an earlier down-switch. Of course, this is the desired behaviour because eventual packet loss is avoided or reduced.

3. When the bandwidth drops during the probing phase, the down-switch delay is approximatively 10s as well, again the span of two

Down-switch	BDW $\simeq$ current rate	BDW $<$ current rate	While probing
Average delay	$\sim$2 RTCP RR (11.4s)	$\sim$1 RTCP RR (6.4s)	$\sim$2 RTCP RR (10.5s)

Table 11.2: Average reaction time in detecting congestion

RTCP reporting periods. This particular case has been taken into consideration because when probing, higher RTT variations are expected, so they are not interpreted as congestion signs. Typically, the next RTCP report after the probing period will indicate a RTT deviation that is over the thresholds, triggering a down-switch.

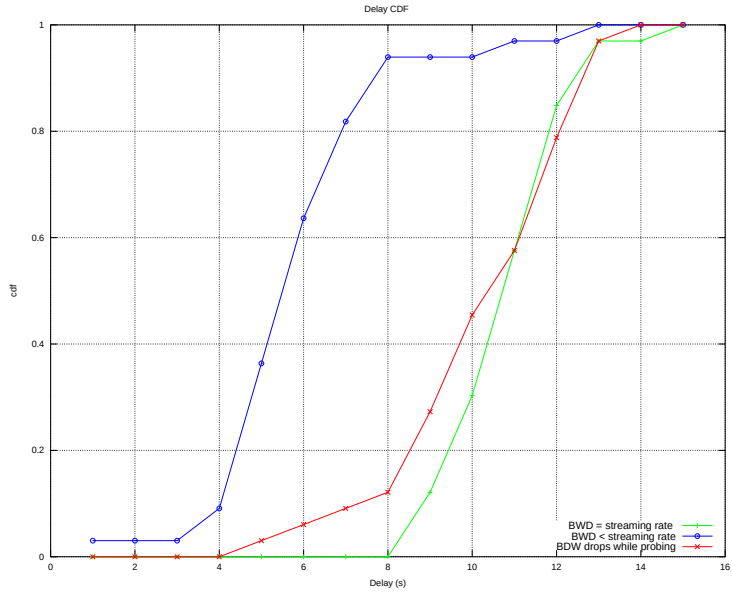

Figure 11.1: CDF for the down-switch delay

The reaction time directly depends on the frequency of the RTCP reports, and the results obtained from these experiments present the worst case, though realistic scenario, where the media player sends RTCP RRs every 5 seconds. The down-switch delay performance is summarized in Table 11.2.

Bandwidth limit(Qdisc)	120%	150%	165%	195%	200%
Up-switch success	16%	49%	51%	71%	83%

Table 11.3: Up-switch success rate for different bandwidth limits

11.2 Up-switch delay

The up-switch delay represents the time interval between the moment the available bandwidth increases to enable the streaming of a higher bit-rate and the moment the new video quality is streamed to the client. It depends on several factors:

- Probing frequency - how often is the "probing cycle" repeated. This delay can be minimised or eliminated by probing all the time, with the exception when congestion is detected.
- Probing cycle - in Section 10.1.3 it was set to 2 RTCP reporting intervals. So, in the worst case scenario when the RTCP messages are sent every 5 seconds, this component adds a delay of approximatively 10 seconds.
- Success rate - is given by the number of successful up-switches for a specific bandwidth limit, as shown in Table 11.3.

Looking at the 195% and 200% bandwidth limits, we deduce a failure rate of 29% and 17% respectively. This means that in average, once every 5 probing cycles the server might not up-switch, although it should, in this case the up-switch delay being increased to 2 probing cycles, equivalent to approximately 20 seconds. For 120%, 150% and 165% bandwidth limits, one can consider the success rate as false positives. Table 11.3 shows that in the worst case scenario, when the available bandwidth is only 1.2 times higher than the current rate, an incorrect up-switch is triggered in 16% of the cases. This is equivalent to the appearance of severe congestion, so the next RTCP report will trigger a down-switch. In the other two cases, an erroneous up-switch does not have such bad consequences since it is similar to the presence of light network congestion. This can be fixed with a down-switch or, for example, a mobile user may be entering an area with better signal coverage so the new rate could be further supported.

These performance results can be compared to the ones achieved by some commercial HAS solutions. The authors of [71] have performed an experimental evaluation of some popular HAS implementations in a bandwidth constrained scenario. For example, the *Microsoft Smooth*

Streaming player reduces the video rate with a 25 s delay after the bandwidth limit has been applied. A similar time interval passes until the bit-rate is raised to a higher level after the bandwidth limitation has been removed. It looks like a more relaxed adaptation approach is used by Microsoft, made possible by a very large buffer size, of about 30s. This allows to keep the same quality for a longer period, without the fear of emptying the player buffer. The Netflix media player has a similar behaviour, but having a much larger buffer, of approximatively 300s, it can receive a bit-rate higher than the available bandwidth for a longer period of time.

11.3 QoE improvement through MOS

The previous two sections evaluated the performance of the algorithm in response to bandwidth variation. It is possible though that the communication channel is in optimal conditions to transport the media, but the user may experience a low quality playback. This can happen if the client device does not have the necessary computing power to decode the streamed video, as might be the case with lower-end mobile devices. The following experiment envisions such a scenario, where the QoE adaptation layer detects a drop in image quality, even if the QoS parameters are way below their thresholds.

11.3.1 Wired scenario

The experiment set-up, similar to the one in Sections 11.1 and 11.2, consists in two PCs, the client and the streaming server, connected through a 100MBps Ethernet link. On the client PC, the Linux `"stress"` command is running to ensure that the processor is utilised at 100% while the GStreamer media client plays the streamed content. Since the processing resources will be shared between the media player and the `"stress"` command, the playback will not be smooth and the NR-VQM plug-in will detect the drop in image quality. For this test, three bit-rates were considered: the highest quality version encoded at 2.5Mbps, the next lower version encoded at 1.8Mbps and the lowest quality video with a bit-rate of 800Kbps[1]. Three scenarios were prepared:

- No down-switch performed, the server always streaming the highest available bit-rate (2.5Mbps)

[1] In this case the encoding rates do not respect the FDFU rule to highlight the effect of a down-switch.

- 1 down-switch has been performed, from 2.5Mbps to 1.8Mbps
- 2 consecutive down-switches, from 2.5Mbps to 1.8Mbps and finally to 800kbps

Fig. 11.2 plots the MOS scores for each of the three scenarios. The `"stress"` command was launched when the second RTCP Receiver Report arrived at the server and was running for the whole duration of the experiment. It can be seen that in all three cases the image quality drops when processor is under stress, the playback rate being approximatively 1 frame per second. A slight improvement in the MOS score can be observed when the bit-rate has been reduced to 1.8Mbps, while a further reduction to 800Kbps improves the user experience even more. The two down-switch moments are marked on the figure with vertical bars.

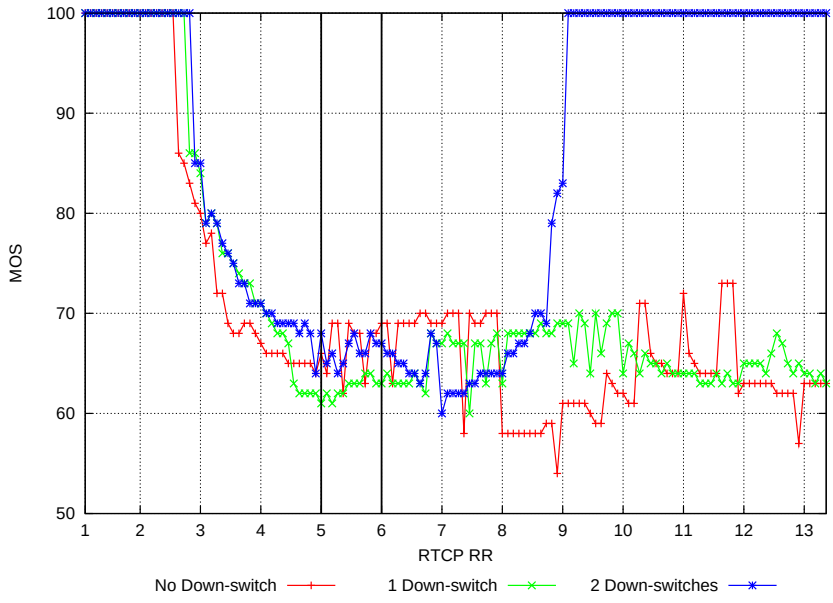

Figure 11.2: Evolution of the MOS score for the 3 test scenarios in the wired access network

To be noticed is that during this experiment there was no packet loss and the RTT deviation was close to zero, so without the QoE adaptation layer, the user experience could not have been improved.

For consistency reasons, the same experiment has been repeated with a WiMax and WiFi network connection.

11.3.2 WiMax scenario

In Fig. 11.3 it can be observed that the evolution of the MOS score is similar in the WiMax and in the wired case: the user experience is slightly improved when the bit rate is reduced. This is to be expected since the network performance does not influence the streaming quality in this scenario.

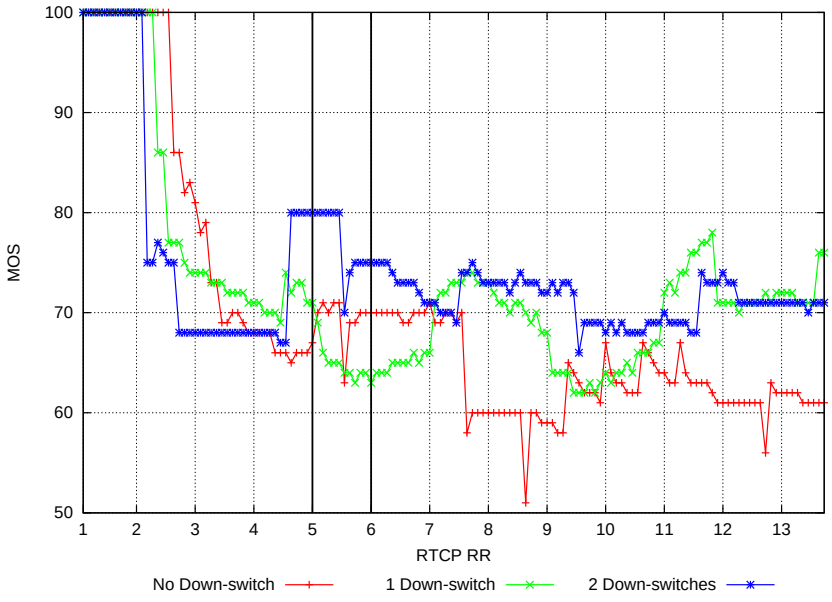

Figure 11.3: Evolution of the MOS score for the 3 test scenarios in the WiMax access network

11.3.3 WiFi scenario

The results from the WiFi tests are not as consistent as the wired and WiMax ones. There are a couple of reasons for that:

- The video bit-rates used in this case had to be lower because the WiFi connection could not support the same encodings as in the WiMax and the wired case. Therefore, the following bit-rates have been used: 350 kbps, 240 kbps and 180 kbps. Since the processing resources needed for decoding these bit-rates is lower, the reported MOS is higher and the switch between the different rates does not visibly affect the MOS score.

- Random packet loss was present during the streaming session (~10 packets lost per session) which affected the MOS score.

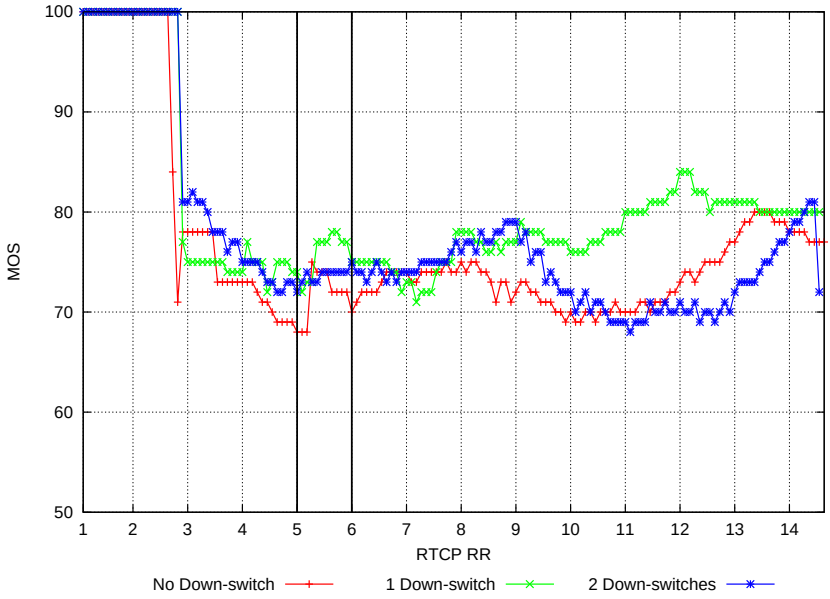

Figure 11.4: Evolution of the MOS score for the 3 test scenarios in the WiFi access network

11.4 Fairness between concurrent flows

This section analyses the behaviour of the adaptation algorithm when the bandwidth is shared between 2 concurrent streaming sessions. For this reason the following experiment has been set-up: considering the FDFU approach, on the streaming server, 3 video bit-rates were available: 350 kbps, 240 kbps and 180 kbps. Two media sessions were started while the available bandwidth between the sender and the receivers was limited to 650 kbps, allowing the parallel stream of one 350 kbps and one 240 kbps video.

The throughput graphs in Fig. 11.5 show that the server favours an equal distribution of video quality between the clients as the throughput curves for the 2 clients present similar variation patterns. The disadvantage is that the available bandwidth is not efficiently used, the total average throughput obtained being 490 kbps, so an average of 75% bandwidth utilisation. We can compare the obtained value with the ideal

case, when one client would receive the 350 kbps video bit-rate while the other would get the 240 kbps encoding which would determine a 90% bandwidth utilisation.

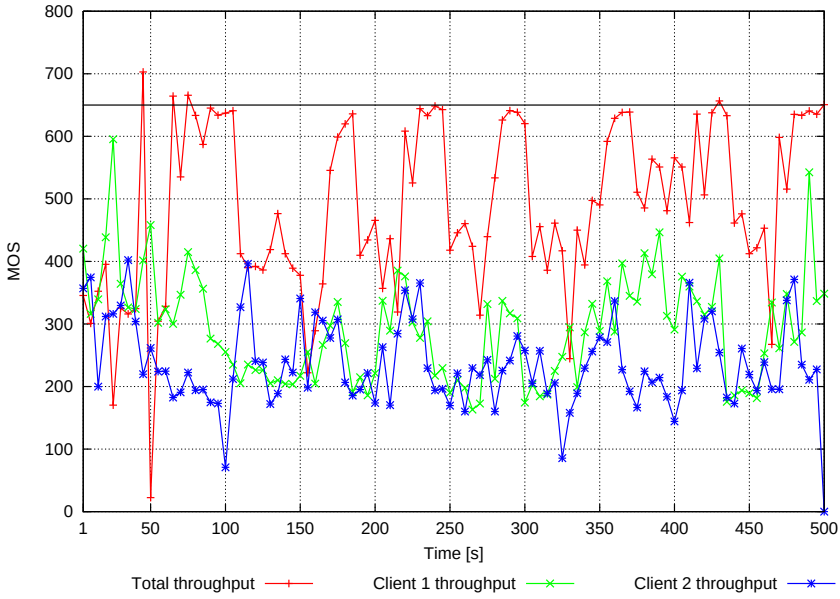

Figure 11.5: Evolution of the throughput

Fig. 11.6 shows the evolution of the RTT deviation and the MOS score as reported by the 2 clients. It can be noticed that the image quality slightly dropped when congestion was induced by up-switching to the highest video quality by one of the clients. The spikes in the RTT deviation show however that there were frequent switches in the video quality, for both clients. This type of behaviour is not desired and should be optimised in the future.

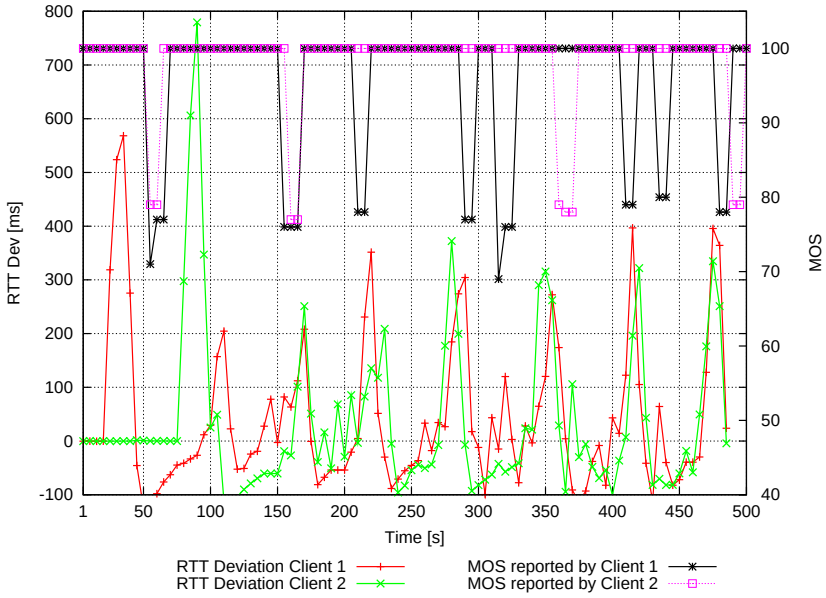

Figure 11.6: Evolution of the MOS score and the RTT deviation for the 2 competing clients

Chapter 12

Validation in Wireless Networks

Because an adaptive streaming solution is intended to be used with clients connected in different wireless environments, three test scenarios were prepared to validate the proposed algorithm:

1. Client connected in a private WiMAX network operated through an Airspan MicroMAX access point.
2. Client connected to a WiFi router.
3. Client connected in a mini test LTE network.

However, we were not fully in control of these experimental set-ups. This has lead to small inconsistencies from one experiment to the other.

To demonstrate the applicability of this algorithm, a case study has been envisioned where the proposed solution is integrated in an interactive streaming platform and tuned to improve both the user experience and interactivity delay.

12.1 Case study: interactive streaming

The latest advances in codec and network technologies led to the development of new streaming services like interactive or multi-view streaming. In interactive streaming the viewer can perform different actions during playback, like selecting different ROIs, view angles, or different zoomed-in and slow-motion scenes, some practical applications being investigated in [72] and [73]. In this type of application, the viewer sends different requests and the server streams the new content. Therefore, it is important that the delay between the moment the command has been sent and

the moment when the new content is displayed is kept to a minimum to increase the user's experience with the service.

12.1.1 Bit-rate variation and content switching

Compared to simple adaptive streaming techniques, where for the same video, the server could stream different quality versions, interactive services require the ability to instantly switch between different quality versions and different contents. For this reason, an existing interactive streaming platform described in [74] has been chosen to integrate the proposed rate adaptation technique. It uses the BSS technique, described in Section 7.1.3, that allows the streaming server to chain multiple video pieces - or clips, as named by the authors in [74] - in a single continuous sequence, so that different streams can be forwarded to the client through a unique and fluent streaming session. As shown in Fig. 12.1, the server can seamlessly switch between different video clips to change the bit-rate or to stream new content requested by the viewer.

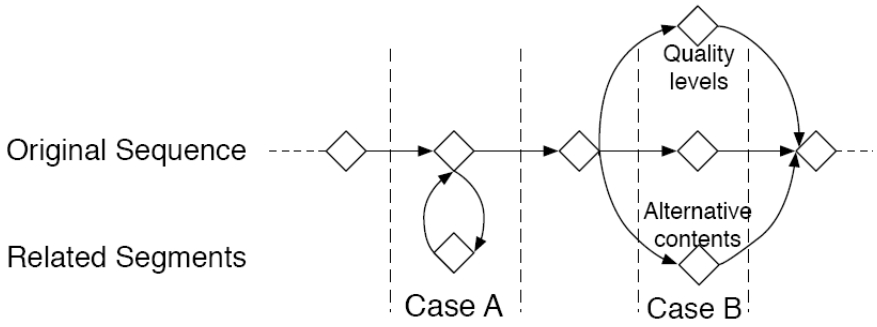

Figure 12.1: Server capabilities to switch between different content and quality encodings

12.1.2 Improving interactivity delay

As already mentioned, in an interactive system it is essential to keep the reaction time as low as possible, the delay between a request at the client side and the consequence of that action in terms of played content should be minimised. Typically, this delay has 3 contributions:

1. The server side delay - depends on how the video stream is split into clips to support interactive services.

2. The end-to-end (E2E) delay, from the server to the client through the network

3. The time required to empty the pre-roll buffer, if there is no remote possibility to flush the video buffer of the player.

The first contribution depends on how the video stream is split into clips to support interactive services. As explained in Section 7.1.3, content can be switched only at key frames so, a large number of clips of shorter duration decreases the coding efficiency, but improves switching flexibility.

The second component is imposed by network characteristics and is independent of the server and the client. Congestion has a significant impact since it increases network latency, but by using the proposed rate adaptation algorithm, network delay will be maintained close to minimum.

The third delay component depends on the client buffer occupancy when an interaction command is launched by the user. Because the buffer is a FIFO type queue, the newly arrived content will have to wait for the old buffered frames to be displayed. So, a small buffer is desired for reactivity, but working with a small buffer makes the problem especially challenging since it increases the risk of starvation. In the absence of a mechanism to flush the client's buffer, the best trade-off would be a dynamic buffer which is filled to a higher level during normal playback and is kept almost empty before content switch. Since the server cannot guess when a user would interact with the player, the best approach would be to maximise the chance that an interaction takes place when the buffer is nearly empty. As explained in Section 10.1 by using the probing design shown in Fig. 10.2, the buffer is almost depleted during the gap. If the viewer sends a command during this period, the server will send the new content in the following burst, significantly reducing the interaction delay. To avoid the exhaustion of the buffer, the server will not probe the network when congestion is detected and a certain guard period after congestion has ended as well. For this reason, the waiting time after congestion has been set to 6 RTCP reports during the experiments.

Hence, by using the bit-rate adaptation algorithm proposed in this thesis in an interactive streaming scenario, there are two elements that are improved to maximize the user's experience:

- The received video quality;

- The reactivity of the streaming system to user interactions.

12.2 Validation in a WiMAX Network

The first series of tests has been performed in a IEEE 802.16(WiMAX) access network. The client was connected through an AirSpan EasyST modem to the Airspan MicroMAX base station. The set-up is presented in Fig. 12.2,

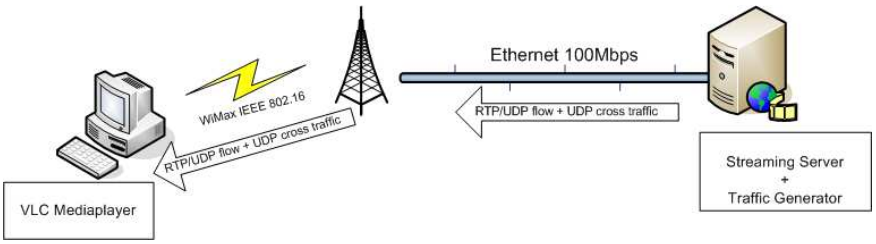

Figure 12.2: WiMAX setup

For this test, three quality versions of the content were available on the server, with the following encoding rates: 2.5 Mbps, 1.7 Mbps and 800 kbps[1] for the lowest quality. Since the capacity of our private WiMAX link is about 6.5Mbps, we simulated a drop in the available bandwidth by sending additional UDP cross traffic over the air interface at a rate of 4.5 Mbps as network stress. The duration of the bandwidth contention is set to about 25s, similar to the throughput variations observed in [75] at driving speeds.

12.2.1 QoS-only based adaptation

In this case only the QoS adaptation layer is enabled, the QoE one not being active. The time evolution of the observed throughputs is shown in Fig. 12.3.

When the cross traffic is sent (from 22s to 47s), the total throughput reaches the maximum capacity of 6.5 Mbps. This is not enough to send the whole 7 Mbps of data (2.5 Mbps video + 4.5 Mbps UDP cross traffic). The server then decides to switch to the next lower encoding quality after approx. 10s and will up-switch back in another 40s, once bandwidth contention is over.

The maximum value of RTT is lower and the congestion period is minimised thanks to the proposed rate adaptation mechanism. In this way the interactivity delay is kept to a minimum during congestion.

[1] In this case the encoding rates do not respect the FDFU rule to highlight the effect of a down-switch.

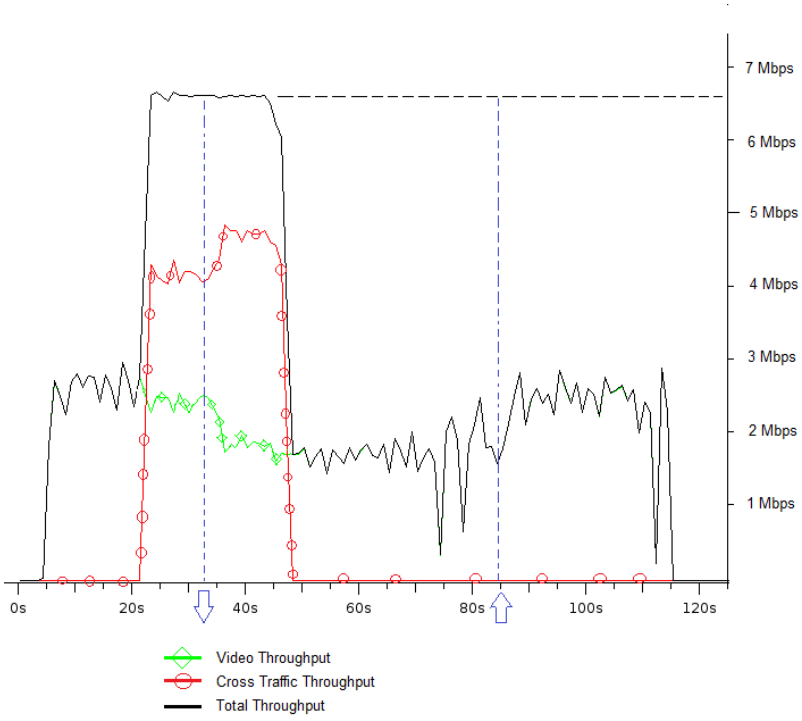

Figure 12.3: Throughput evolution during WiMAX test

Experiment type	Average RTT during cross traffic
With Adaptation	270ms
Without Adaptation	650ms

Table 12.1: Average RTT during congestion

Moreover, packet loss is avoided since the network buffers do not get overflooded. Playback remains smooth, without image artefacts. Table 12.1 shows the average of the RTT during the congestion period as measured from four different runs of the experiment.

12.2.2 QoS and QoE based adaptation

In a second series of experiments, the QoE adaptation layer has been enabled to see if the user experience will be increased during the congestion period compared to when only the QoS layer is used. Several runs have shown however that the QoS adaptation is more efficient in addressing congestion since the RTT deviation threshold is reached before a drop in the MOS score occurs. So in this particular scenario, when only network issues are present, the MOS statistics serve just as a measurement of the

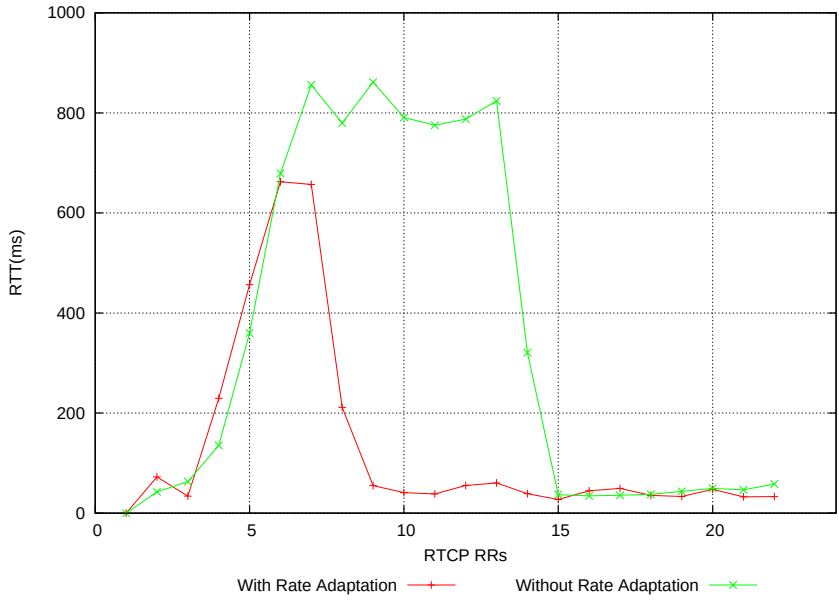

Figure 12.4: RTT evolution in the WiMAX test

user experience. Fig. 12.5 shows that the down-switch is triggered by the increase of the RTT deviation before the image quality is degraded. In fact, in this experiment, the 6th RTCP message reported 1.2% packet loss, which produced a slight distortion in the picture quality.

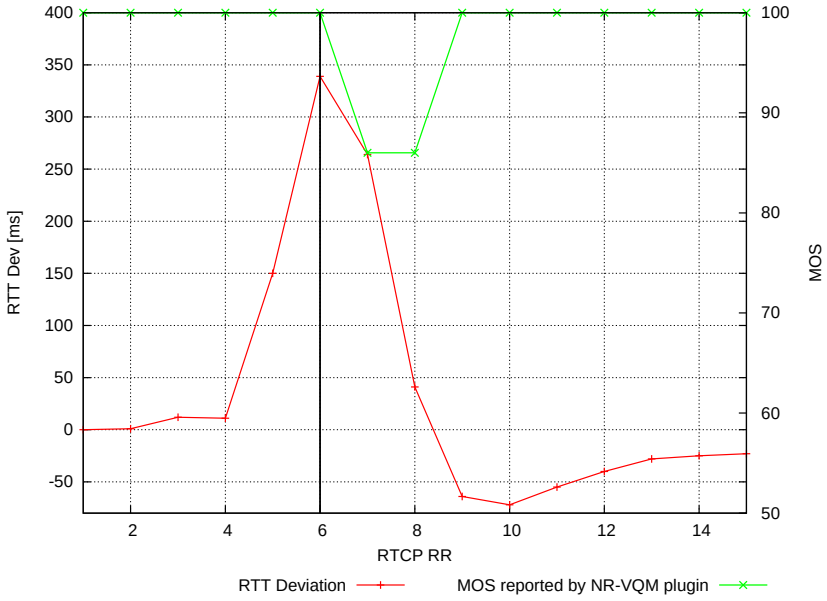

Figure 12.5: RTT deviation and MOS evolution in the WiMAX test

12.3 Validation in a WiFi Network

In the WiFi test, instead of simulating network congestion as network stress, the effects of signal degradation have been studied. The client is connected to a WiFi 802.11g router and starts the streaming session near the access point, then walks away about 20 m from the router losing line-of-sight and then returns to the initial position. Although a multi hop experiment has not been performed, the tests completed in [68] show that delay evolution is similar to single hop paths. During this mobility test, the signal is not lost, but suffers degradation so the available bandwidth decreases with distance and increases back again when the client approaches the WiFi router. The available versions of the content to be streamed were encoded at respectively 350 kbps, 240 kbps and 180 kbps.

12.3.1 QoS-only based adaptation

Again, the QoE layer was disabled, the experiment was ran with and without QoS rate adaptation and results are shown in Fig.12.6 and Fig.12.7.

Compared to the WiMAX tests, packet loss was experienced in both cases, although limited to the switching period to a lower encoding in

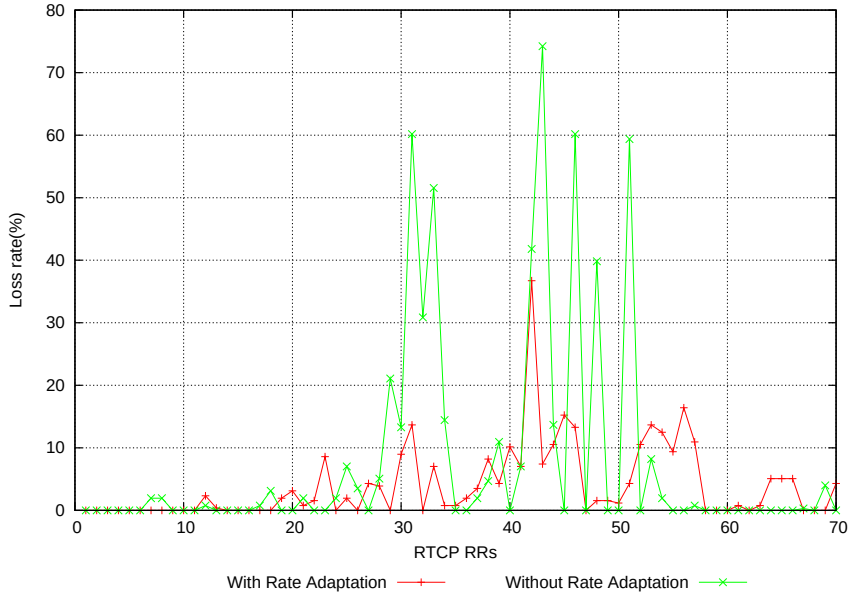

Figure 12.6: Loss rate during WiFi test

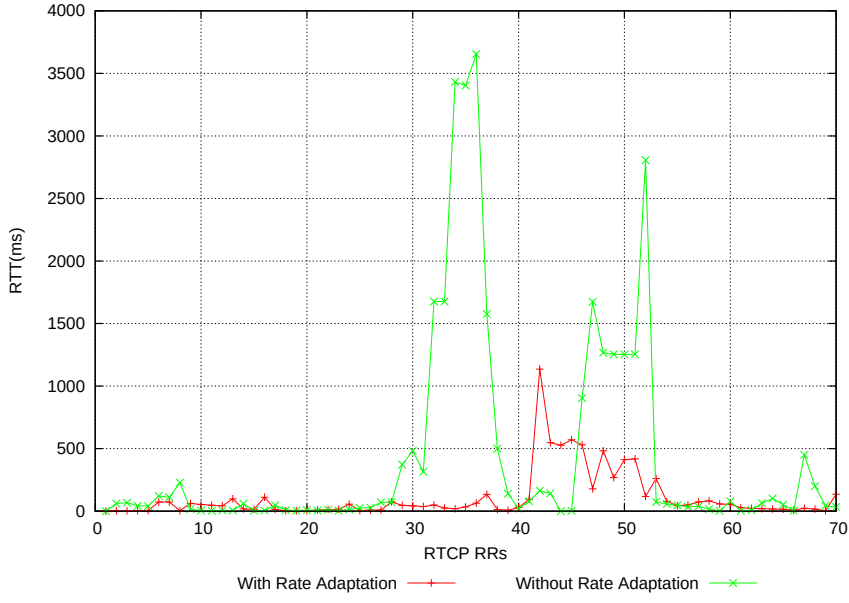

Figure 12.7: RTT evolution during WiFi test

Experiment type	Average packet loss
With Adaptation	3.6%
Without Adaptation	8.2%

Table 12.2: Average loss rate for the whole streaming session

the case with rate adaptation. Table 12.2 shows the average packet loss during the whole streaming session obtained from five different runs of the same experiment. Compared to the earlier QDisc and WiMAX experiments, packet loss was more severe in the WiFi environment.

12.3.2 QoS and QoE based adaptation

In the same conditions, a second series of tests have been performed with the QoE layer enabled. Because the nature of the perturbations were network specific only, the QoS thresholds were reached before, or at the same time as the QoE ones. This happens because the MOS score drops when image degradation is present, while the RTT deviation threshold is set up so it prevents image degradation by triggering the down-switch earlier. In this case the QoE adaptation layer serves as a back-up for the QoS layer.

Fig. 12.8 shows the evolution of RTT deviation and the MOS score during a streaming session. Marked with the first vertical line is the moment when the QoS thresholds have been reached (which triggered a down-switch), while the second vertical line marks the drop in the MOS score, which would have triggered the down-switch. The delay between the two moments is 2 RTCP RR, equivalent to approx. 10s.

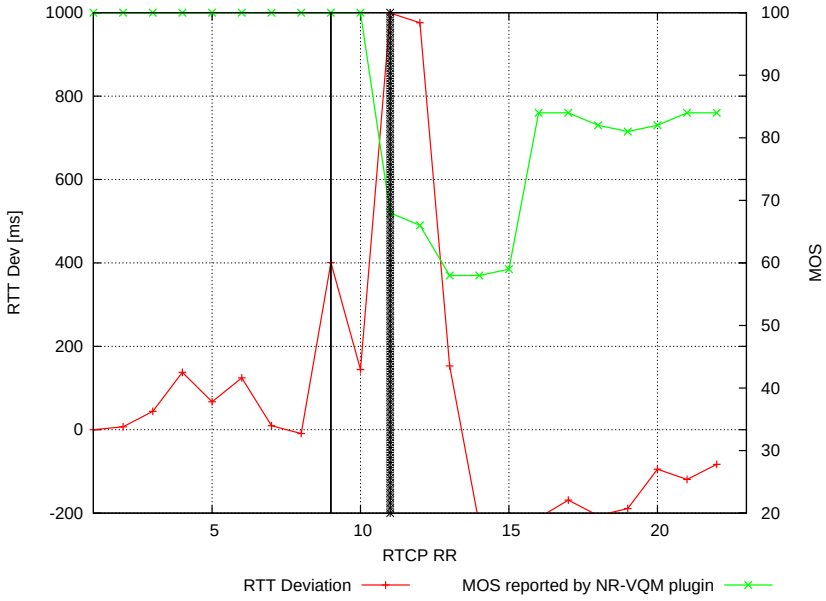

Figure 12.8: RTT deviation and MOS score evolution during a WiFi streaming test

12.4 Validation in a mini LTE network

LTE networks are just being deployed nowadays, either for the public or as small scaled, fully functional testbeds. The experiments in this section were made in the autumn of 2010 in a test LTE mini-network, hosted by a local operator. A part of the material presented in this section has been published in [68] together with Laurent Schumacher and Gille Gomand.

12.4.1 Test Environment

The test environment, presented in Fig. 12.9, was made of three outdoor, urban cells and one indoor femto-cell. The equipment was compliant with LTE Rel'8 May 2008 interim release. Its set-up was basic: default bearer, 10-MHz bandwidth in 2.6 GHz frequency band.

The User Equipment (UE) on which our experiments have been performed was a laptop running Windows XP. A prototype modem was connected to the laptop through USB. A proprietary software enabling to monitor the performance of the Radio Access Network (RAN) was also running on the laptop, enabling to log those parameters for post-processing. The streaming server was hosted on a different site, 12 hops

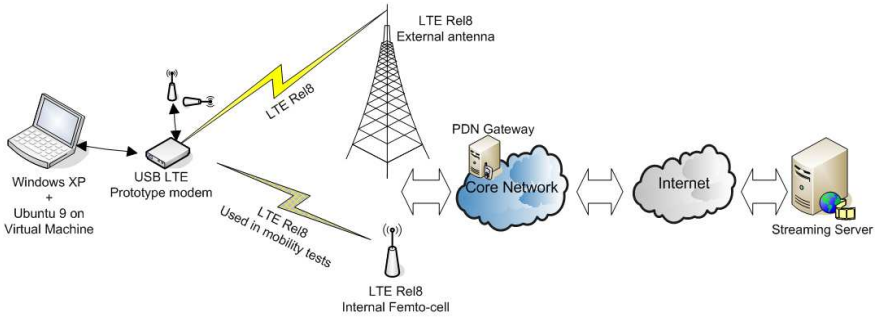

Figure 12.9: LTE experiments set-up

away, so an internet connection was used to access it.

12.4.2 Experiments

The scope of the experiments was to measure the QoS and QoE parameters of the received video in the LTE environment. The streaming experiments were performed both in a static and in a mobile scenario. In the latter case, the UE was carried around at walking speed, in order to trigger hand-overs between an outdoor cell and the indoor femto-cell. The goal was to observe how the playback would be affected by signal degradation and congestion. For this reason, the E2E RTT was recorded, as computed by the streaming server based on RTCP messages. The QoE on the client side was measured using the NR-VQM prototype implemented as a GStreamer plug-in, as explained in Section 3.2.3.

12.4.3 Static scenario

Several streaming sessions were run over both UDP and TCP. We were forced to use TCP for the sake of the NR-VQM measurements. The NR-VQM plug-in was running in a virtual machine on the Windows XP laptop, and UDP port mapping conflicted with the NAT configuration on the virtual machine software. UDP sessions were performed directly from the host operating system, without the virtual machine being involved.

While streaming, congestion (network stress) was simulated by running a speed test on `www.speedtest.net` on the same machine. This event triggered an increase in the RTT and determined a decrease in the MOS value computed by the NR-VQM algorithm on the client side, where the viewer could observe some jerkiness in the playback. The duration of the congestion period was 10s, as can be seen from the throughput graph in Fig. 12.10. The MOS is also shown on this graph. The sharp

drop of the score illustrates the impact of the congestion period onto the user experience. The corresponding RTT evolution is displayed in Fig. 12.11. When the player runs on the virtual machine (TCP case), the E2E delay is much higher, 1.5s in average, compared to the UDP case which exhibits the typical LTE RTT (35-40 ms). Also, the reported RTT variation is very high, frequently exceeding 1s from one RTCP report to another. Considering the thresholds chosen to trigger a down-switch, in this particular case of streaming content over TCP to a client on a virtual machine, the adaptive streaming algorithm would not perform correctly.

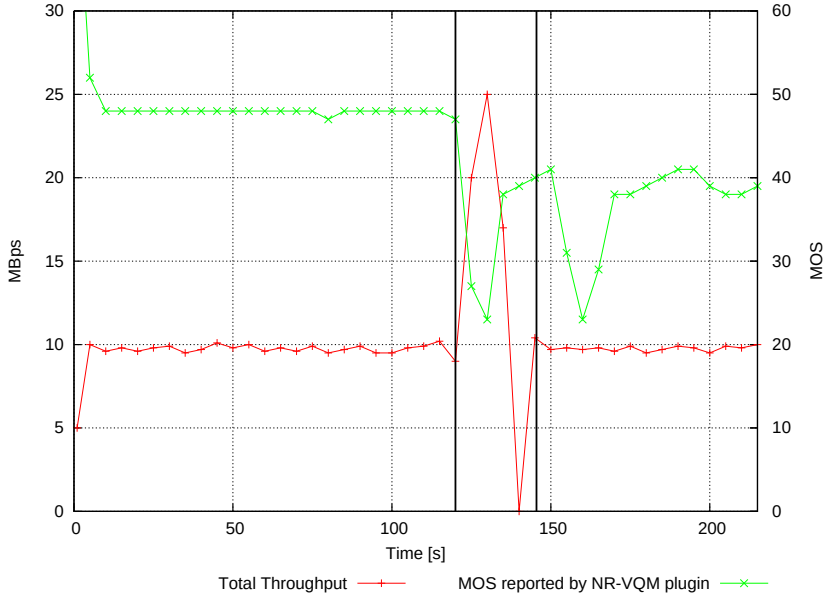

Figure 12.10: Overall throughput and MOS evolution (static, UDP)

In [76] the authors compare the delay increase induced by different Linux-based virtual environments. This increase could go up to 100 ms in worst-case scenarios, when heavy contending TCP traffic was present. Since we used a more general, commercial, virtualisation solution which was not optimised for our specific set-up, we believe RTTs higher than the 100 ms observed in [76] are possible, and we therefore blame the huge E2E delay of the TCP case on the use of virtualisation.

Due to this behaviour, the maximum MOS reported by the NR-VQM plugin is 49, much lower than the usual value, which should be equal to 100 when the playback is smooth. If the adaptation algorithm had been used, the low MOS score would have triggered a down-switch,

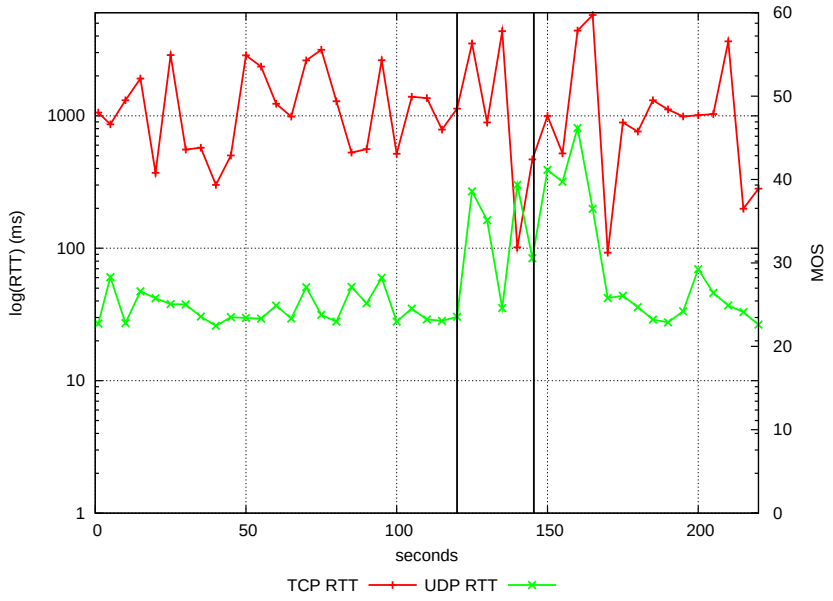

Figure 12.11: RTT evolution during static streaming session

Reported MOS	Subjective Scale	Video Quality	Impairment Perception
49	5	Excellent	Imperceptible
39	4	Good	Perceptible but not annoying
29	3	Fair	Slightly Annoying
19	2	Poor	Annoying
10	1	Bad	Very Annoying

Table 12.3: Corresponding video quality and impairment perception for the NR-VQM computed MOS score in the LTE tests

although this would not have been necessary. Even if this affected the results, we can consider the nominal MOS=49 as a reference for excellent video quality in this particular case, since the playback was not affected by the virtual machine. However, during the congestion period the MOS dropped quite significantly. Considering the mapping defined in Table 9.3, the same correlation can be made with this particular MOS reference, as shown in Table 12.3. In Fig. 12.10 it can be seen that MOS score drops to a value between 20 and 30, indicating "Fair" to "Good" video quality.

12.4.4 Mobile scenario

The experiments from the static scenario have been repeated while the UE was moving between the 2 LTE cells. Besides the RTT and the MOS, different cell parameters (e.g. `CellID`) were logged in order to determine when a handover was triggered. Logs were collected from both UDP and TCP sessions.

On top of the streaming session, an additional FTP transfer of a large file was initiated in order to increase the total throughput up to the network's capacity. This would simulate a worst case scenario, when a user would be watching a video encoded at a rate close to her/his maximum achievable throughput. In this case, even a small decrease in signal quality could affect the playback.

In Fig. 12.12 the three handovers triggered during the UDP streaming session are marked with vertical lines. Their effect can be observed on the RTT graph. However, the first spike on the RTT graph was produced long after the first handover occurred, when the signal quality was very good again. It is likely to rather be a network issue on the way to the server than an access issue in the LTE network. Despite these handovers, no packet loss occurred and the playback quality was perfect. It can be seen that the RTT deviation is far below the 100ms limit which would trigger a down-switch.

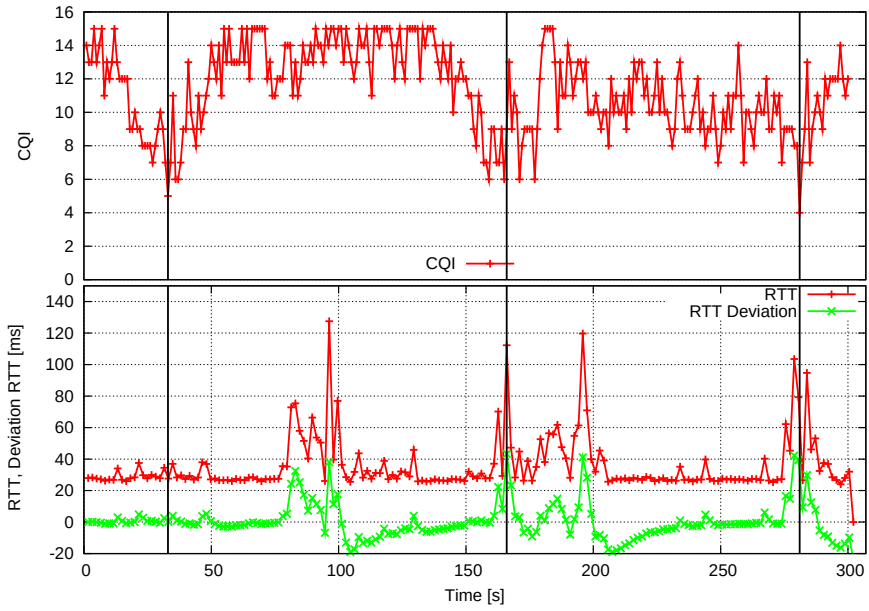

Figure 12.12: Wideband CQI and RTT evolution (mobile, UDP)

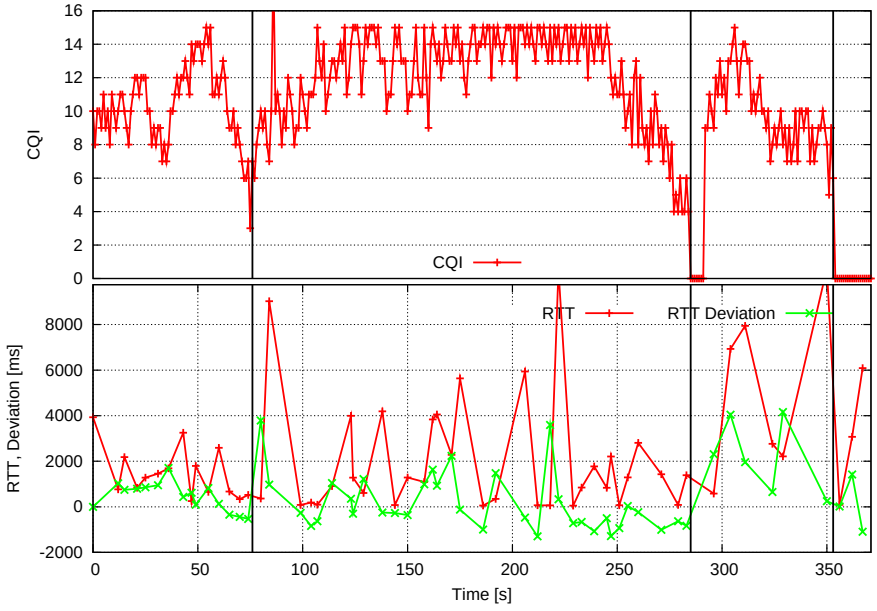

Figure 12.13: Wideband CQI and RTT evolution (mobile, TCP)

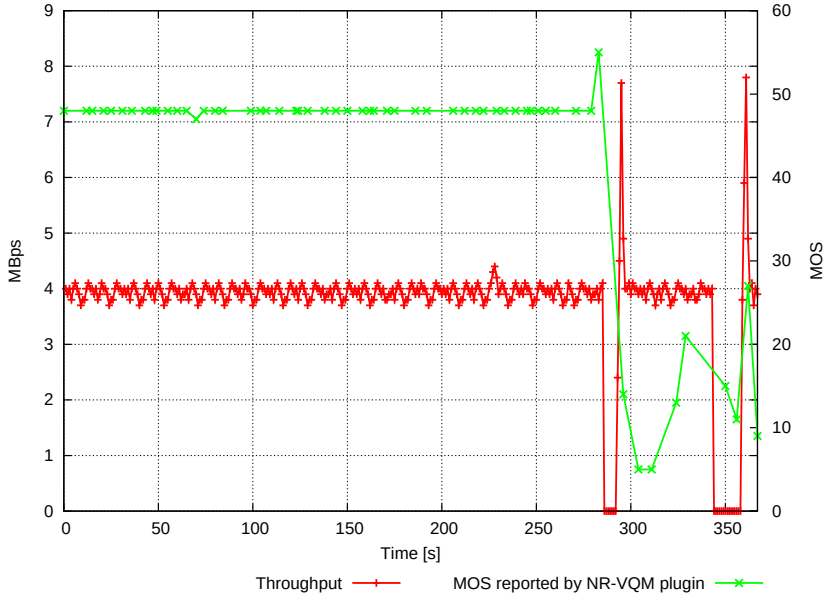

Figure 12.14: Overall throughput and MOS evolution (mobile, TCP)

We repeated the experiment, this time using a TCP connection with the virtual machine and logging the MOS computed by the NR-VQM

plug-in. Fig. 12.13 plots the CQI evolution during this test, with the corresponding RTT and RTT deviation. Again, the use of the virtual machine generated very high RTTs along with very high values for the deviation. A first handover took place after approximately 75s (marked on the graph with a vertical line). Close to the end of the session we can observe two sections where the CQI is missing. That happened because the UE passed through a "blind spot" where the connection was lost and the UE entered the cell detection mode. The first handover produced just an increase in the RTT, but the signal loss forced the playback to stop, causing very high RTTs and a decrease of the MOS. When the connection re-established, the playback resumed but since the UE was in a low reception zone, the quality was still low, as seen from the VQM graph in Fig. 12.14. Finally, the signal was lost again and the streaming session ended before the connection was restored.

12.5 Conclusion

In this chapter a "proof of concept" has been presented for the proposed adaptive streaming algorithm. In the particular case of interactive streaming, the probing technique was adjusted to serve two purposes: to determine whether the network supports a higher encoding quality and to reduce interactivity delay. Further on, the experiments showed that the QoS and QoE based rate adaptation algorithm performs correctly in different wireless environments, confirming that the down-switch and the up-switch thresholds were properly determined. The field tests ran in a mini LTE network have shown how network congestion or mobility consequences affect the playback quality when streaming close the maximum achievable throughput. Although the measurements were affected by the use of a virtual machine, the MOS values collected through the NR-VQM plug-in showed an evolution similar to what was obtained in the experiments presented in Section 9.2. Extrapolating, this validates that the QoE switching thresholds were correctly set, but reveals that additional work is needed for improving the functionality of the NR-VQM plug-in.

Part VI

Conclusion and Perspectives

Chapter 13

Achievements

The present dissertation aimed at designing, building and validating an adaptive streaming algorithm that uses network parameters and video quality measurements at the client side to estimate the experience of the viewer. Based on these evaluations, the streaming server decides which quality version of the video will be sent to the client. As discussed in Chapter 7, several techniques were proposed before, but the solution described in this report exhibits a number of innovative elements that make it unique.

13.1 Layered design

First, it is based on a two tier approach taking into consideration QoS parameters and QoE estimation when paired with a compatible media player. The combination of the two types of metrics increases the speed and the accuracy of the adaptation algorithm:

- QoS based adaptation takes into consideration only delivery problems but can detect in advance a possible situation that would lead to image degradation. It assesses whether the network conditions support the current streaming rate or a higher rate, using only the standard RTCP feedback received from the client. Based on the estimated network conditions, the server increases or decreases the video bit-rate.

- QoE based adaptation offers more accurate measurements regarding user experience, since it directly analyses the image that appears on the viewer's display. As shown in Chapter 3, the best way to do it in a streaming scenario is to use a No-Reference objective Video Quality Metric on the client side. This means that not only

transport issues are detected, but any kind of condition that would affect the image quality, like the lack of processing power of mobile devices, buffering issues, etc. The downside of using only this adaptation layer is that it detects problems when the image quality is already affected and it is not well suited to discover if the network allows an increase of video quality.

13.2 Network state estimation

The second important accomplishment consists in the estimation of network state, optimised to work only with the standard implementation of the RTP/RTCP protocol. It combines suggestions made by the authors in [1] along with the adaptation of the formulas used in [17]. Section 2.3 proposed a new way to compute the RTT variation in a streaming session by adapting the formulas used for the calculation of the TCP Retransmission Timeout timer. This method offers a better accuracy compared to the jitter values reported in the RTCP packets, especially in the case of videos encoded with the H.264 codec [77].

To detect an increase in the available bandwidth, an original network probing mechanism has been introduced in Section 10.1. This technique is based on the network's response in terms of RTT deviation to the additional load introduced by an RTP packet burst. It avoids overfilling or emptying the player's buffer by making a pause in the transmission just before or after the burst, so it can be used without a buffer reporting mechanism. Because it uses the RTP packets from the current video stream for probing, this method has a couple of advantages over other approaches used to determine available bandwidth:

- it does not send unnecessary data over the network
- it does not require the deployment of a dedicated client application to analyse the probing traffic
- if traffic classification is applied along the path between server and client, probing packets will be buffered in the same queue with the other RTP packets, giving an accurate estimation of the available bandwidth.

13.3 Algorithm tuning and validation

Following the FDFU approach, the parameters of the algorithm have been tuned to offer a balance between an aggressive and a more relaxed

adaptation strategy. For this reason, Chapters 9 and 10 present several tests that have been made to analyse the RTT and therefore the RTT deviation in different bandwidth limited, streaming scenarios. The obtained results along with measurements made in various wireless environments have led to the conclusion that a threshold of 100ms for the RTT deviation can be used for both down-switch or up-switch scenarios. Different set of tests determined the down-switch threshold for the MOS score, when the QoE adaptation layer is active.

Such algorithm would be useless without any validation. For this reason, **Part V** of this dissertation is entirely dedicated to validation and performance tests in different, real, wireless networks. Since it relies on RTCP messages to receive feedback information, the speed and accuracy of the algorithm depend on the frequency of those reports. In the worst case scenario, when the reporting interval is about 5 seconds long, the performances are similar to the ones achieved by some commercial HAS solutions, like *Microsoft Smooth Streaming* player or *Netflix* media player.

For validation purposes, three test scenarios were prepared, each in a different wireless access network: WiMAX, WiFi and LTE. To show the flexibility of the whole concept, the algorithm has been tuned specifically for an interactive streaming application, where it can improve both the user experience and interactivity delay. The experiments proved that the adaptation algorithm performs correctly in different wireless environments, confirming that the down-switch and the up-switch thresholds were properly determined. However, they have also pointed out that additional work is needed to refine the functionality of the NR-VQM plug-in.

Chapter 14

Perspectives and evolution

14.1 Perspectives

Due to its advantages in terms of ease of deployment and reaching users, HTTP along with HAS has been the technique chosen by the majority of service providers to deliver their media over the internet. Nevertheless, as stated in the beginning of Chapter 8, there is still room for RTP adaptive streaming. TCP is not an efficient protocol for transporting real time media in environments characterised by fast bandwidth variations and random packet loss, as is the case for wireless scenarios. Therefore, when used on top of UDP, RTP adaptive streaming can successfully be deployed in a mobile network, for VoD or television services. The case study presented in Section 12.1 and described in more detail in [78] shows a possible application where the adaptive algorithm can be integrated. Due to the small amount of data used to probe the network, the proposed algorithm can be deployed in a live streaming scenario by buffering a small portion of the video stream before sending it to the viewers.

Another issue concerns the reporting mechanisms used for sending the feedback from the player to the media server. As **Part III** has shown, lots of extensions exist but only the standard RTCP messages are implemented in the commercial media players. The RTCP-XR QoE metric report block is the only extension dedicated to MOS reporting but it did not become a standard, so only prototype implementations exist. This also limits at the moment a large scale usage of the QoE adaptation layer.

14.2 Future development

Further updates should focus on improving the performance of the algorithm: accuracy of the probing technique, functionality of the NR-VQM plug-in in different scenarios. The easiest method to improve performance is to increase the frequency of the RTCP reports. RFC3556 [79] defines an SDP extension that can be used by a streaming server to specify the bandwidth allowed for RTCP packets. Although the standard was published in 2003, most of the media players today do not support this extension. The next step would be to optimize the down-switch and up-switch thresholds for the behaviour of a specific network. For example, different limits could be set for different time periods: during working hours a more aggressive strategy should be envisioned when the probability of congestion is higher; a more relaxed one for weekend and evenings.

Another element that could be improved is probing accuracy, by using a similar algorithm to the one described in [57]. If implemented directly on the server, it can use video packets as probing data, but since each probing packet sent has to be analysed individually on the client side, it requires an adaptation of the media player as well.

As described above, the outcome of this thesis is a rate-adaptation algorithm for RTP streaming scenarios that can be deployed in production with minor modifications. What makes it attractive is that it can be used with any kind of media player that supports the minimum RTP/RTCP standard, while offering advanced functionality if the media player is capable of reporting an MOS score.

Part VII

Appendices

Appendix A

Peer Reviewed Publications

A.1 Offering Streaming Rate Adaptation to Common Media Players

Presented at: IEEE International Conference on Multimedia and Expo (ICME), 2011

OFFERING STREAMING RATE ADAPTATION TO COMMON MEDIA PLAYERS

George Toma, Laurent Schumacher* and Christophe De Vleeschouwer†*

*Faculté d'Informatique, FUNDP, Namur, Belgium
Email: {gto,lsc}@info.fundp.ac.be
†ICTEAM, UCL, Louvain-la-Neuve, Belgium
Email: christophe.devleeschouwer@uclouvain.be

ABSTRACT

This paper describes an adaptive streaming technique that exploits temporal concatenation of H.264/AVC video bitstreams and uses only standard RTCP reports as feedback mechanism. As a result, common media players like VLC, QuickTime or GStreamer based, can be used in a streaming session with improved viewing experience when accessing video content through bandwidth constrained connections. Further, an original probing technique that uses video packets as probing data has been developed in order to assess whether the available bandwidth allows streaming at a higher bitrate, maximizing thus video quality and user experience. The proposed solution has been tested in real wireless scenarios, showing that video quality can indeed be improved even for standard media players.

***Index Terms*—** Adaptive algorithm, streaming media, media players, mobile/wireless communication

1. INTRODUCTION

Thanks to the development of new telecommunication technologies (UMTS, HSDPA, WiMAX, LTE)[1] allowing high bandwidth connections, television-like services, be it streaming, IPTV or VoD, started to gain a significant market share in the mobile world. However, due to the lossy nature of wireless links (slow and fast fading, mobility issues, handovers), the bandwidth available for certain users can drastically decrease in a short amount of time, affecting the experience of the user with the service. Also, during peak-time hours, the telecommunication cells tend to get congested and therefore can hardly offer constant high quality services. In such cases, the user will experience image degradation, jerkiness or video re-buffering. This means that in the case of bandwidth constrained connections the playback should remain smooth without re-buffering or jerkiness and when the network allows, the viewer should receive the best achievable image quality. This translates into the ability to select the appropriate encoding rate of the chosen content, based on the available throughput.

To assess these problems, one of the solutions proposed in the literature and then adopted in the industry was to automatically adapt the video quality, based on the available network bandwidth. From our point of view, current adaptation techniques proposed in the literature, either rely on TCP's built-in flow control [1] or need RTCP extensions (typically 3GPP Rel 6 compliant) that limits the use of the streaming framework to specific media players [2] or [3]. Another approach is to use tools that compute available bandwidth by explicitly probing the network, sending packet pairs or packet trains to destination. Such a tool is Wbest [4]. It is used by the authors of [5]. The drawback of such probing tools is that they also need to deploy a dedicated client application which analyses the probing packets. Moreover, they have a good accuracy only in certain conditions.

The goal of this paper is to design, implement and test a stream adaptation technique of Baseline H.264 encoded videos, which offers rate adaptation to common media players that only implement the standard specifications of the RTP/RTCP protocol suite[2].

The solution proposed in this paper is based on the streaming platform proposed in [6, 7] and implemented in the open source Live555 Media Server. Its main feature is temporal content pipelining. Baseline H.264 encoded video is split into small segments and each segment is streamed one after another without a new RTSP negotiation, enabling seamless content switching for the client. This offers a great flexibility since the next video segment can be part of a different video scene or can have a different encoding more suitable to the client device and to current network conditions.

2. AUTOMATIC BITRATE SELECTION ALGORITHM

The main idea behind the proposed solution is to assess whether the network conditions support the current streaming rate or a higher rate, using only the standard RTCP feedback received from the client. Based on the estimated network conditions, the server would switch between videos encoded at different quality levels. Although this approach is similar to earlier proposals, its main advantage is that it does not need information about the buffer state, as the network conditions are assessed based on the RTT deviation. This is why it does

[1]Broadcasting technologies like DVB were not considered in this paper, the focus being on cellular and wireless access-networks.

[2]This paper therefore focuses on the traditional way of streaming content, and does not consider the HTTP streaming approach currently in debate at IETF and quickly gaining support and adoption.

978-1-61284-350-6/11/$26.00 ©2011 IEEE

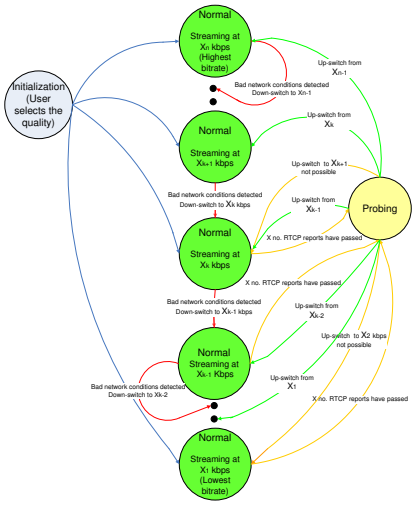

Fig. 1. Adaptation algorithm

not need additional RTCP extensions on the player side. The adaptation algorithm is presented in Fig. 1 as a *Finite State Machine* (FSM) with several *Normal* states and two additional phases: *Initialisation* and *Probing*.

2.1. Initialisation

This is the first phase of the algorithm, it includes the RTSP negotiation and network discovery. The user would have the choice of selecting the appropriate video quality suited for his/her network capabilities. Even if the initial client's selection is not optimal, our proposed adaptation scheme will help the server to eventually adjust and send the video encoded at a rate that best matches the available network bandwidth.

To be on the safe side, the server could start streaming at the lowest quality rate, then increase it until a first packet loss occurs. This sounds like Slow Start in TCP. This scheme is however not implemented in our set-up.

2.2. Normal X_k

In this state the server sends the media at a constant rate of X_k kbps, where $k \in \{1..N\}$ and N is the number of supported bitrate versions. The server also monitors the network conditions by analysing the RTCP reports. If the network conditions impose it, the server will immediately start to stream a lower encoded version of the current content, going in Normal $X_k - 1$ state (down-switch). If the network is stable the server will go to the *Probing* state where it tries to determine if the available bandwidth is high enough to sustain the streaming of a higher quality encoding of the content. If the *Probing* state indicates that it is possible to stream a higher encoding, the server will move to $X_k + 1$ state (up-switch) , streaming the next available quality level.

According to [8], a *Fast bitrate Decrease Slow bitrate restore Up* (FDSU) approach is the most suitable way to assess network congestion. Using this method, the server switches to a clip encoded at approx. 60% of the current encoding. Although high oscillations in bit rate are not recommended because the decrease in quality is easily observed by the user, this approach limits the number of future packet losses due to congestion, which would have greater impact on video quality. This technique implies that false positives should be kept to a minimum, otherwise the user experience can be heavily affected. However, a slow bitrate increase implies producing many quality versions of the same content, which puts a burden on the post processing and storage of several versions. Consequently we will rather use a *Fast bitrate Decrease Fast bitrate restore Up* (FDFU) approach.

Typically, we would recommend to switch between three *Normal* states. For example in a cellular environment, each *Normal* state would be defined to encompass the average rate of a cellular generation, e.g. EDGE (140 kbps), UMTS (200 kbps) and HSPA (350 kbps), as measured in [9].

2.3. Probing

This is an auxiliary state, since the video content keeps on being streamed to the client at reference rate X_k kbps. However, in this case silent gaps and bursts of RTP packets alternate in order to estimate the available network bandwidth. The main idea behind this technique is to temporarily send the video frames at a higher rate (burst) to put the network under stress. If the bandwidth limit is close to the current bitrate, the packets sent in advance will queue in the network buffers and the RTCP reports will feed-back high RTT values at the server. Consequently, from those RTT values, the streaming server can assess whether the available network bandwidth is high enough to switch to a higher bitrate. If this is the case, then the server should switch from X_k kbps to X_{k+1} kbps. Otherwise it will resume regular streaming at X_k kbps.

The advantage of this technique against tools that compute the available network bandwidth by sending packet trains or packet chirps (for instance abing [10], pathchirp [11] or Wbest [4]) is that it does not send extra data over the network, since the RTP packets would have been sent anyway. In addition, it does not require the deployment of a dedicated client application to analyse the probing traffic.

Its drawback however is that the amount of data which can be sent in advance is limited due to the risk of client buffer overflow. To overcome this issue, the burst of RTP packets has to be followed or preceded by a pause in the transmission, the so-called gap. This allows the data from the buffer to be consumed, or to refill the buffer to its average occupancy respectively.

The respective positions of the burst and the gap depend on the risk one wants to mitigate. If we choose to have the

burst first, followed by the gap, we minimize the risk of starvation at the client. An empty buffer forces the playback to stop during re-buffering, which leads to a degraded user experience. On the other hand, a full buffer adds delay in an interactive streaming scenario. Indeed, in the absence of a mechanism to remotely flush the client buffer, the player will consume the old content received during a burst before switching to the display of the new requested content.

Another weakness of our scheme is the limited reactivity of it, as it proceeds by processing RTCP feed-back. Since it is aimed to be used with common mediaplayers, the standard RTCP reporting (one report every 5 s) does not allow very quick response but it still provides a significant increase in playback quality compared to a non adaptive situation. Even when compared to HTTP Adaptive Streaming (HAS) scenarios, the reaction time could be better when high quality video is streamed, since HAS players would delay the TCP feedback to avoid buffer overflow as stated in [12].

3. ESTIMATING NETWORK STATE

As stated in Section 2, the server has the ability to fast switch between different H.264 encoded clips depending on current network state. Next we will describe what elements reported by the feedback protocol are used to estimate the network conditions.

Since all media players that support RTP streaming also support RTCP feed-back, the *Receiver Reports* (RR) and the *Sender Reports* (SR) can be used to compute the RTT, jitter, packet loss and average throughput. To estimate the state of the network, the RTT[3] and packet loss ratio will be used.

3.1. Round Trip Time

The RTT is computed by the server using the method presented in [13]. Although some links are very asymmetric in terms of RTT, the variation of the delay will be reflected in the final value. This is important since a fast increase in the RTT suggests that congestion is about to take place in the network. Because the variation nature of the instantaneous RTT is spiky, two variables will be used to characterize delay evolution: a smoothed RTT and the RTT deviation. The formulas were inspired by the computation of the TCP Retransmission Timer [14], but have been modified so as to get to know whether the deviation increases or decreases, and to increase the speed of convergence to the instant deviation:

$$\begin{aligned} &\text{Smoothed}_{RTT} \\ &\quad = \ (1-\alpha) * \text{Smoothed}_{RTT} + \alpha * \text{Instant}_{RTT} \quad (1) \\ &\text{Deviation}_{RTT} \\ &\quad = \ (1-\beta) * \text{Deviation}_{RTT} \\ &\qquad +\beta * (\text{Instant}_{RTT} - \text{Smoothed}_{RTT}) \quad (2) \end{aligned}$$

where α is 0.125 and β is 0.5. The network state will be reflected by the evolution of Deviation_{RTT} over a specific number of consecutive RTCP reports.

[3]Despite being a two-way time measurement, the RTT is regarded as a good estimate of the E2E delay.

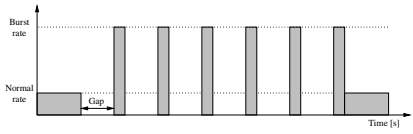

Fig. 2. Probing Cycle

3.2. Packet Loss Ratio

The RR contains two fields related to lost packets: fraction loss and cumulative loss.

The first represents the ratio between the number of lost packets and expected number of packets since the emission of the last RR or SR, while the latter represents the number of lost packets since the beginning of the session. Using the current and previous RRs, the server can compute the total number of lost packets for the reporting interval as the difference of two consecutive cumulative loss reports. By book-keeping and comparing the values for the number of lost packets and the loss ratio between 2 reports with fixed thresholds, the server can decide to down-switch the transmission rate based on the loss characteristics of the connection

4. DESIGN OF THE PROBING SET-UP

When probing the network, we would like to know if the current available bandwidth allows to switch to the next encoding rate, which should be about 60% higher than the current rate, according to our FDFU approach. However, since streaming closely to the bandwidth limit could lead to high RTTs and packet loss ratios, we would aim to up-switch only when the bandwidth limit is almost twice as high as the current streaming rate.

We have discovered that the maximum gap we can make before emptying the buffer with VLC and GStreamer media players is about 1s for robust transmission.

Because we want to prevent buffer under-run or buffer overflow, the length of the gap is strictly related to the length of the burst. We therefore compute it as:

$$\text{Gap}_{Length} = \frac{Burst_{Length}}{FPS} - \frac{Burst_{Length} - 1}{FPS\, Probing_{Factor}} \quad (3)$$

where $Burst_{Length}$ represents the probing duration expressed in number of frames, FPS is the video frame-rate and the $Probing_{Factor}$ represents the frame rate increase.

After testing several values for the $Probing_{Factor}$ and the $Burst_{Length}$, the set-up that provided the most consistent results was obtained with a $Probing_{Factor}$ of 4 and a $Burst_{Length}$ of 32 frames which returned a gap of 970 ms. We could not increase the $Probing_{Factor}$ or $Burst_{Length}$ further because the gap would increase even more and the media player buffer would under-run. To stress even more the network, the probing cycle is repeated six times over a period of 2-3 RTCP reports, as shown in Fig.2

5. ESTABLISHING SWITCHING THRESHOLDS

In this section the rationale for setting the parameters that determine a down-switch or an up-switch, introduced in Section 3, will be exposed.

5.1. Down-switch Thresholds

To determine the events that cause a down-switch, we need to know what is the deviation in different networks under normal conditions and under congestion. To this end, we used measurements in live 3G networks [15, 16, 17] to feed the formulas in (1) and (2). It can be shown that the Deviation$_{RTT}$ only goes above 60-70ms (in absolute value) when the current transmission rate is close to the maximum available bandwidth, but it remains under this value even in the case of a GPRS connection, whereas the jitter is higher than in other mobile networks. Also, the authors of [18] have reported that in 90% of the cases, the jitter was smaller than 100 ms in their measurements. Consequently, if the absolute Deviation$_{RTT}$ value is higher than 100 ms for two consecutive reports, this should be interpreted as a sign of congestion. For the sake of accuracy the number of RTCP RRs taken into account should be higher. However, when the media client has the standard implementation of the RTP/RTCP protocol (like VLC or QuickTime) it sends RTCP reports every 5 s. Waiting for more than two reports would therefore lead to a reaction time greater than 10 s, which is not acceptable.

In some cases, when the congestion level is high, the deviation increases rapidly. To limit the effects produced in this situation, the server will down-switch if the deviation is higher than 300 ms.

Besides the RTT deviation, the server will take into consideration the packet loss ratio as well. Since the wireless environment is a lossy one, it is possible to have a small amount of losses even if the network can sustain the current streaming rate. Based on the experiments we performed (Section 6), the loss rate limit between RTCP reports has been set to 10% but is taken into consideration only if the total number of lost packets is higher than 10. We experienced satisfactory behaviour with these values in a wide range of wireless access networks, from WiFi to LTE.

5.2. Up-switch Thresholds

After each probing cycle, the server would decide whether to up-switch or not, based on Deviation$_{RTT}$ derived from the RTCP RRs.

As explained in Section 4 we aim to switch up only when the available bandwidth is twice as high as the current encoding rate. Several tests have been performed in a QDisc limited Ethernet network to observe the RTT deviation when the probing is done under different bandwidth limits. Six scenarios have been considered where the bandwidth was respectively limited to 2, 1.85, 1.7, 1.5, 1.35 and 1.2 times the current streaming rate and the RTT deviation was collected during each probing cycle. The Cumulative Distribution Function (CDF) is plotted in Fig. 3 with data gathered from approximately one hour of streamed video for each case. Looking at Fig. 3, we can observe the Deviation$_{RTT}$ as a function of the bandwidth margin. For instance, when probing under a limit two times higher than the current rate, the deviation is under 100 ms in 90% of the cases. Extrapolating from downstream UDP throughput from [19], one could possibly model the available bandwidth from UDP streaming as an exponential distribution parametrised to C, the nominal capacity of the wireless set-up. We can plot then the upswitch probability based on throughput distribution in different access networks in Fig. 4. For a deviation $d_0 = 50$ ms, an up-switch has a 90% success rate provided the streaming rate R is lower than 300 kbps in 3G networks, 500 kbps in WiMAX scenario, 1 Mbps in WiFi 802.11b and 5 Mbps in WiFi 802.11g. Considering a sequence at 1 Mbps streamed on a 3G network, the upswitch has a 30% success rate if the observed deviation is up to 200 ms, whereas this rate increases to 40% if the deviation is as low as 50 ms.

For all our tests, we have considered the deviation upswitch threshold of 100 ms because it offers the best trade-off between a low percentage of false positives and a high success up-switch rate. It also matches the down-switch threshold.

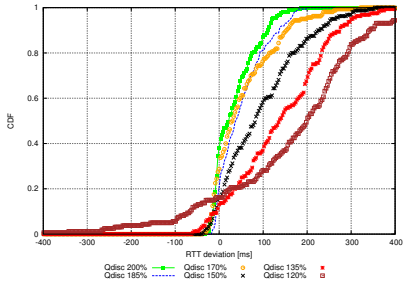

Fig. 3. CDFs for the six scenarios

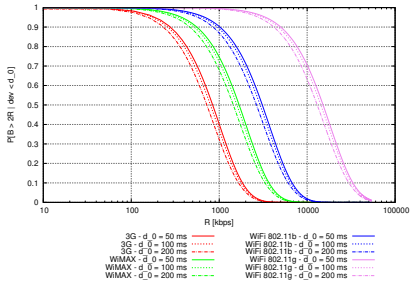

Fig. 4. Probability that there is enough bandwidth to upswitch s.t. observed deviation in different access networks

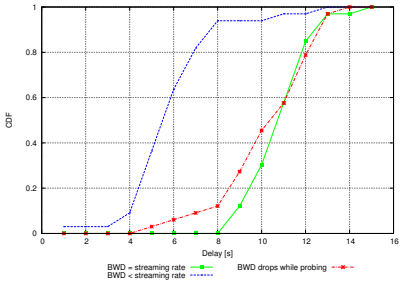

Fig. 5. CDF for the down-switch delay

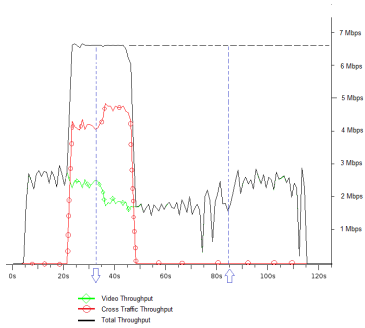

Fig. 6. Throughput evolution during WiMax test

6. TESTS AND RESULTS

6.1. Performance evaluation of the proposed platform

In this section we will present the three tests we performed to show the improvements in Quality of Experience over the same solution without rate adaptation.

6.1.1. Convergence speed of the Rate Adaptation algorithm

In the first experiment we tested the reactivity of the algorithm, with the parameters presented in Section 2. The set-up consists 2 PCs, one hosting the modified version of a LiveMedia555 streaming server and the other hosting VLC media player, connected via 100 Mbps Ethernet interfaces. The available bandwidth between the two can be reduced using the Linux Traffic Control tool, to simulate congestion or signal degradation in wireless networks. During several streaming sessions the available bandwidth was reduced close to, or below the current streaming rate and the reaction time between bandwidth reduction and an actual down-switch was registered. The CDF for the delay is plotted in Fig. 5, where three scenarios have been considered:

1. When the bandwidth drops to a value closer to the current streaming rate, the down-switch delay is in average equal to 10 s, which represents the duration of approximatively two RTCP reporting periods.
2. When the bandwidth drops to a value at least 20% lower than the current streaming rate, the system reacts faster, in about 5-6 s, which represents the duration of about one RTCP reporting period.
3. When the bandwidth drops during the probing phase, the down-switch delay is approximatively 10 s as well, again the span of two RTCP reporting periods.

The reaction time directly depends on the frequency of the RTCP reports, and the results obtained in this paper present a worst case though realistic scenario, where the media player sends RTCP RRs every 5 s.

The up-switch delay depends on the probing cycle duration. In Section 4, it was fixed to six RTCP reporting periods. Hence, if the network bandwidth allows, the server will choose a higher streaming rate after 8 RTCP RRs (probing cycle + probing duration), the equivalent of approximately 40 s.

6.1.2. Tests in a WiMax Access Network

Because the platform is intended to be used with clients wirelessly connected, two test scenarios were prepared:

1. Client connected in a private WiMax network operated through an Airspan MicroMAX access point,
2. Client connected to a WiFi router.

For this test, three different versions of the content were available on the server, with different encoding rates: 2.5 Mbps, 1.7 Mbps and 800 kbps for the lowest quality. Since the capacity of our private WiMax network is about 6.5Mbps, we simulated a drop in the available bandwidth by sending additional UDP cross traffic over the air interface at a rate of 4.5 Mbps. The duration of the bandwidth contention is set to about 25 s, similar to the throughput variations observed in [20] at driving speeds. The time evolution of the observed throughputs is shown in Fig. 6. When the cross traffic is sent (from 22 s to 47 s), the total throughput reaches the maximum capacity of 6.5 Mbps. This is not enough to send the whole 7 Mbps of data (2.5 Mbps video + 4.5 Mbps UDP cross traffic). The server then decides to switch to the next lower encoding quality after approx. 10 s and will up-switch back in another 40s, once bandwidth contention is over.

For this scenario, Fig. 7 compares the time evolution of the RTT, with and without rate adaptation. The maximum value of RTT is lower and also the congestion period is minimised thanks to the proposed rate adaptation mechanism. Moreover, packet loss is avoided since the network buffer does not get overflooded. Playback remains smooth, without image artefacts. Table 1 shows the average of the RTT during the congestion period as measured from four different runs of the experiment.

Experiment type	Average RTT during cross traffic	Std. dev.
With Adaptation	270ms	248ms
Without Adaptation	650ms	254ms

Table 1. Average RTT during congestion

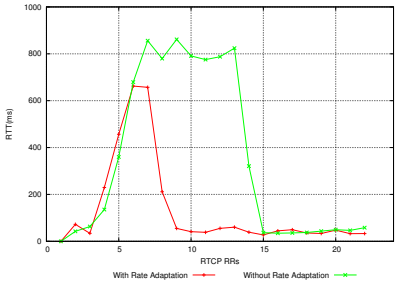

Fig. 7. RTT evolution in the WiMax test

6.1.3. Tests in a WiFi Access Network

In the WiFi test, the client is connected to a WiFi 802.11g router and starts the streaming session near the access point, then walks away about 20 m from the router loosing line-of-sight and then returns to the initial position. During this mobility test, the signal is not lost, but suffers degradation so the available bandwidth decreases with distance and increases back again when the client approaches the WiFi router. The available versions of the content to be streamed were encoded at respectively 380 kbps, 240 kbps and 180 kbps. Again, the experiment was ran with and without rate adaptation and results are shown in Fig. 8 and Fig. 9. The RTT was kept low.

However, packet loss was experienced in both cases, although limited to the switching period to a lower encoding in the case with rate adaptation. Table 2 shows the average

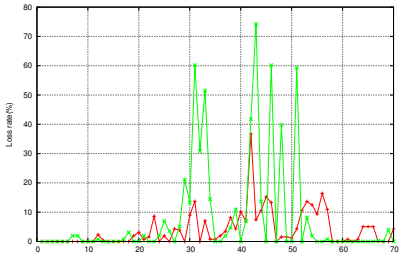

Fig. 8. Loss rate during WiFi test

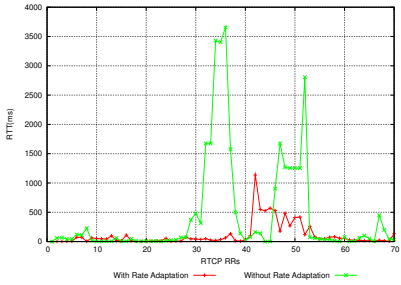

Fig. 9. RTT evolution during WiFi test

Experiment type	Average packet loss	Std. dev.
With Adaptation	3.6%	6.1%
Without Adaptation	8.2%	17%

Table 2. Average loss rate for the whole streaming session

packet loss during the whole streaming session obtained from five different runs of the same experiment. Compared to the earlier QDisc and WiMax experiments, packet loss was more severe in the WiFi environment. Unfortunately, the losses affected the image quality, even when the available bandwidth was higher than the streaming rate.

7. CONCLUSION

We have presented a streaming system that can offer bitrate adaptation to common media players that implement only the standard RTP/RTCP protocol suite. We have shown the performance of the system and its behaviour in two wireless scenarios, underlining the advantage of this solution over a non-adaptive one.

8. REFERENCES

[1] S. C. Liew L. S. Lam, Jack Y. B. Lee and W. Wang, "A Transparent Rate Adaptation Algorithm for Streaming Video over the Internet," in *Proceedings of the 18th International Conference on Advanced Information Networking and Application*, Fukuoka, Japan, 2004.

[2] T. Schierl and T. Wiegand, "H.264/AVC Rate Adaptation for Internet Streaming," in *14th International Packet Video Workshop (PV)*, Irvine, CA, USA, December 2004.

[3] T. Schierl, T.and Wiegand and M. Kampmann, "3GPP Compliant Adaptive Wireless Video Streaming Using H.264/AVC," in *IEEE International Conference on Image Processing, ICIP 2005.*, vol. 3.

[4] Mingzhe Li, Mark Claypool, and Robert Kinicki, "Wbest: a Bandwidth Estimation Tool for IEEE 802.11

Wireless Networks," in *Proceedings of 33rd IEEE Conference on Local Computer Networks (LCN)*, pp. 374–381.

[5] C. Mairal and M. Agueh, "Smooth and Scalable Wireless JPEG 2000 Images and Video Streaming with Dynamic Bandwidth Estimation," in *Advances in Multimedia (MMEDIA)*, pp. 174–179.

[6] Etienne Bömcke and Christophe De Vleeschouwer, "An Interactive Video Streaming Architecture for H.264/AVC Compliant Players," in *Proceedings of the 2009 IEEE International Conference on Multimedia and Expo (ICME'09)*, pp. 1554–1555.

[7] I.A. Fernandez, F. Chen, F. Lavigne, X. Desurmont, and C. DeVleeschouwer, "Browsing Sport Content Through an Interactive H.264 Streaming Session," in *Advances in Multimedia (MMEDIA)*, pp. 155–161.

[8] A.N.M. Zaheduzzaman Sarker, "A Study over Adaptive Real Time Video over LTE," Ph.D. thesis, 2007, Luleå University of Technology.

[9] W. Eklof, "Adapting Video Quality to Radio Links with Different Characteristics," M.Sc. Thesis, Sweden, 2008.

[10] Jiri Navratil and R. Les Cottrell, "ABwE: A Practical Approach to Available Bandwidth Estimation," in *Passive and Active Measurement (PAM) Workshop 2003 Proceedings, La Jolla*, 2003.

[11] Vinay Ribeiro, Rudolf Riedi, Richard Baraniuk, Jiri Navratil, and Les Cotrell, "pathchirp: Efficient Available Bandwidth Estimation for Network Paths," in *Passive and Active Measurement (PAM) Workshop 2003 Proceedings, La Jolla*, 2003.

[12] Danny De Vleeschauwer and David Robinson, "TCP: From Data to Streaming Video," Alcatel-Lucent TechZine, [online], available at http://www2.alcatel-lucent.com/blogs/techzine/2011/tcp-from-data-to-streaming-video/, last visited: April 18, 2011

[13] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, July 2003, Internet Engineering Task Force.

[14] V. Paxon and M. Allman, "Computing TCPs Retransmission Timer," RFC 2988, November 2000.

[15] Ken Guild Marcos Paredes Farrera, Martin Fleury and Mohammed Ghanbari, "Measurement and Analysis Study of Congestion Detection for Internet Video Streaming," Journal of Communications, vol. 5, no. 2, pp. 169–177, February 2010.

[16] Joachim Fabini, Wolfgang Karner, Lukas Wallentin, and Thomas Baumgartner, "The Illusion of Being Deterministic -Application-Level Considerations on Delay in 3G HSPA Networks," in *Proceedings of the 8th International IFIP-TC 6 Networking Conference (NETWORKING '09)*, Berlin, Heidelberg, 2009, pp. 301–312, Springer-Verlag.

[17] Peter Romirer-Maierhofer, Fabio Ricciato, Alessandro D'Alconzo, Robert Franzan, and Wolfgang Karner, "Network-Wide Measurements of TCP RTT in 3G," in *Proceedings of the First International Workshop on Traffic Monitoring and Analysis (TMA '09)*, Berlin, Heidelberg, 2009, pp. 17–25, Springer-Verlag.

[18] Soohyun Cho Hyung-Keon Ryu Jaewha Lee Yeoungseok Lee Sue Moon Keon Jang, Mongnam Han, "3G and 3.5G Wireless Network Performance Measured from Moving Cars and High-Speed Trains," in *Proceedings of the 1st ACM workshop on Mobile Internet through Cellular Networks (MICNET '09)*, pp. 19–24.

[19] Arun Venkataramani Aruna Balasubramanian, Ratul Mahajan, "Augmenting mobile 3g using wifi: Measurement, system design and implementation," in *MobiSys 2010*, San Francisco, USA, 2010.

[20] Richard Gass and Christophe Diot, "An Experimental Performance Comparison of 3G and Wi-Fi," in *Proceedings of 33rd IEEE Conference on Local Computer Networks (LCN)*, Beijing, China, 2009.

A.2 Performance Evaluation of Indoor Internet Access over a Test LTE Mini-Network

Presented at: 14th International Symposium on Wireless Personal Multimedia Communications (WPMC), 2011

Performance Evaluation of Indoor Internet Access over a Test LTE Mini-Network

Laurent SCHUMACHER, Gille GOMAND, George TOMA
Faculty of Computer Science
FUNDP - The University of Namur, Belgium
Email: {lsc,ggo,gto}@info.fundp.ac.be

***Abstract*—This paper reports the results of experiments performed in a test Long Term Evolution mini-network (Rel'8 May 2008 interim release). The experiments were focused on the indoor performance of some Internet applications like bulk file transfer and video streaming. It appears that the 3^{rd} Generation Partnership Project performance targets w.r.t. Round-Trip Time reduction and throughput have been met, even with suboptimal channel quality. Spatial and polarisation diversities are also able to significantly enhance the user experience.**

***Keywords*-4G Mobile Communication, System Performance**

I. INTRODUCTION

For several years, mobile data traffic has been announced repeatedly to be on the verge of an exponential rise, but these many forecasts have all been shown wrong. With the massive adoption of smartphones, and the many applications to be found on their app stores, the end-user has eventually found an incentive to generate mobile data traffic [1]. Simultaneously, the rapid success of HTTP Live Streaming [2] offers contributes to the steep rise of traffic.

Network operators have been waiting for years to see this rise, upgrading their radio access technology to *Enhanced Data Rates for GSM Evolution* (EDGE), *Universal Mobile Telecommunications System* (UMTS), *High Speed Downlink Packet Access* (HSDPA) and most recently *Long Term Evolution* (LTE).

Partly because LTE networks are only rolling out, partly because the enthusiasm around mobile communications has faded away since the beginning of the century, there are not yet so many entries in the open literature presenting performance analysis of such networks. References [3]–[6] document outdoor performance of LTE Rel'8 networks in urban areas. Rural areas are tackled in [7], with a 850-MHz, home-brewed nomadic testbed.

Recently, we have had the opportunity to perform an indoor measurement campaign on a test LTE mini-network consisting of four cells. This paper reports the results of this performance analysis. In Section II, the main features of the cellular network we tested are presented. The experiments we performed are described in Section III. The results of these experiments are presented in Section IV. Finally, conclusions are drawn in Section V.

II. ENVIRONMENT

The test LTE mini-network we were granted access for a week to was made of three outdoor, urban cells and one indoor femto-cell. The equipment was compliant with LTE Rel'8 May 2008 interim release. Its set-up was basic: default bearer, 10-MHz bandwidth in 2.6 GHz frequency band.

The *User Equipment* (UE) on which our experiments have been performed was a laptop running Windows XP. A prototype modem was connected to the laptop through USB. A proprietary software enabling to monitor the performance of the *Radio Access Network* (RAN) was also running on the laptop, enabling to log those parameters for post-processing.

Two external antennas could be connected to the prototype modem. In some scenarios, to be called "Case A" later on, the external antennas were not plugged to the prototype modem. In the other scenarios, the external antennas were connected to the modem. Their presence boosted the *Receive Signal Strength Indicator* (RSSI) by 20 dB. To assess the impact of spatial and polarisation diversity, their relative positioning could be changed. In "Case B", the antennas were co-located and placed orthogonally. In "Case C", the antennas were orthogonal, separated by a varying distance d. Finally, in "Case D", the antennas were also separated by a varying distance d but set up in parallel. Fig. 1 illustrates the various cases. Moreover, 2x2 *Multiple-Input, Multiple-Output* (MIMO) transmit/receive schemes could be exploited.

The *Radio Link Controller* (RLC) operated in the *Acknowledged Mode* (AM). The reporting mode was the aperiodic wideband *Channel Quality Indicator* (CQI) using the *Physical Uplink Shared Channel* (PUSCH). The CQI reflects the level of noise and interference experienced by the receiver on a particular portion of the channel and used by the *evolved Node B* (eNodeB) as an input to the process for scheduling traffic [8].

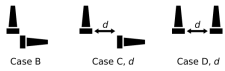

Figure 1. Antenna scenarios

III. EXPERIMENTS

The experiments had been designed following a review of the open literature. Several articles report measurement campaigns of 3G cellular networks. A good survey is available on the *European research portal on Traffic Monitoring and Analysis* (TMA) [9]. We could also add references [10] and [11]. Since the high available throughput of LTE networks can enable a whole range of video services, being VoD or live TV programs, we also ran several RTP streaming sessions as a different set of experiments.

Unlike most of the experiments described in the open litterature, all our experiments were performed indoor. Two of them were static, whereas some nomadicity was introduced in the third experiment, when the laptop was moved back and forth between the coverage areas of an outdoor cell and the femto-cell. Each experiment consisted in several runs performed successively, to strive towards statistical significance while enjoying stationary conditions.

First, we sent *Internet Control Message Protocol* (ICMP) Echo requests (e.g. `ping`) from the LTE network, as in [10]. We targeted two different nodes, first a remote server at our university, then the *Packet Data Network Gateway* (PDN-GW) of the cellular network. According to `traceroute`, the remote server was 12 hops away from the UE. The ICMP packets were exhibiting a growing size. As already mentioned, the UE was static during the whole experiment.

As a second experiment, we generated bulk downlink traffic, from the remote server to the UE. For security reasons related to the firewall policy at our university, we used the *Secure File Transfer Protocol* (SFTP) protocol. Again, the UE was static.

Finally, we streamed content from the remote server at our university to the UE and measured the quality of the received video. The streaming server was a modified version of Live555 media server which logged the data received in RTCP reports. VLC and GStreamer were used as media players. The streaming experiments were performed both in a static and in a mobile scenario. In the latter case, the UE was carried around at walking speed, in order to trigger handovers between an outdoor cell and the indoor femto-cell. The goal was to observe how the playback would be affected by signal degradation and congestion. For this reason, the *End to End* (E2E) *Round Trip Time* (RTT) was recorded, as computed by the streaming server based on RTCP messages. Moreover, the *Quality of Experience* (QoE) on the client side was measured, using a prototype of a *Non Reference Video Quality Metric* (NR-VQM) [12], [13] implemented as a GStreamer plug-in running on a *Virtual Machine* (VM) [14]. The streaming sessions could only be performed without the external antennas ("Case A"). Compared to the Ping and SFTP experiments, additional contending traffic was generated, in order to trigger network congestion while streaming.

Table I summarizes the experimental set-ups. Their lack of consistency is explained by the fact that we had limited control on them, and only a few days to perform the measurement campaign. Consequently, we had to tune our experiments on the fly to observe meaningful phenomena and collect relevant results.

Experiment	Direction	Mobility	Transport	Antenna Case	Contending Traffic
PING	Uplink	Static	ICMP	A,B,D	None
SFTP	Downlink	Static	TCP	A,C	None
RTP	Downlink	Static	TCP (VM)	A	Speedtest
			UDP	A	Speedtest
		Mobile	TCP (VM)	A	FTP
			UDP	A	FTP

Table I
EXPERIMENT SUMMARY

IV. RESULTS

A. ICMP experiments

During the first experiment, ICMP requests were sent uplink, from the UE to a remote server at our university first, then to the PDN-GW. Each `ping` request was sent ten times in a row. The size of the ICMP packet grew continuously following the powers of two, from 1,024 to 65,500 Bytes.

Fig. 2 presents the RTTs reported by ICMP when targetting our remote server, whereas the RTTs observed when putting the PDN-GW under stress are shown in Fig. 3. The average RTT is plotted, as well as the range of the observations. To enable comparison between HSDPA and LTE delays in the access network, the RTTs measured on an HSDPA network operated by Vodafone in Spain and reported in [10] are also plotted on Fig. 3. Additionnally, on both figures, the reported CQI values are plotted, also with their dynamic range. During both experiments, the reported CQIs lied between 10 and 12 on the average. The reader should keep in mind that this experiment was performed indoor and statically.

As shown on Fig. 3, LTE delivers its RTT reduction promise [15]: the RTT to the closest IP node is around 30 ms in LTE vs. 110 ms in HSDPA (ICMP packet size = 1,024 Bytes).

The external antennas are obviously boosting the quality of the link, as shown in Fig. 2. The reported RTT is significantly lower in Case B compared to Case A, despite the fact that the observed CQI was better in Case A than in Case B.

During these ICMP experiments, some packets were lost on the way, triggering a time-out after four seconds. Because the loss rate increased with the packet size, as shown in Fig. 4, and losses only occured when testing the remote server, these losses were likely an unfortunate result of IPv4 fragmentation. Case A was more affected by these losses than Case B.

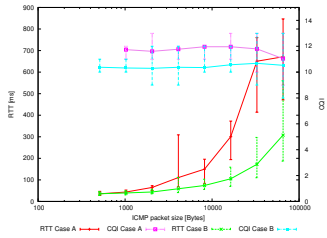

Figure 2. Remote server pinging from UE

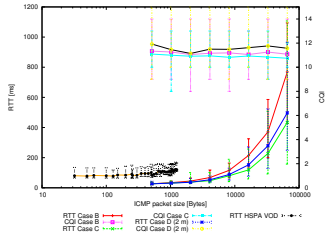

Figure 3. PDN-GW pinging from UE

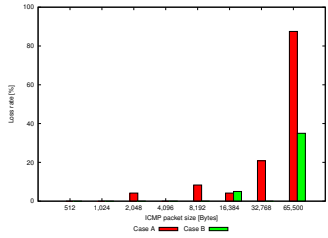

Figure 4. Losses during remote server pinging

These losses should definitely not be disregarded. Indeed, when comparing Case B scenario of Figs 2 and 3, one can notice that the reported RTT is oddly greater for the PDN-GW than for the remote server. This can not be explained by worse channel conditions, as the CQI is slightly higher for the PDN-GW than for the remote server. Actually, the comparison is biased, because it only takes into account the segments that successfully went through. But for Case B, up to 35% of the large segments (65,500-Byte long) suffer from losses. If we force the RTT of these segments to the time-out, i.e. 4 s, and plot the *cumulative distribution function* (cdf) of all transmitted segments, we end up with Fig. 5. It confirms that globally, the RTT of the PDN-GW is indeed smaller than the RTT of the remote server.

Finally, we investigated the impact of spatial diversity (Case D), by varying the distance d between the two external antennas. Fig. 6

shows that this parameter has a significant impact, since one can observe up to a three-fold increase between extreme RTT values.

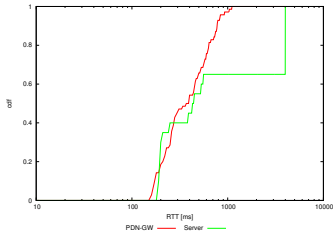

Figure 5. cdf of the RTTs, Case B, ICMP packet size $= 65,500$ Bytes

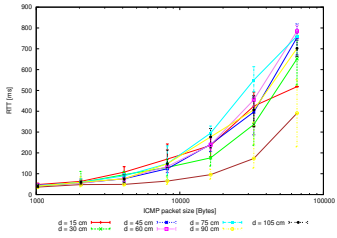

Figure 6. Incidence of spatial diversity on remote server pinging

B. SFTP experiment

The second experiment consisted in the bulk downlink transfer of a large file (157 MB) through SFTP, with and without external antennas (respectively Case C and Case A). It was run seven times for each case, with the seven runs spread over two days. The location of the external antennas was different from one day to the other, leading us to post-process separately the traces collected on these two days.

On average, the transfer was completed in 86.6 s, hence achieving a goodput of 14.5 Mbps. Most of the TCP segments were 1,396-Byte long and their RTT was around 30 ms, as seen on Fig. 2. This corresponds to a *Bandwidth-Delay Product* (BDP) of 300 kbits. The reader should notice that the transfer was the only activity at the time in the network.

When detailing the measured goodputs in Table II, it appears that the use of external antennas was detrimental on Day 2.

Looking at the loss rates experienced during these experiments, listed in Table III, the reader can assess that these bulk transfers were affected by very few segment losses. This will be illustrated even more clearly in Figs. 7 and 8. They show the progress of the sequence number of the TCP segments and the CQI vs. time.

The linear increase on Fig. 7 reveals a stable connection. Indeed, a detailed analysis of the traffic revealed that only 9 TCP segments out of 113,724 have been regarded as lost. Thanks to the sustained traffic of the connection, these losses were quickly assessed by the sender through duplicate ACKs (up to 89 dupACKs in a row). It triggered Fast Retransmit [16] and prevented a significant decrease of the throughput. Actually, the TCP flow was regulated by a `Receive Window` of 35,900 Bytes, smaller than the BDP.

The connection was much less stable on Fig. 8, with a very changing CQI, even falling down to 7. The connection traces logged three times more segment losses, i.e. 27, among which only 8 triggered Fast Retransmit. During this experiment, the `Receive Window` was announced as large as 143,600 Bytes, much higher than the BDP. As a result, congestion control regulated the TCP flow. This is clear on the figure, where the reader can see that the bitrate varied during the connection, with almost no traffic at all for the first twenty seconds.

	Case A	Case C	Set average
Day 1	11.5	16.7	13.6
Day 2	18.7	16.0	17.2
Time average	12.9	16.5	14.5

Table II
AVERAGE DOWNLINK GOODPUTS (IN MBPS)

	Case A	Case C	Set average
Day 1	0.014	0.008	0.011
Day 2	0.011	0.014	0.012
Time average	0.013	0.010	0.011

Table III
AVERAGE LOSS RATES IN THE DOWNLINK (IN %)

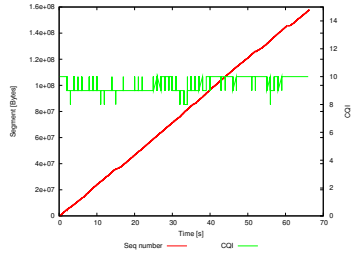

Figure 7. TCP time/sequence graph (`Receive Window` $<$ BDP)

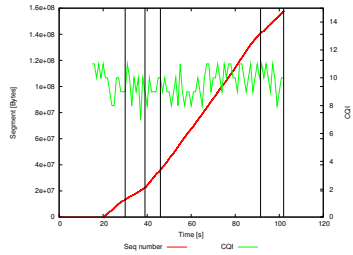

Figure 8. TCP time/sequence graph (`Receive Window` $>$ BDP)

C. Streaming experiments

1) Static scenario: Several streaming sessions were ran over both UDP and TCP. We were forced to use TCP for the sake of the NR-VQM measurements. The NR-VQM plug-in was running in a virtual machine on the Windows XP laptop, and UDP port mapping conflicted with the *Network Address Translation* (NAT) configuration on the virtual machine software. UDP sessions were

performed directly from the host operating system, without the virtual machine being involved.

While streaming, congestion was simulated by running a speed test on `www.speedtest.net` on the same machine. This event triggered an increase in the RTT and determined a decrease in the *Mean Opinion Score* (MOS) value computed by the NR-VQM algorithm on the client side, where the viewer could observe some jerkiness in the playback. The duration of the congestion period was 10 s, as can be seen from the throughput graph in Fig. 9. The MOS is also shown on this graph. The sharp drop of the score illustrates the impact of the congestion period onto the user experience. The corresponding RTT evolution is displayed in Fig. 10. As we can see, when the player runs on the virtual machine (TCP case), the E2E delay is much higher, 1.5 s in average, compared to the UDP case which exhibits the typical LTE RTT (35-40 ms). Also, the reported RTT variation is very high, frequently exceeding 1 s from one RTCP report to another.

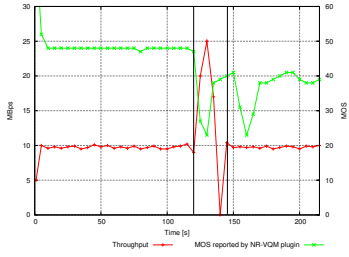

Figure 9. Overall throughput and MOS evolution (static, UDP)

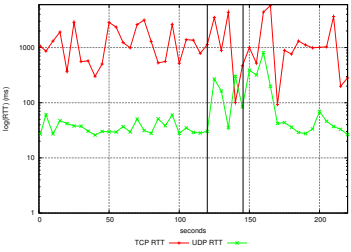

Figure 10. RTT evolution during static streaming session

In [17] the authors compare the delay increase induced by different Linux-based virtual environments. This increase could go up to 100 ms in worst-case scenarios, when heavy contending TCP traffic was present. Since we used a more general, commercial, virtualisation solution which was not optimised for our specific set-up, we believe RTTs higher than the 100 ms observed in [17] are possible, and we therefore blame the huge E2E delay of the TCP case to the use of virtualisation. Due to this behaviour, the MOS reported by the player is much lower than the usual value, which should be 100 when the playback is smooth. Although this affected the results, we can consider the nominal MOS=45 as valid since the playback quality was not much affected by this. However, during the congestion period the MOS dropped quite significantly and playback jerkiness could be observed by the authors. Because the NRVQM algorithm has a good correlation of 0.9 with a standard subjective quality evaluation scale [12], [13], we can map these results onto that scale. Thus, for the congestion period, the results indicate a "Fair to Good" video quality, where a MOS of 50 means "Excellent" quality in this case.

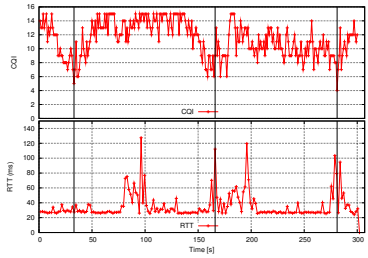

Figure 11. Wideband CQI and RTT evolution (mobile, UDP)

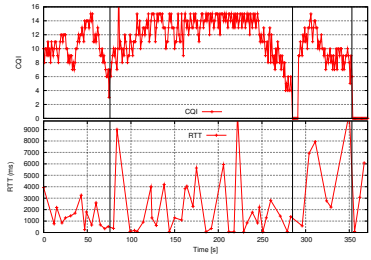

Figure 12. Wideband CQI and RTT evolution (mobile, TCP)

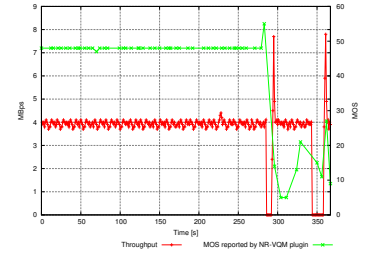

Figure 13. Overall throughput and MOS evolution (mobile, TCP)

2) Mobile scenario: The experiments from the static scenario have been repeated while the UE was moving between the 2 LTE cells. Besides the RTT and the MOS, different cell parameters (e.g. `CellID`) were logged in order to determine when a handover was triggered. Logs were collected from both UDP and TCP sessions.

On top of the streaming session, an additional FTP transfer of a large file was initiated in order to increase the total throughput up to the network's capacity. This would simulate a worst case

scenario, when a user would be watching a video encoded at a rate close to her/his maximum achievable throughput. In this case, even a small decrease in signal quality could affect the playback.

In Fig. 11 the three handovers triggered during the UDP streaming session are marked with vertical lines. Their effect can be observed on the RTT graph. However, the first spike on the RTT graph was produced long after the first handover occurred, when the signal quality was very good again. It is likely to rather be a network issue on the way to the server than an access issue in the LTE network. Despite these handovers, no packet loss occured and the playback quality was perfect.

We repeated the experiment, this time using a TCP connection with the virtual machine and logging the MOS computed by the NR-VQM plug-in. In Fig. 12 we can see a plot of the CQI evolution during this test with the corresponding RTT. A first handover took place after approximately 75 s (marked on the graph with a vertical line). Close to the end of the session we can observe two sections where the CQI is missing. That happened because the UE passed through a "blind spot" where the connection was lost and the UE entered the cell detection mode. The first handover produced just an increase in the RTT, but the signal loss forced the playback to stop, causing very high RTTs and a decrease of the MOS. When the connection re-established, the playback resumed but since the UE was in a low reception zone, the quality was still low, as seen from the VQM graph in Fig. 13. Finally, the signal was lost again and the streaming session ended before the connection was restored.

V. Conclusion

Experiments performed on a test LTE mini-network have shown that the 3GPP requirements have been achieved: user-plane RAN RTT in the order of 10 ms, average data rates above 10 Mbps. These experiments also revealed that a proper antenna spacing can significantly improve the performance of the connection. QoS and QoE parameters were monitored for several streaming sessions to see how network congestion or mobility consequences affect the playback quality when streaming close to the maximum achievable throughput. The QoE indicated by the NR-VQM measurements showed Fair to Good video quality during congestion period in the static session and Poor or Bad quality during handovers. Although the results were affected by the use of a virtual machine, we can still see that congestion or signal degradation can affect the playback quality, especially when TCP is used as the transport protocol.

Acknowledgment

The authors would like to thank the management of the operator for giving us the opportunity to stress their mini-network, and its staff for the assistance during the tests. They would also like to thank the anonymous reviewers of an earlier version of the paper, submitted to the IEEE 2011 International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM 2011), for raising the IPv4 fragmentation issue.

The third author would also like to acknowledge the support of the Walloon region as part of the WIST 2 *Worthy visuAL Content on Mobile* (WalCoMo) project (Grant #616,448) [18].

References

[1] Allot. Global Mobile Broadband Traffic Report Shows Significant 72% Growth in Worldwide Mobile Data Bandwidth usage in H2, 2009. [online]. February 8, 2010 [Cited May 12, 2011]. Available from: <http://www.allot.com/Allot_MobileTrends_Report_Shows_-Significant_Growth.html>.

[2] R. Pantos. HTTP Live Streaming (draft-pantos-http-live-streaming-05. Technical report, November 2010. [online]. November 19, 2010 [Cited May 12, 2011]. Available from: <http://tools.ietf.org/html/draft-pantos-http-live-streaming>.

[3] Jonas Karlsson and Mathias Riback. Initial Field Performance Measurements of LTE. *Ericsson Review*, 85(3):22–28, 2008.

[4] Johan Furuskog, Karl Werner, Mathias Riback, and Bo Hagerman. Field Trials of LTE with 4x4 MIMO. *Ericsson Review*, 87(1):10–15, 2010.

[5] R. Irmer, H.-p. Mayer, A. Weber, V. Braun, M. Schmidt, M. Ohm, N. Ahr, A. Zoch, C. Jandura, P. Marsch, and G. Fettweis. Multisite field trial for LTE and advanced concepts. *Communications Magazine, IEEE*, 47(2):92–98, february 2009.

[6] J. Robson. The LTE/SAE trial initiative: Taking LTE-SAE from specification to rollout. *Communications Magazine, IEEE*, 47(4):82–88, april 2009.

[7] Kaltenberger, Florian and Ghaffar, R. and Latif, I. and Knopp, R. and Nussbaum, D. and Callewaert, H. and Scot, Gaël. Comparison of LTE Transmission Modes in Rural Areas at 800Mhz. COST 2100 TD(10)12080, November 2011.

[8] Rumney, Moray. *LTE and the Evolution to 4G Wireless : Design and Measurement Challenges*. Agilent Technologies, 2009.

[9] TMA. Monitoring 3G Cellular Networks. [online]. 2010 [Cited May 12, 2011]. Available from: <http://www.tma-portal.eu/topics/monitoring-3g-cellular-networks/>.

[10] C. Serrano, B. Garriga, J. Velasco, J. Urbano, S. Tenorio, and M. Sierra. Latency in Broad-Band Mobile Networks. In *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pages 1 –7, April 2009.

[11] Tobias Hoßfeld and Andreas Binzenhöfer. Analysis of Skype VoIP traffic in UMTS: End-to-end QoS and QoE Measurements. *Computer Networks*, 52(3):650 – 666, 2008.

[12] Jean-Charles Gicquel Ricardo R. Pastrana-Vidal. Automatic quality assessment of video fluidity impairments using a no-reference metric. [online]. 2006 [Cited May 12, 2011]. Available from: <http://de.o2.com/ext/o2/wizard/index?page_id=16677; category_id=;state=online>.

[13] Jean-Charles Gicquel Ricardo R. Pastrana-Vidal. A no-reference video quality metric based on a human assessment model. [online]. 2007 [Cited May 12, 2011]. Available from: <http://enpub.fulton.asu.edu/resp/vpqm2006/papers06/311.pdf; category_id=;state=online>.

[14] Ronny Henkes. Adaptation automatique de la qualité d'un streaming vidéo par les protocoles RTP/RTCP/RTSP. [Master Thesis] Facultés Universitaires Notre-Dame de la Paix, Namur, June 2010 [Cited May 12, 2011].

[15] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network. Requirements for E-UTRAN. Technical Report TR 25.913, December 2004.

[16] M. Allman, V. Paxson, and E. Blanton. TCP Congestion Control (RFC 5681). Technical report, September 2009. [online]. September, 2009 [Cited May 12, 2011]. Available from: <http://tools.ietf.org/html/rfc5681>.

[17] Jon Whiteaker, Fabian Schneider, and Renata Teixeira. Explaining packet delays under virtualization. *SIGCOMM Comput. Commun. Rev.*, 41:38–44.

[18] WalCoMo. Worthy visuAL Content on Mobile project website. [online]. May 12, 2011 [Cited May 12, 2011]. Available from: <http://www.fundp.ac.be/recherche/projets/page_view/-07296002/>.

A.3 An Interactive Video Streaming Architecture Featuring Bitrate Adaptation

Published in: Journal of Communications, Vol 7, No 4 (2012)

An Interactive Video Streaming Architecture Featuring Bitrate Adaptation

Ivan Alen Fernandez, Christophe De Vleeschouwer
ICTeam, Université Catholique de Louvain, Belgium
George Toma, Laurent Schumacher
FUNDP Namur, Belgium
Email: {ivan.alen,christophe.devleeschouwer}@uclouvain.be
{george.toma,laurent.schumacher}@fundp.ac.be

***Abstract*— This paper describes an interactive and adaptive streaming architecture that exploits temporal concatenation of H.264/AVC video bit-streams to dynamically adapt to both user commands and network conditions. The architecture has been designed to improve the viewing experience when accessing video content through individual and potentially bandwidth constrained connections. On the one hand, the user commands typically gives the client the opportunity to select interactively a preferred version among the multiple video clips that are made available to render the scene, e.g. using different view angles, or zoomed-in and slow-motion factors. On the other hand, the adaptation to the network bandwidth ensures effective management of the client buffer, which appears to be fundamental to reduce the client-server interaction latency, while maximizing video quality and preventing buffer underflow. In addition to user interaction and network adaptation, the deployment of fully autonomous infrastructures for interactive content distribution also requires the development of automatic versioning methods. Hence, the paper also surveys a number of approaches proposed for this purpose in surveillance and sport event contexts. Both objective metrics and subjective experiments are exploited to assess our system.**

***Index Terms*— interactive streaming, clip versioning, RoI extraction, bitrate adaption, H.264/AVC.**

I. Introduction

Streaming services are becoming the highlight of value-added mobile services. Lately, the number of streaming applications developed on smart and cell phones increased dramatically, to give access to more and more multimedia contents. Based on the latest developments of the wireless data network, and the adoption of compression technologies such as H.264 [1]–[3], several media players have been designed and implemented for mobile handsets. In addition, due to the massive diversification of mobile users, and because of the shortage of mobile network bandwidth, the concept of client profile has been earning more and more importance. Its default purpose is to offer different streaming quality levels and different contents to the clients, depending on the purchased services.

This paper introduces an integrated architecture to support service diversification through adaptive and interactive streaming capabilities. The proposed system aims at offering personalized experience when accessing high resolution video content through individual and potentially bandwidth constrained connections. Fundamentally, the underlying architecture relies on the concatenation of pre-encoded clips to adapt to a pre-defined set of user commands, as well as to the network conditions. On the one hand, the user commands typically give the client the opportunity to select interactively a preferred option among the multiple video clips that are made available to render a given scene, e.g using different view angles, or different zoomed-in and slow-motion factors. On the other hand, adaptation to the network bandwidth is obtained through intelligent and dynamic switching between the multiple versions of the content that have been generated by encoding the same video at different quality (and thus bitrate) levels. The implementation of an effective switching strategy adapts the bit-rate of the delivered stream to the available bandwidth of the current link. It ensures accurate control of the client buffer, which is fundamental to reduce the client-server interaction latency while maximizing video quality and preventing buffer underflow.

Now that we have introduced the main principles of our proposed architecture, we detail the motivations underlying our investigations, and stress the arguments that make our work original, relevant and timely.

The need for interactive mobile streaming solutions naturally arose from the two following observations. At first, due to mobile network bandwidth limitation, it is often not possible to transmit large rate video streams, which in turns constraints the resolution of the streamed video images. On the other hand, content produced for conventional wired broadcast or captured by surveillance networks is gaining in resolution. As a consequence, this content has to be down-sampled to be accessed through mobile networks, thereby losing a lot of its value. A possible solution might be to manually produce a second version of the content that is dedicated to low resolution accesses. However, this solution is expensive and not ap-

This paper is based on "Browsing Sport Content Through an Interactive H.264 Streaming Session," by I. A. Fernandez, F. Chen, F. Lavigne, X. Desurmont, and C. De Vleeschouwer, which appeared in the Proceedings of the 2nd International Conference on Advances in Multimedia (MMEDIA), Athens, Greece, June 2010. © 2010 IEEE.

This work was supported in part by Walloon Region projects Sportic, Walcomo and Belgian NSF.

Manuscript received April 15, 2011; revised July 15, 2011; accepted October 15, 2011.

propriate in many application scenarios (e.g. surveillance or real-time post-production of broadcast content). For those cases, the only alternative is to design automatic video processing systems that produce low resolution content out of the native high-resolution content. Simple down-sampling of the native content is not appropriate since it results in the loss of many visual details. A preferred solution consists in cropping the initial content, to focus on Regions-of-interest (RoI). Such kind of automated tools have already been investigated in the literature [4]–[9], and a general conclusion is that none of the existing method is 100% reliable in the way it defines RoIs. Therefore, human supervision of the process is always required to check that the automatic content adaptation system behaves properly. Besides, in some cases, more than one region are likely to be of interest for the user. Our interactive framework proposes to circumvent those issues by allowing the end-user to decide about the rendering option he/she would like to visualize among a finite set of options that have been pre-computed based on automatic systems. Conversely, the above observation also reveals that the recent advances in automatic analysis and production of content [10]–[13] offer an unprecedented opportunity to deploy interactive and personalized services at low cost. In particular, the ability to identify the spatial regions or the temporal actions of interest in a video directly supports the automatic creation of several options to render an event, e.g. by skipping non-interesting actions or zooming on RoIs. Hence, no manual pre-processing of the content is required any more before actual exploitation of the interactive infrastructure.

Our paper develops and assesses the integrated components involved in the deployment of an interactive and adaptive streaming solution. The main contributions of the paper include:

- The design of the adaptive streaming architecture, based on the temporal concatenation of pre-encoded video clips. In practice, client-transparent switching between versions is enabled by splitting each produced video in short clips that are temporally pipelined by the server, based on user's requests, network conditions estimation or video content metadata and interaction strategies. From the client's perspective, everything happens as if a single pre-encoded video was streamed by the server. This is in contrast with the solution developed in [14], which supports continuous interactive Pan/Tilt/Zoom navigation while streaming high-resolution content, but therefore relies on dedicated spatial-random-access-enabled video coding.
- The development of control mechanisms, to adapt the streaming rate to network bandwidth. A number of works have already addressed the problem of adapting the sending rate of a streaming session to match the observed network conditions. Our work fundamentally differs from those earlier contributions by the fact that it puts a strong emphasis on maintaining a small client buffer all along the streaming session, thereby reducing the interaction latency[1]. This is obtained through the definition of an original and cautious probing strategy combined with careful analysis of the RTCP feedbacks.
- The definition of interactive commands, and the development of automatic methods to extract multiple rendering options out of a single high-resolution video. Such automatic versioning methods are indeed required to support the deployment of fully autonomous infrastructures for interactive content distribution. To address this issue, we survey some of our earlier contributions [15], [16] to explain how different video streams can be extracted out of high resolution content in a fully automatic manner both in the videosurveillance [17]–[20] and sport broadcast context [21]–[24]. Spatial and temporal adaptations are considered. Spatially, we crop the high resolution content to extract a lower resolution image that focuses on some automatically detected RoI(s). This solution provides an alternative to the regular sub-sampling of the initial content. Temporally, the automatic segmentation of the event into semantically meaningful actions or events can support fluent and efficient browsing across the video sources. Notably, a significant advantage of the interactive access scenario, compared to the fully automatic creation of personalized content, is that it gives the last decision about the way to vision the content in the hands of the final user. This is especially important since most video analysis tools remain prone to errors. Subjective tests have been considered to assess the experience offered to end-users by the proposed interactive architecture. They demonstrate the relevance of the approach.

The remaining of the paper is organized as follows: Section II presents the proposed architecture for interactive video streaming, through client-transparent temporal concatenation of pre-encoded video clips. In Section III, we describe the algorithm for bit-rate adaptation. Section IV introduces the interaction commands, together with automatic tools to version video surveillance and broadcast content. Finally, Section V presents some qualitative and quantitative results to validate our system. Section VI concludes.

II. Interactive Browsing Architecture

The main objective of our architecture is to offer interactivity to any client of a mobile video streaming session using an H.264/AVC compliant player. At the same time, the architecture supports bit-rate adaptation, so as to match dynamic bandwidth constraints while maximizing playback quality. Both capabilities are offered based on a

[1]Reducing latency by trashing the client buffer when the user sends a clip switching command is not a desired solution since it would result in a waste of resources. It would also significantly increase the system complexity, due to the need to inform the client about the actual fraction of the buffer that should be trashed to save latency while preserving transparent and continuous streaming.

generic content pipelining feature. As depicted in Figure 1 the communication is established with the client through the Real Time Streaming Protocol (RTSP). Video is forwarded using the RTP protocol. RTCP feedbacks are then used for dynamic bit-rate adaptation, while RTSP commands interpretation supports interactive browsing capabilities. In this section, we briefly introduce the different modules involved in the architecture. Additional details regarding bitrate adaptation and content versioning for interactive streaming will be presented in Sections III and IV, respectively.

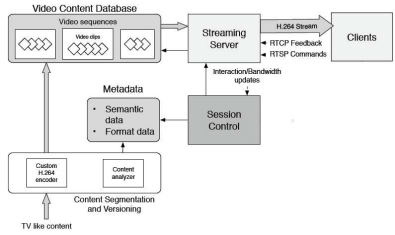

Figure 1. Diagram of the architecture's workflow

A. Architecture of the Streaming Server

The architecture on the server side is composed of 3 main components: the content segmentation and versioning unit, the streaming server and the session control module.

1) *The Enhanced Content Creation Unit* fills the Video Content Database, without actively taking part afterwards in the streaming system. Its purpose is threefold:

- It analyses the TV like video content to identify RoIs and produce several versions (replay, quality, view angle etc.) and zoomed-in alternatives of the content.
- It divides the video sequences in small pieces that are encoded based on H.264 according to the requirements explained in sections II-B and IV.
- It generates the metadata (shown in Section II-C) that are required to model and define the interactive playing options and quality versions associated to the different clips. Therefore, the metadata information is used by the session control to monitor the streaming session in response to the interactive user requests.

2) *The Streaming Server Module* is the core of the system, which supports client-transparent interactive streaming through on-the-fly content pipelining. Client-transparent temporal content pipelining allows the server to stream multiple successive video streams in a single session, without negotiating with the client the establishment of a new streaming session. Hence, with this feature the server is able to change the streamed content while maintaining a unique output stream and keeping the existing session uninterrupted. As a result, both a temporal and computational gain are achieved as the client does not need to establish more than one streaming session. The streaming server delivers all the data content through the Real-time Transport Protocol (RTP).

3) *The Session Control Module* determines, at every moment, which video clip has to be transmitted next. This unit consequently decides the video clips that are concatenated based on requests from the client, the estimated network status and on alternative versions offered by the enhanced content creation unit. Therefore, the session control is an essential part of the system, as it monitors almost any information flowing through the system.

B. Temporal Content Pipelining

Temporal content pipelining is the technique that allows a streaming server to juxtapose multiple streams in a single continuous sequence, so that multiple streams can be forwarded to the client through a unique and fluent streaming session. The key for implementing this functionality is the session control module using the advanced features of the H.264 codec [25], regarding the encoding parameters transmission.

The H.264 standard defines two kinds of parameter sets: sequence parameter sets (*SPS*) and picture parameter sets (*PPS*). The first applies to a wide range of frames, while the latter only applies to specific ones. Every Network Adaptation Layer (*NAL*) unit containing data information includes in its header a parameter linking it to a *PPS*, which in turn links to a specific *SPS*. In our architecture, all clips are encoded independently from each other. Since the first *NAL* unit of an H.264 segment always contains the *SPS* and the *PPS*, multiple sequences can be transmitted consecutively without any interruption, and the output is still compliant to the H.264 standard. When necessary, on the client's side, a unique sequence is received, which however, is built step by step by the server. The *SPS* are updated between two pipelined segments.

C. Session Control and Metadata

The session control processes the user's feedback, the RTCP statistics, and uses the metadata associated to the clips, to decide which clip should be delivered next. As described in Section IV, the metadata information is generated by the content (segmentation) and versioning unit, and is stored within a Extensible Markup Language (XML) file.

From a semantic point of view, we distinguish two different cases on the server side, depending on whether storytelling continuity has to be ensured or not when switching between clips. When temporal continuity is required, clip switching can only occur at the intersection between two consecutive clips. Those instants are depicted with vertical dashed lines in Figure 2. For this reason, the sequences have to be divided in very small clips, as each clip has to be completely delivered before switching. Otherwise the browsing capabilities would

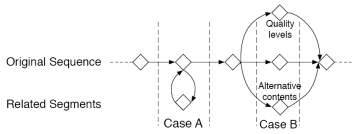

Figure 2. Metadata considered structures

only be offered on a coarse granularity basis. In cases for which temporal continuity is not required, as happens when the user wants to skip some non-relevant content, any buffered data in the server is discarded, so as to start reading the new clip file as soon as possible, thereby reducing to the minimum the overall latency associated to the switching mechanism. Like in the previous cases, the playback proceeds without causing any decoding error and the streaming behaviour is not damaged, performing the switching flawlessly.

From a functional point of view, two different kinds of temporal relationships between clips are envisioned, as depicted in Figure 2. Case *A* typically corresponds to an optional inclusion of a replay within the stream. The sequence playback is resumed after the additional content without any offset. The same relationship can be considered if *target advertising* is inserted in the stream according to the user preferences. In contrast, case *B* considers contending versions, which means that only one version is actually included in the output stream. As an example, possible contending alternatives include videos at different resolutions (zooming), fast-forward/regular speed mode, and different video quality versions. Hence, this case is extensively exploited to react to the interaction commands sent by the client. In Section IV, we define those commands, and survey a number of solutions that can be used to automatically generate the multiple rendering options that are of interest to the user, when visualizing high-resolution surveillance or sport event content. In addition, this case *B* also provides the possibility to switch between different quality (and thus bit-rate) levels, depending on the bandwidth limitation and in a completely transparent way for the user. In Section III, we explain in details how network probing can be implemented to infer the state of the network by increasing the burstiness of sent-out packets. We also describe how RTCP feedback monitoring can be exploited to decide at which rate the content should be forwarded by the server. Those two aspects are fundamental to adapt to mobile network fluctuations, thereby preserving video quality while limiting the size of the client buffer, and thus the interaction latency.

D. Interactivity with Video Player

The system's interactivity relies on the RTSP commands that are exchanged between the server and the client. This communication channel is already established and can be used to obtain feedback from the client. The user must be able to send a switching command, which induces a system response according to its content. The browsing features are then triggered by sending the appropriate request to the server.

A standard RTSP message is used by the client player to communicate its feedback. The considered RTSP command in our architecture is *OPTIONS*, as described in [26]. Combined with the header *Require*, it provides an efficient and simple way to signal user's feedback during the transmission. A specific value in the field of this header such as "*Switching-feature*", directly associates the RTSP command with the browsing capabilities of our server. A new line in the header, starting like "*Switching-Parameter:* " signals and conveys the different possible requests of the user (zooming, replay or fast forward mode). The mentioned interactive requests are associated one-by-one to new-functional buttons of the player's interface. These buttons consequently trigger a RTSP command from the user side when they are pressed. As an alternative, many clients such as the VLC Video Player, implement a seek function by sending the command PLAY with a parameter called Range [26]. Not only does it trigger a stream playback, but it may also seek inside the stream. While our server has been designed to attend such request, the browsing capabilities are further limited by this scenario. As an example, in a multi-angle camera scenario, the user has to send several requests to switch in between all the available sequences in round-trip without being able to access directly to the desired one.

III. Automatic Bit-Rate Adaptation Through Version Switching

To ensure a good user experience when streaming in wireless networks, it is necessary to adapt the streaming rate to frequent bandwidth variations. The video bit-rate should be reduced in the presence of congestion or a low quality link, but should be kept as high as possible to maximise the image quality. This section investigates the control problem in the particular case of our proposed interactive streaming framework. In Section III-A we present previous work related to stream adaptation and motivate why a new technique is needed to improve interaction delay, besides received video quality. Section III-B shows how congestion or signal degradation is detected. Eventually, Section III-C introduces our proposed rate adaptation algorithm, which prevents client buffer starvation in presence of congestion (to preserve playback fluency), while keeping the streaming rate close to the available bandwidth. Sections III-D and III-E explain the proposed probing mechanism used to determine the available bandwith and to keep a reasonably small buffer to preserve interactivity.

A. Motivation of the Chosen Adaptation Algorithm

In an interactive streaming scenario, there are two elements that contribute to improve the user's experience:

- The received video quality;
- The reactivity of the streaming system to user interactions.

Maximising the received video quality is a challenging task, especially in the context of varying mobile network conditions. This means that in the case of bandwidth constrained connections the playback should remain smooth without re-buffering or jerkiness and when the network allows, the viewer should receive the best achievable image quality. This translates into the ability to select the appropriate encoding rate of the chosen content, based on the available throughput.

A number of bit-rate adaptation techniques have already been proposed in the literature, but they generally don't care about interactivity. Even more, some of these solutions like the ones presented in [27], [28] and [29] require a custom-built client which would limit the use of the adaptive streaming framework to those specific media-players.

We propose a bit-rate adaptation algorithm that attempt to maximize both the received video quality and the system reactivity. In an interactive system, it is essential to keep the reaction time as low as possible, the delay between a request at the client side and the consequence of that action in terms of played content should be minimised. This delay has 3 contributions:

1) The server side delay, if the request arrives just after the initial frames of a clip (temporal consistency is targeted).
2) The end-to-end (E2E) delay, from the server to the client through the network
3) The time required to empty the pre-roll buffer, since there is no remote possibility to flush the video buffer of a player.

The first contribution depends on how the video stream is split into clips to support interactive services. As explained in Section II-C, we recommend to use short clips, thereby reducing the upper bound of this delay to 700 ms. More details about this issue can be found in [15]. The second component is imposed by network at hand and is independent of the server and the client. The third component of delay depends on the client buffer fullness when an interaction command is launched by the user. It can be reduced by trying to keep the client buffer level as low as possible. This can only be achieved in the presence of fine rate adaptation mechanisms. Those mechanisms have a double objective: they attempt to maximize the streaming rate while preventing the buffer to become empty when the network conditions become worse. Working with a small buffer makes the problem especially challenging since it increases the risk of interrupting the playback. Hence, rate adaptation is severely constrained by interactivity, which imposes to keep small buffers.For this reason, in Section III-C we propose an original probing mechanism that empties the buffer to a certain level by a pause in the transmission, the so-called gap, while the following burst of packets brings back the buffer to its normal position. This approach allows to probe the network because, during the burst, data is forwarded at a faster instantaneous rate than the average streaming rate. Both the congestion detection and the network probing methods are further described below.

B. Estimating Network State

Network conditions are estimated from the information sent back by the client through the periodic RTCP reports. Specifically, the Receiver Reports (RR) and the Sender Reports (SR) from the RTCP protocol will be used to compute the RTT, jitter, packet loss and average throughput.

The RTT[2] can be computed by the server using the method presented in [30]. A fast increase in the RTT suggests that congestion is about to take place in one or more links across the network path. Because the variation nature of the instantaneous RTT is spiky, two variables will be used to characterize its evolution: a smoothed RTT and the RTT deviation. The formulas are inspired by the method adopted to compute TCP Retransmission timer [31], but have been modified so as to:

- Know whether the deviation increases or decreases
- Increase the impact of the instantaneous measurements compared to past reports

The formulas write as follows:

$$\begin{aligned} &\text{Smoothed}_{RTT} \\ &= (1-\alpha) * \text{Smoothed}_{RTT} \\ &\quad +\alpha * \text{Instant}_{RTT} \end{aligned} \tag{1}$$

$$\begin{aligned} &\text{Deviation} \\ &= (1-\beta) * \text{Deviation} \\ &\quad +\beta * (\text{Instant}_{RTT} - \text{Smoothed}_{RTT}) \end{aligned} \tag{2}$$

where α and β are both 0.5.

The network state is then inferred from the evolution of the Deviation parameter over a specific number of consecutive RTCP reports. As a rule of thumb, we consider that a network encounters congestion once the Deviation value is higher than 100 ms for two consecutive reports. The rest of the paragraph illustrates the empirical study that has led to this rule. Figures 3 and 4, plot the formulas in (1) and (2), along with the instantaneous RTT in two distinct scenarios. In Figure 3, the Network Emulator (NetEm) Linux module is used to reduce the network bandwidth to the video bitrate, for a limited time period. In Figure 4, the RTT distribution is based on measurements in live 3G networks [32]–[34]. We focus on GPRS measurements as they exhibit the largest RTT variations. One observes that the Deviation only goes above 60-70ms (in absolute value) when the current transmission rate is close to the maximum available bandwidth, but remains under this value in absence of congestion even in the case of a GPRS connection, whereas the jitter is higher than in other mobile networks. Also, the authors of [35] have reported that in 90% of the cases, the jitter was smaller than 100ms in their measurements.

Consequently, if the absolute Deviation value is higher than 100 ms for two consecutive reports, this should be

[2]Despite being a two-way time measurement, the RTT is regarded as a good estimate of the E2E delay.

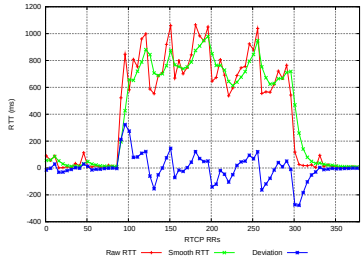

Figure 3. RTT evolution in a congested network. Bandwidth reduction applied at 90s and removed at 300s.

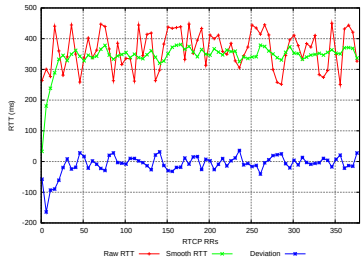

Figure 4. RTT evolution in GPRS network.

interpreted as a sign of congestion. To increase decision robustness, a large number of RTCP RRs should be taken into consideration. However, when the media client supports a standard implementation of the RTP/RTCP protocol, it sends RTCP reports every 5 s (like VLC, QuickTime). Waiting for more than 2 reports would therefore lead to a reaction time longer than 10 s, which is not acceptable. Hence, in practice, decision about congestion state is taken based on two observations of large RTT deviation.

As explained above, alternative clues for congestion detection lie in the fields of the receiver report that are related to lost packets, namely the fraction loss and the cumulative loss. The first represents the ratio between the number of lost packets and expected number of packets since the emission of the last RR or SR, while the latter represents the number of lost packets since the beginning of the session.

A combination of the two reports will be used to decide about congestion and to consider a down-switch in the transmission rate. Specifically, using the current and previous RRs, the server can compute the total number of lost packets for the reporting interval:

$$\begin{aligned} \text{nr\_lost\_packets} &= \text{current\_cumulative\_report} \\ &\quad -\text{previous\_cumulative\_report} \quad (3) \end{aligned}$$

This value, combined with the fraction loss provides insight into the loss status. Congestion is inferred when a sufficient number of packets has been lost on a sufficient long history, or equivalently when a sufficient number of packets have been lost on a sufficient recent history. In practice, congestion is assumed when packet loss ratio is higher than 10% and the total number of lost packets between the current and the previous RTCP report is higher than 10 packets. These threshold parameters were chosen after several simulations under a QDisc limited Ethernet network and in a real WiMax and WiFi access networks. Also, the results presented in [35] and [32]–[34] were taken into consideration

C. Adaptation Algorithm

As stated in Section II-C, the *Streaming Server Module* has the ability to switch between different H.264 encoded clips, meaning that it can seamlessly switch between versions of the same video encoded at different rates. We therefore define a down-switch as the change in the streaming chain to a lower quality encoding and the up-switch the change to a higher quality encoding. The adaptation algorithm is based on congestion detection and network probing mechanism. Our proposed scheme is presented in Figure 5 as a Finite State Machine (FSM) with three main phases: Initialisation, Probing and Normal.

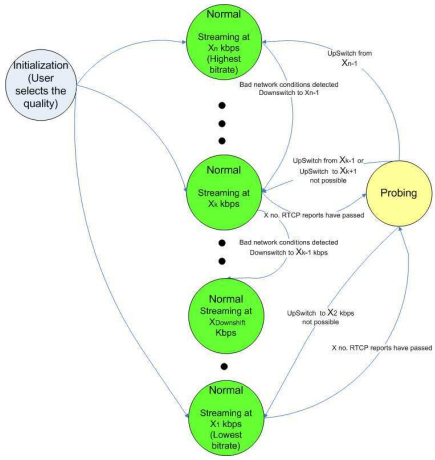

Figure 5. Adaptation algorithm

1) Initialisation state: This is the initial phase of the algorithm, it includes the RTSP negotiation and network discovery, when the server collects statistics about the current state of the network. The first two RTCP reports are used for the initialisation of $Smoothed_{RTT}$ and $Deviation_{RTT}$. In this phase, the server sends the video encoded at a bitrate that is close to the quality requested by the user.

2) Normal X_k state: In this state the server sends the media at a constant rate of X_k kbps and analyses the RTCP reports, where $k \in \{1..N\}$ and N is the number of supported bitrate versions. From here, depending on

network conditions, the server can remain in the same state, can pass to X_{k-1} through a down-switch or can go to the $Probing$ state to assess whether there is enough bandwidth to up-switch to X_{k+1}. Consequently, it is necessary to define possible bitrates for the X_k states and the moments when a change of state is needed.

According to [27], a Fast bitrate Decrease Slow bitrate restore Up (FDSU) approach is the most suitable way to assess network congestion. Using this method, the server switches to a clip encoded at approx. 60% of the current encoding, avoiding severe image degradation. However, a slow bitrate increase implies producing many quality versions of the same content, which puts a burden on the post processing and storage of several versions. Hence, we will use a Fast bitrate Decrease Fast bitrate restore Up (FDFU) approach. This approach implies that the number of X_k states can be reduced to 3. For example in a cellular environment, each state would be defined to encompass the average rate of a cellular generation, e.g. EDGE (140 kbps), UMTS (200 kbps) and HSPA (350 kbps), as measured in [36].

A down-switch is performed when the network bandwidth cannot support the current streaming rate. As explained in Section III-B, this means that is should be triggered when the RTT Deviation is higher than 100ms for 2 consecutive RTCP reports, or when packet loss ratio is higher than 10% and the total number of lost packets between current and previous RTCP reports is higher than 10 packets. To increase the responsiveness of the algorithm, if the deviation exceeds 300 ms, the server will immediately down-switch to a lower rate because such high values indicate severe network congestion. The server repeats the down-switch only if the deviation continues to increase.

After a configurable number of RTCP reports, the server will go into the probing state only if the network does not have signs of congestion.

3) Probing state: This is an auxiliary state, in which silent gaps and bursts of RTP packets alternate in order to estimate whether additional bandwidth is available in the network. The main idea behind this technique is to send the video frames at a higher rate (burst) to put the network under stress. If the bandwidth limit is close to the current bitrate, the packets sent at a higher rate would queue in the network buffers and the RTCP would report high RTTs at the server. Consequently, from the RTT values reported in the RTCP reports, the streaming server can assess whether the available network bandwidth is high enough to switch to a higher bitrate. If this is the case, then the server should switch from X_k kbps to X_{k+1} kbps. Otherwise it will resume regular streaming at X_k kbps.

The advantage of this probing technique compared to the tools that compute the available network bandwidth by sending packet trains or packet chirps (for instance abing [37], pathchirp [38] or Wbest [39]) is that it does not send extra data over the network. In addition, there is no need for deploying a tool on the client side to analyse the packets.

A possible drawback of the proposed approach is the fact that the amount of data which can be sent at a faster rate is limited due to the risk of client buffer overflow. To overcome this issue, the burst of RTP packets has to be followed or preceded by a pause in the transmission, the so-called gap). This allows the data from the buffer to be consumed, or to refill the buffer to its average occupancy respectively. If we choose to have the burst first, followed by the gap, we minimize the risk of emptying the client buffer, but increase the average size of the buffer during the probing process which impairs interactivity.

Since we aim to reduce the interaction time as much as possible, we did choose to pause the transmission first and then send the burst to probe the network.

D. Choosing burst and gap size for probing period

When probing the network, we would like to know if the current available bandwidth allows to switch to the next encoding rate, which should be at about 60% higher than the current rate, according to FDFU approach. However, since streaming closely to the bandwidth limit could lead to high RTT and packet loss, we have decided to up-switch only when the bandwidth limit is almost twice as high as the current streaming rate.

Because we want to have a neutral impact on the buffer after a complete probing cycle, the length of the gap is strictly related to the length of the burst. We therefore define the Gap, in seconds, as below:

$$\begin{aligned} \text{Gap}_{duration} \\ = \quad & (Burst_{Length}) * \frac{1}{FPS} - (Burst_{Length} - 1) \\ & * \frac{1}{FPS} * \frac{1}{Probing_{Factor}} \end{aligned} \tag{4}$$

where $Burst_{Length}$ represents the probing duration expressed in number of frames, FPS is the video frame-rate and the $Probing_{Factor}$ represents the frame-rate increase. For example, for a 25fps video, if we stream at twice the rate (the $Probing_{Factor}$ would be 2, streaming at 50 fps, but keeping the same presentation time, so the frames would be displayed at the correct speed to the viewer) for 31 frames, the 32nd frame would represent the Gap of 660 ms. We have discovered however that sending the video at twice the frame-rate, does not put a significant load on the network because the burst period is short compared to the Gap. The $Probing_{Factor}$ was increased to 4, with the same $Burst_{Length}$ of 32 frames which returned a Gap of 970 ms. We could not increase the $Probing_{Factor}$ further because the Gap would increase even more and the media player buffers would need a higher playout value which would affect interactivity. Moreover, for most conventional players, the maximum gap we can make before emptying the buffer is about 1s for robust transmission. As depicted in Figure 6, to stress even more the network, the probing cycle is repeated 6 times which covers the period of 2-3 RTCP reports.

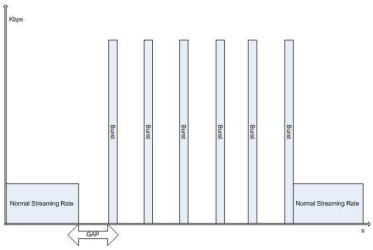

Figure 6. Probing cycle

E. Definition of up-switch thresholds

After each probing cycle, the server has to decide whether to up-switch or not, based on the RTT deviation observed from the RTCP reports. As derived in Annex A, one can associate the expected available bandwidth to the observed deviation in RTT. As explained in Section III-D we aim to switch up only when the available bandwidth is twice as high as the current encoding rate. The mathematical derivations in Annex A, when parametrized based on actual network measurements, reveal that if the deviation observed after probing is smaller than 100ms, there is a 90% chance that the available bandwidth is equal or higher than twice the rate. Hence, we have adopted a threshold of 100ms in RTT deviation to decide whether to up-switch (deviation below threshold) or not (deviation above threshold).

IV. Automatic content definition and versioning

In previous sections, we have presented a an adaptive streaming framework that gives the user the opportunity to switch interactively between multiple versions of a visual content. However, in addition to user interaction and network adaptation, the deployment of fully autonomous infrastructures for interactive content distribution also requires the development of automatic versioning methods. Hence, this section completes the picture by introducing a number of approaches proposed for this purpose in two different scenarios: sport event broadcasting and video surveillance. Typically, the (automatically) produced low-resolution versions include a sub-sampled version of the native video, plus one or several zoomed-in (and cropped) versions, each one focusing on one (of the) RoI(s) detected in the native high-resolution video. In terms of interactive functionalities, users can select the original video which offers a general view of the scene or select videos that focus on specific RoIs. In some application scenarios, replays of hot spots actions are also proposed to the user.

A. Interactive commands and browsing options

In the soccer video context, three browsing capabilities are offered: alternative fast forward mode, replay of hotspots and zooming over the RoI. Figure 7 presents the interaction strategy supported by our framework, initially introduced in [15].

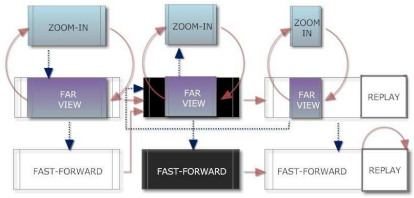

Figure 7. Switching control strategy. Dashed arrows represent potential requests from the user, while continuous arrows depict automatic connections between versions based on the interaction strategy. The central segment corresponds to an important action of the match.

Fast forward mode is available for the user during all the playback. When this mode is active, the video replay of the involved actions is skipped. Every time the playback reaches a highlight segment of the game, the fast-forward mode is automatically switched to regular mode catching the attention of the user. Zoom-in is available in regular mode for far camera shots. The viewer has always the faculty to decide the mode that he/she considers convenient to receive. At the beginning of every new segment the user can request the replay of the segment that has been displayed previously. After the repeated segment is displayed, the playback of the current segment where the replay was requested is recovered without any offset.

For video surveillance, automatic RoI extraction methods are used in order to extract the moving objects of the scene. Examples of such methods are presented in [16], [40] and [41]. Alternative videos are then generated by cropping the areas of the image containing the objects of interest. An example is depicted in Figure 8. The last column contains the available video versions at a given time instant.

Figure 8. New "zoomed versions" of video stream. First row is the original video stream. Second row is created when a first mobile object appears in the scene. Third row is created when a second object is detected and tracked (the abandoned luggage). Forth row is the stream that includes the two mobile objects.

B. Temporal consistency and division in shots, clips and segments

To provide the temporal browsing capabilities, different levels of division granularity are considered. Starting from the native raw content, our system automatically splits it into non-overlapping and semantically consistent segments. Each segment is then divided into shots, based on conventional view boundary detection. Shots are finally split in small clips. These clips support our browsing capabilities during the whole playback in a temporally consistent way, following the metadata considerations described in II-C. Hence, switching between versions should be allowed between shots, meaning that a boundary between shots should also define a boundary between clips. The same holds for segments.

In the surveillance context, the shot denotes the entire video sequence, and segments are segmented based on activity detection. In the sport broadcast context, a shot is defined as a portion of the native video that has been produced with constant or smoothly evolving camera parameters. This approach is based on average difference between consecutive histogram-features as already described in [15]. Afterwards, the shots are classified in different view types: replays, non-grass close-up views and close, medium or far grass view camera. At the end, far views are computed in order to obtain an alternative zoomed-in version that is stored in the enhanced content creation unit. Interested readers may refer to [15] for more details about shot definition, and view type classification. They can also refer to [16] for the automatic generation of zoomed-in versions in case of far view.

Figure 9 presents an example of our framework applied to soccer game. The resolution of a football game video extracted from TV broadcasting is automatically adapted to a small device screen. The zoomed-in sequences are offered to the user as an alternative replacing the original segments upon request.

(a) (b)

Figure 9. Original and processed zoom versions of the same frame.

Finally, segments are defined as shots closely related in terms of story semantic, e.g., shots for an attacking action in football. Proposed by the authors in [42], semantically meaningful segmentation is achieved based on a general diagram of state transition, which consists in one round of offense/ defence as described in Figure 10. For completeness, we note that audio or video analysis tools [43] have been designed to highlight key actions automatically, thereby offering additional browsing granularity. We conclude that many algorithms do already exist to fed or interactive framework in a fully automatic manner, making its practical deployment realistic, since manual intervention is not required to create dedicated interactive content.

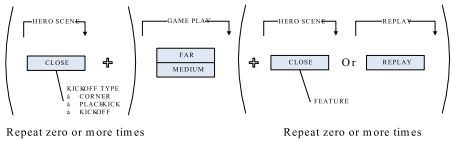

Figure 10. General structure of a gameplay and view types.

V. Tests and Results

A. Performance evaluation of the proposed platform

In this section we perform 3 types of tests to show the improvements in interactivity delay and in quality of experience over the same solution without rate adaptation. Although scalability tests have not been made, being a VoD platform, it inherits the typical VoD scalability issues where multicast and broadcast techniques are not used.

1) Convergence speed of the Rate Adaptation algorithm: In the first experiment we test the reactivity of the algorithm, with the parameters presented in section III-C. The set-up consists 2 PCs, one hosting the modified version of a LiveMedia555 streaming server and the other hosting VLC media player, connected via 100Mbps Ethernet interfaces. The available bandwidth between the two can be reduced using the Linux Traffic Control tool, to simulate congestion or signal degradation in wireless networks. During several streaming sessions the available bandwidth was reduced close to, or below the current streaming rate and the reaction time between bandwidth reduction and an actual down-switch was registered. The cumulative distribution function for the delay is plotted in Figure 11, where 3 cases can be distinguished:

- when the bandwidth drops to a value closer to the current streaming rate, the down-switch delay is in average equal to 10s, which represents the duration of approximatively 2 RTCP reports.
- when the bandwidth drops to a value at least 20% lower than the current streaming rate, the system reacts faster, in about 5-6s, which represents the duration of about 1 RTCP report.
- the bandwidth drops during probing, the down-switch delay is approximatively 10s as well, the duration of 2 RTCP reports

Table I summarizes the performance of the adaptation algorithm in case of a decrease in the available bandwidth.

Bandwidth drop level	Bandwidth = current rate	Bandwidth < current rate	Bandwidth drops while probing
Avgerage reaction time	11.4s (2 RTCP RR)	6.4s (1 RTCP RR)	10.5s (2RTCP RR)

TABLE I.
Summary of down-switch reaction time

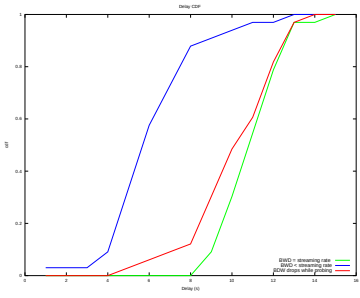

Figure 11. CDF for the down-switch delay

The reaction time directly depends on the frequency of the RTCP reports, and the results obtained in this paper present the worse case scenario, where the media player used sends 1 RTCP report every 5s.

The up-switch delay depends on the probing frequency, which was fixed to each 6 RTCP reports for the duration of the tests, and on the probing success. So if the bandwidth allows it, the system would choose a higher streaming rate after 8 RTCP (probing frequency + probing duration) reports, the equivalent of approx 40s. If frequent feedback is used, for example 1 RTCP RR each second, the reaction time would be reduced to 8s. Probing success percentage is given in Table II. We can observe that even in the ideal case when the bandwidth is almost twice as high (195%) as the current rate, we have an up-switch success of only 71%. This means that in 30% of the cases the quality should have been increased by the server, but it was not. The reason is that the video rate is not perfectly constant and it might happen that during probing the actual bandwidth limit may be smaller and therefore higher deviation may result. In the case of 150% and 165% the success rate is 50% which can be considered as a false positive because we only want to increase the quality when available bandwidth is twice as high. This event is not a harmful one though because the network should be able to support the higher rate for a while and in a moving scenario the signal strength will continue to increase so no harm was done in the end. The most important is the low percentage of up-switching in the worst case scenario (120% limit) when the server increases the video quality introducing congestion in the network.

Bandwidth limit (QDisc)	120%	150%	165%	195%
Up-switch success rate	16%	49%	51%	71%

TABLE II.
UP-SWITCH SUCCESS RATE

Compared to the adaptive streaming solution proposed in [44], although we did not have access to their platform to test it in similar conditions, we can see that performance is similar when detecting a bandwidth drop (approx. 6s delay) but going back to the original quality takes longer in our solution. This may be influenced by the frequency of the RTCP reports, which is not specified in [44]. The adaptation algorithm is implemented to achieve a trade off between fast reactivity and stability and without imposing special restrictions to the media player. During down-switch, by design, the system reacts faster, because increased RTT reflects a problem and we want to avoid high delays and packet loss. The up-switch takes longer because frequent switching in video quality is not desired.

2) Tests in a WiMax Access Network: Because the platform is intended to be used with clients connected in a wireless environment, two test scenarios were prepared, first with the client connected in a WiMax access Network and second with the client connected to a WiFi router. The WiMax setup is presented in Figure 12.

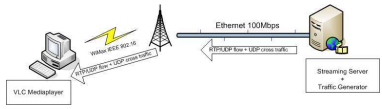

Figure 12. WiMax setup

For this test, where available 3 different versions of the content on the server: first encoded at 2.5Mbps, second at 1.7Mbps and the lowest quality encoded at 800Kbps. Since the capacity of our WiMax channel is about 6.5Mbps, we simulated a drop in the available bandwidth by sending additional UDP cross traffic over the air interface at a rate of 4.5Mbps. The duration of the bandwidth limitation is set to about 25s, similar to the throughput variations observed in [45] at driving speed.

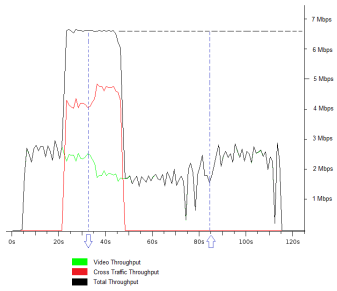

Figure 13. Throughput evolution during WiMax test

In Figure 13 when the cross traffic is sent (red line in the figure), the total throughput (the black line) reaches the maximum capacity of 6.5Mbps, which is not enough to send the whole 7Mbps of data (2.5Mbps video + 4.5Mbps cross traffic). The server decides to switch to the next lower encoding quality after approx 10s and will up-switch back in another 40s, after the bandwidth limitation has passed. If we compare the RTT evolution to the case

where no rate adaptation was used in Fig. 14, we can see that the maximum value of RTT is lower and also the congestion period is minimised. In this way interactivity speed is not affected suffered and also the packet loss rate is kept to 0 so the QoE was maximised. In Table III, we have the average of the RTT during the congestion period obtained from 4 different runs of the same experiment.

Experiment type	Average RTT during cross traffic
With Adaptation	270ms
Without Adaptation	650ms

TABLE III.
AVERAGE RTT DURING CONGESTION

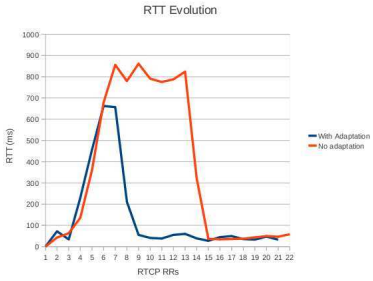

Figure 14. RTT evolution in the WiMax test

3) Tests in a WiFi Access Network: In the WiFi test, the client was connected to a WiFi 802.11g router configured in NAT mode with port forwarding enabled to allow UDP traffic. Although a multi hop experiment has not been considered in this paper, experiments performed in [46] show that delay evolution is similar to the one observed in single hop paths. The streaming session started near the access point, then the PC was moved away about 20m from the router loosing line-of-sight and then returned to the initial position. During this mobility test, the signal was not lost, but suffered degradation so the available bandwidth decreased with distance and increased back again when the client approached the WiFi router. In this case, the content versions available on the server were encoded at 380Kbps, 240Kbps and 180Kbps. Again, the experiment was ran once with the rate adaptation enabled and once without adaptation with the results shown in figures Fig. 15 and Fig. 16

We can see again that the RTT was kept low to improve the interactivity delay and that packet loss was also limited by the switch to a lower encoding, more suited to the network conditions. Compared to the WiMax and Qdisc tests, packet loss was more severe in the WiFi environment and it affected the image quality even if the available bandwidth was higher than the streaming rate.

In Table IV, we have the average packet loss during the whole streaming session obtained from 5 different runs of the same experiment.

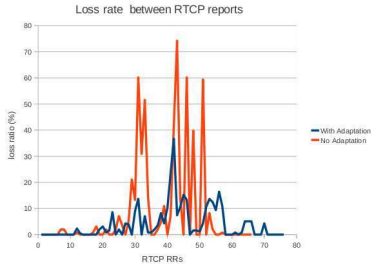

Figure 15. Loss rate during WiFi test

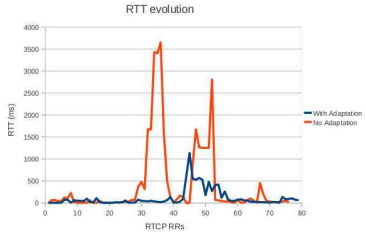

Figure 16. RTT evolution during WiFi test

Experiment type	Average packet loss
With Adaptation	3.6%
Without Adaptation	8.2%

TABLE IV.
AVERAGE LOSS RATE FOR THE WHOLE STREAMING SESSION

B. Cost of Compression Induced by Segmentation, and Switching Latency

The streaming abilities are implemented using the liveMedia library that has been extended to deliver H.264 files. Our tests have revealed that the fact that the video sequence is segmented in small clips, as described in Section II, does not penalize the fluency of the streaming playback. On the server side, although clips have to be pipelined dynamically in the transmission buffer, the processing load is not dramatically increased, and the correct rhythm of delivery of RTP packets is preserved even during the probing stage.

However, slight bitrate cost and some constraints are applied over the encoder H.264, in order to enable adaptive streaming and video content segmentation:

1) The overall compression speed is clearly damaged as the encoding process of every sequence is divided in the multiple clips and several alternative versions are provided. Nevertheless, the scenarios we consider are based on *on demand* video content. Hence, all the clips are preprocessed and included in the video database *in advance*, and because of this, the performance is not damaged.

2) Every new clip has to start with a new Instantaneous Decoding Refresh (*IDR*) frame, penalizing the encoding flexibility. Therefore, the segmentation in multiple pieces

of every sequence constraints the maximum size of the *GOP* (Group of Pictures) to the size of the encoded clips. Moreover, bitrate overhead is resulting from the use of *IDR refresh* frames. For this reason, a trade-off between the time of the system's response to the user's feedback, and the size of the clip has to be achieved, as every clip has to be completely delivered before starting to send the new one (due to the constraint of switching between versions in a temporally consistent way). If the clips are short, the system switches the playback very fast independently of the instant when the user's request is received. However, the penalty in terms of bitrate increases when the clip size decreases (*GOP* is also small increasing the bitrate). The opposite result occurs if the clips are longer. In our simulations we used sequences encoded at 25 *fps* and clip segmentation approximately every 15 frames. On the one hand, using 1 *GOP* per clip, a *GOP* of 15 frames is good enough in order not to penalize the global bitrate. The global loss in quality in PSNR in the luminance and chroma is less than 0.5 dBs respect to encoding the same sequence without the *GOP* constraint across several bitrate configurations (as depicted in Figure 17). On the other hand, the maximum latency in the server due to the clip segmentation is less than 700 milliseconds, as in the worst of the cases, the server has just sent the first frame of a new clip when receiving the request to switch the content. This delay is a good approach as depending on the Round Trip Time (*RTT*) of the wireless network and the pre-roll buffer of the player, the minimal delay is already in the order of 2 seconds. This cost is also low when measuring the quality loss with other techniques such as Structural similarity (SSIM). In this case, when handling very low bitrates (150-600 kbps) the loss can drop until 0,002 meanwhile for higher bitrates (1200-2000 kbps) this difference is lower than 0,0005.

3) Finally, it is also important to consider the increment of bitrate due to the *SPS* and *PPS* headers that are used in every new video clip. In the case that all the video sequence is encoded once, they have to be sent to the client just one time at the beginning. This is not the case when the sequence is split in several clips as in our framework. In Table V we include the increment of bitrate for different video resolutions at different levels of quality (by modifying the quantization parameter: *QP*). As we can observe, the cost of the headers is very low and almost negligible for higher quality encoding parameters (*QP*=16). The size of the header is almost constant in every case, independently of the encoding parameters that are being used. Hence, when the quality of the image is increased at the cost of spending bitrate, the related cost of the headers gets lower and lower. The video segmentation occurs again approximately every 15 frames.

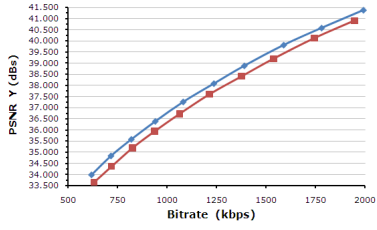

Figure 17. Video quality comparison in the luminance component when applying or not the GOP constraint. Red line represents a sequence encoded with GOP 15, while the blue line depicts the same sequence encoded without GOP restrictions. The Bitrate is computed for different QPs.

Sequence dimensions	**Quantization Parameter**	**Bitrate increment** (%)
176x144	16	0,86
176x144	32	5,95
352x288	16	0,68
352x288	32	5,73
720x576	16	0,76
720x576	32	3,84

TABLE V.
INCREMENT OF BITRATE USING VIDEO SEGMENTATION DUE TO THE REQUIRED SPS AND PPS HEADERS TO SYNCHRONIZE THE DECODER

The global interaction delay has also been measured through several tests (100 samples per case). This delay is considered as the difference between the time the user presses de request button and the new content starts to be displayed in the player. Hence, it sums up the time needed to forward the client request to the server, the time elapsed before the server gets the opportunity to switch between clips (this is proportional to the clip duration), and the buffer size (we assume no buffer flushing). As shown in Table VI the global delay depends on the probing strategy, and is decreased thanks to the proposed adaptive streaming strategy. Pausing the delivery of content before a new probing attempt increases the margin of time the server has to switch to another clip, due to a client request. Obviously, during the pause, one should take care not to empty the pre-roll buffer of the client, which is regulated from the beginning of the video transmission. In contrast, if the probing is implemented by increasing the delivery rate before pausing the system, the interaction delay is increased compared to a system without probing (see last line of Table VI).

Experiment type	Average delay
Without Adaptation	2.28 s
With our model	2.18 s
With burst before pause	2.44 s

TABLE VI.
AVERAGE GLOBAL DELAY TO THE USER REQUEST.

C. Validation of the Interactive Features Through Subjective Tests

Our platform was tested through questionnaires answered by 20 different people. Te viewers were asked to exploit the interactive features of our system in different video sequences related to sport content and video surveillance respectively. The soccer demo contained 10 minutes of a match with some highlights of a match. The

video surveillance sequence, of similar duration, consisted on scenes in open air parking where different people and vehicles pass by. From the results of the experiments, we depict the satisfaction of the viewers with our browsing capabilities and the way they handle them when they get used to the platform. The latest was approached by a second round of demos after filling the questionnaire.

In soccer, the three browsing features were valued by at least 90% of the viewers as very or quite profitable (5 or 4 out of 5 in our score ranking). The interaction strategy was also generally approved. Some users might still prefer the non zoomed-in version proposed by default for far view shots or the resumption of a segment after a *replay on demand* from its beginning. The transparent switching from fast forward to regular mode at the beginning of a highlight was well appreciated for all. The favourite feature was the replay (65% of the users) while zoom-in(out) was the most used according to our records of. The main complaint of the users was that the zooming factor, although well centered over the RoI, had only one level and should be more aggressive to be distinguished from the original version. Nevertheless, this issue is associated to computer vision algorithms and not to the proposed practical functionalities.

In video surveillance, all the users considered very or quite profitable the capacity of selecting single RoIs from the general view and focusing over them. Also the users do not perceive any loss of quality when dealing with HD sequences where the view is split in 4 different cameras and they can focus the one with an available RoI. The round-trip strategy is clear for all but 80% consider it not practical when dealing with many RoIs at the same time due to the limitations of the GUI. Most of the viewers also appreciate the video contents based on focusing over two or more RoIs (85%) and the original view alternative in which the detected RoI is compressed with higher quality than the background (95%).

In global, all the testers considered the video streaming fluent enough compared to other standard streaming servers. No one could notice any issue devoted to the change of rate delivery due to the bitrate adaptation algorithm as the video rate did not change or got anyhow stuck. Temporally consistency was also generally approved and well appreciated. 40% of the users still noticed some small video artefacts in some occasions after pressing a button for an request. This factor just related to the video player performance was still not considered as damaging (not ranked in any case more than 3 out of 5 in our scale). The interaction delay was considered a very important factor for 85% of the viewers and particularly critical in video surveillance. Finally, 70% of the users considered that the video player interface could be slightly improved. Although considered simple and handy, for a 55% of the users the GUI should be more intuitive. More buttons or plots over the video should then be used for a more direct, easier and clearer navigation over the different content alternatives.

VI. Conclusion

In this paper, we described a flexible interactive streaming system, over one underlying key mechanism: temporal content pipelining, which allows to switch the video content at whichever point of the playback in a temporally consistent way. This mechanism uses the client's feedback, requiring only one open streaming session per user and no advanced implementation mechanisms. Furthermore, we implement a streaming quality adaptation algorithm based on the RTCP feedback received from the client. In this algorithm, rather than just focusing on its general purpose, a novel probing technique embedded to decrease the interaction delay of our interactive system. Experimental results validate our adaptive rate algorithm and show that the video segmentation does not have any effect in the fluency of the streaming playback and in addition, the bitrate is not significantly increased. Therefore, the browsing features do not damage the global performance of the system. We also present three different switching capabilities when streaming video soccer content: zooming over RoIs, fast forward and additional replays selection. All together, subjectively increases the perceived quality of the streaming experience. The profits of our architecture mainly rely on supporting personalized content selection according to the interaction with the viewer and the automatic video quality adaptation. Finally, our framework is also able to include, for example, targeted advertising just by implementing the concept of client profile. In addition to the interactive selection of versioned video segments, the architecture is also designed to allow the insertion of promotional or any other kind of content in the middle of the main streaming playback. Later, the playback can be resumed directly without any kind of offset, interruption or efficiency cost. Hence, our interactive architecture can be extended to offer support to multiple streaming applications. In this paper we focus on adapting broadcasting TV soccer and video surveillance content for smart phone terminals and wireless networks.

Appendix

A. Definition of up-switch thresholds

Let T_0 be the outage probability that the available bandwidth B is greater or equal to twice the bit rate of the video sequence R. We would then look for the deviation threshold d_0 such that

$$P\left[B \geq 2\,R | dev \leq d_0\right] \geq T_0 \tag{5}$$

Conversely, one could set the deviation threshold d_0 and compute the outage probability T_0.

In the `qdisc` set-up, we have measured the deviation in $i = 6$ different scenarii ($B_i \in \{1.2, 1.35, 1.5, 1.7, 1.85, 2\}\,R$). We therefore know

$$P\left[dev \leq d_0 | B = B_i\right] = \int_{-\infty}^{d_0} T_{dev|B=B_i}\left(dev\right) ddev \tag{6}$$

$$= cdf_{dev|B=B_i}\left(d_0\right) \tag{7}$$

We can regard those six scenarii as a sampling of the frequency domain, so as to write the average cdf as

$$\begin{aligned} & P\left[dev \leq d_0\right] \\ = \; & P\left[dev \leq d_0 | B = B_1\right] P\left[B < B_{1,2}\right] \\ & + \sum_{i=2}^{5} P\left[dev \leq d_0 | B = B_i\right] \\ & \quad P\left[B_{(i-1),i} \leq B < B_{i,(i+1)}\right] \\ & + P\left[dev \leq d_0 | B = B_6\right] P\left[B \geq B_{5,6}\right] \end{aligned} \quad (8)$$

where

$$B_{i,j} = \frac{B_i + B_j}{2} \quad (9)$$

Returning to (5), we can write

$$P\left[B \geq 2\,R | dev \leq d_0\right] = 1 - P\left[B < 2\,R | dev \leq d_0\right] \quad (10)$$

This last conditional probability can be transformed thanks to the Bayes formula into

$$\begin{aligned} & P\left[B < 2\,R | dev \leq d_0\right] \\ = \; & \frac{P\left[B < 2\,R\right]\; P\left[dev \leq d_0 | B < 2\,R\right]}{P\left[dev \leq d_0\right]} \end{aligned} \quad (11)$$

In (11), $P\left[B < 2\,R\right]$ depends on the wireless set-up under consideration. Extrapolating from downstream UDP throughput from [47], one could possibly model the available bandwidth from UDP streaming as an exponential distribution parametrised to C, the nominal capacity of the wireless set-up, such that

$$P\left[B < 2\,R\right] = 1 - \exp\left[-\left(\frac{3}{C}\right) 2\,R\right] \quad (12)$$

Based on relations (6-8) and on the bandwidth model (12), we would get

$$\begin{aligned} & P\left[dev \leq d_0 | B < 2\,R\right] \\ = \; & P\left[dev \leq d_0 | B = B_1\right] P\left[B < B_{1,2}\right] \\ & + \sum_{i=2}^{5} P\left[dev \leq d_0 | B = B_i\right] \\ & \quad P\left[B_{(i-1),i} \leq B < B_{i,(i-1)}\right] \\ & + P\left[dev \leq d_0 | B = B_6\right] P\left[B_{5,6} \leq B < 2\,R\right] \end{aligned} \quad (13)$$

$$\begin{aligned} = \; & \left\{1 - \exp\left[-\left(\tfrac{3}{C}\right) 1.275\,R\right]\right\} cdf_{dev|B=B_1}\left(d_0\right) \\ & + \left\{ \begin{array}{l} \exp\left[-\left(\tfrac{3}{C}\right) 1.275\,R\right] \\ -\exp\left[-\left(\tfrac{3}{C}\right) 1.425\,R\right] \end{array} \right\} cdf_{dev|B=B_2}\left(d_0\right) \\ & + \left\{ \begin{array}{l} \exp\left[-\left(\tfrac{3}{C}\right) 1.425\,R\right] \\ -\exp\left[-\left(\tfrac{3}{C}\right) 1.6\,R\right] \end{array} \right\} cdf_{dev|B=B_3}\left(d_0\right) \\ & + \left\{ \begin{array}{l} \exp\left[-\left(\tfrac{3}{C}\right) 1.6\,R\right] \\ -\exp\left[-\left(\tfrac{3}{C}\right) 1.775\,R\right] \end{array} \right\} cdf_{dev|B=B_4}\left(d_0\right) \\ & + \left\{ \begin{array}{l} \exp\left[-\left(\tfrac{3}{C}\right) 1.775\,R\right] \\ -\exp\left[-\left(\tfrac{3}{C}\right) 1.925\,R\right] \end{array} \right\} cdf_{dev|B=B_5}\left(d_0\right) \\ & + \left\{ \begin{array}{l} \exp\left[-\left(\tfrac{3}{C}\right) 1.925\,R\right] \\ -\exp\left[-\left(\tfrac{3}{C}\right) 2\,R\right] \end{array} \right\} cdf_{dev|B=B_6}\left(d_0\right) \end{aligned} \quad (14)$$

$$\begin{aligned} & P\left[dev \leq d_0\right] \\ = \; & P\left[dev \leq d_0 | B < 2\,R\right] \\ & + cdf_{dev|B=B_6}\left(d_0\right) P\left[B \geq 2\,R\right] \end{aligned} \quad (15)$$

$$\begin{aligned} = \; & P\left[dev \leq d_0 | B < 2\,R\right] \\ & + \exp\left[-\left(\frac{3}{C}\right) 2\,R\right] cdf_{dev|B=B_6} \end{aligned} \quad (16)$$

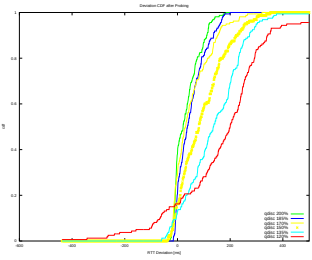

Figure 18. CDFs for the six scenarios

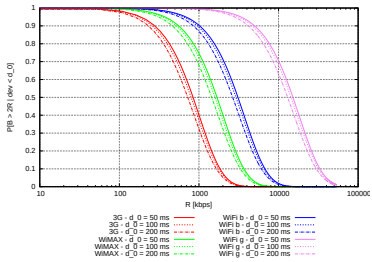

Figure 19. Probability that there is enough bandwidth to upswitch s.t. observed deviation. C is worth respectively 3 Mbps (3G), 6 Mbps (IEEE 802.16 - WiMAX), 11 Mbps (IEEE 802.11b - WiFi) and 54 Mbps (IEEE 802.11g - WiFi)

Looking at Fig. 18, we can measure Table VII:

Margin	50 ms	100ms	200ms
1.2	.2	.3	.5
1.35	.3	.4	.7
1.5	.4	.6	.9
1.7	.6	.75	.95
1.85	.6	.8	1
2	.7	.85	1

TABLE VII.
DEVIATION SAMPLES FROM FIG. 18

Injecting values of Table VII into relations (14) and (16), we can plot the probability (10) in Fig. 19. For a deviation $d_0 = 50$ ms, an upswitch has a 90% success rate provided the streaming rate R is lower than 300 kbps in 3G networks, 500 kbps in WiMAX scenario, 1 Mbps in WiFi b and 5 Mbps in WiFi g. Considering a sequence at 1 Mbps streamed on a 3G network, the upswitch has a 30% success rate if the observed deviation is up to 200 ms, whereas this rate increases to 40% if the deviation is as low as 50 ms.

REFERENCES

[1] A. Argyriou and V. Madisetti, "Streaming h.264/avc video over the internet," in *Consumer Communications and Networking Conference*, Jan. 2004, pp. 169–174.

[2] M. F. Sayit and T. Tunah, "Video streaming with h.264 over the internet," in *Signal Processing and Communications Applications*, Apr. 2006, pp. 1–4.

[3] Z. Li and Z. Zhang, "Real-time streaming and robust streaming h.264/avc video," in *Third International Conference on Image and Graphics*, Dec. 2004, pp. 353–356.

[4] J.Lu, G.Lafruit, and F.Catthoor, "Fast reliable multi-scale motion region detection in video processing," in *ICASSP*, vol. 1, April 2007, pp. 689–692.

[5] P.Baccichet, X.Zhu, and B.Girod, "Network-aware h.264/avc region-of-interest coding for a multi-camera wireless surveillance network," in *Picture Coding Symp.*, April 2006.

[6] T. Bae, T. C.Thang, D.Y.Kim, Y.M.Ro, J.W.Kang, and J.G.Kim, "Multiple region-of-interest support in scalable video coding," in *ETRI Journal*, vol. 28, April 2006, pp. 239–242.

[7] C. DeVleeschouwer, T. Nilsson, K. Denolf, and J. Bormans, "Algorithmic and architectural co-design of a motion-estimation engine for low power video devices," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, Dec. 2002.

[8] R. Sutter, K.DeWolf, S.Lerouge, and R.VandeWalle, "Lightweight object tracking in compressed video streams demonstrated in region-of-interest coding," *EURASIP*, pp. 59–75, January 2007.

[9] A. Senior, A. Hampapur, Y.-L. Tian, L. Brown, S. Pankanti, and R. Bolle, "Appearance models for occlusion handling," *Image and Vision Computing*, vol. 24, no. 11, pp. 1233–1243, November 2006.

[10] J. You, G. Liu, and L. Sun, "A multiple visual models based perceptive framework for multilevel video summarization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 3, pp. 273–285, March 2007.

[11] A.Cavallaro, O.Steiger, and T.Ebrahimi, "Semantic video analysis for adaptive content delivery and automatic description," in *IEEE Tran. on CSVT*, vol. 15, October 2005, pp. 1200–1209.

[12] A. G. Money and H. Agius, "Video summarization: a conceptual framework and survey of the state of the art," *Journal of Visual Communication and Image Representation*, vol. 19, pp. 121–143, 2008.

[13] B. T. Truong and S. Venkatesh, "Video abstraction: A systematic review and classification," in *ACM Transactions on Multmedia Computing, Communication and Applicationl*, vol. 3, 2007.

[14] A. Mavlankar and B. Girod, "Spatial-random-access-enabled video coding for interactive virtual pan/tilt/zoom functionality," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 5, pp. 577–588, 2011.

[15] I. A. Fernandez, F.Chen, F.Lavigne, X.Desurmont, and C. DeVleeschouwer, "Browsing sport content through an interactive h.264 streaming session," in *MMEDIA*, Athens, June 2010.

[16] I. A. Fernandez, F.Lavigne, X.Desurmont, and C. DeVleeschouwer, "Worthy visual content on mobile through interactive video streaming," in *ICME:2010 IEEE International Conference on Multimedia and Expo*, Singapore, July 2010.

[17] W. You, M. S. H. Sabirina, and M. Kima, "Real-time detection and tracking of multiple objects with partial decoding in h.264/avc bitstream domain," in *SPIE*, vol. 7244, Feb. 2009.

[18] W.Wang, J.Yang, and W.Gao, "Modeling background and segmenting moving objects from compressed video," *IEEE transactions on circuits and systems for video technology*, vol. 18, pp. 670–681, May 2008.

[19] A. Gyaourova, C. Kamath, and S.-C. Cheung, "Block matching for object tracking," *University of California Radiation Laboratory-TR*, vol. 200271, October 2003.

[20] D.J.Thirde, M.Borg, V.Valentin, L. Barthelemy, J.Aguilera, G.Fernandez, J.M.Ferryman, F.Bremond, M.Thonnat, and M.Kampel, "People and vehicle tracking for visual surveillance," in *VS 06: Proceedings of the IEEE International Workshop on Visual Surveillance*. ACM, Mai. 2006.

[21] X. Yu and D. Farin, "Current and emerging topics in sports video processing," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2005.

[22] Y. Takahashi, N. Nitta, and N. Babaguchi, "Video summarization for large sports video archives," *Multimedia and Expo, IEEE International Conference on*, vol. 0, pp. 1170–1173, 2005.

[23] L. Sun and G. Liu, "Field lines and players detection and recognition in soccer video," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASPP)*, 2009, pp. 1237–1240.

[24] D. Tjondronegoro, Y. Chen, and B. Pham, "Highlights for more complete sports video summarization," *IEEE Transactions on Multimedia*, vol. 11, pp. 22–37, 2004.

[25] ITU-T, "H.264: Advanced video coding for generic audiovisual services," *Series H : Audiovisual and multimedia systems*, 2005.

[26] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (rtsp)," RFC 2326 (Proposed Standard), Apr. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2326.txt

[27] A. Z. Sarker, "A study over adaptive real time video over lte," in *Ph.D. thesis*, Lulea University of Technology, 2007.

[28] T. Schierl and T. Wiegand, "H.264/avc rate daptation for internet streaming," in *14th International Packet Video Workshop (PV)*, Irvine, CA, USA, December 2004.

[29] I. RealNetworks, "Helix mobile server rate control for mobile networks," 2008.

[30] H. Schulzrinn, S. Casner, R. Frederick, , and V. Jacobson, "Rtp: A transport protocol for real-time applications," in *Tech. Rep., IETF RFC 3550*, July 2003.

[31] V. Paxon and M. Allman, "Computing tcps retransmission timer," in *Tech. Rep., IETF RFC 2988*, November 2000.

[32] M. P. Farrera, M. Fleury, K. Guild, and M. Ghanbari, "Measurement and analysis study of congestion detection for internet video streaming," in *Journal of Communications, vol. 5, no. 2, pp. 169–177*, February 2010.

[33] J. Fabini, W. Karner, L. Wallentin, and T. Baumgartner, "The Illusion of Being Deterministic -Application-Level Considerations on Delay in 3G HSPA Networks," in *NETWORKING '09: 8th International IFIP-TC 6 Networking Conference*. Berlin, Heidelberg: Springer-Verlag., 2009, pp. 301–312.

[34] P. R. Maierhofer, F. Ricciato, A. D'Alconzo, R. Franzan, and W. Karner, "Network-wide measurements of tcp rtt in 3g," in *TMA 09: First International Workshop on Traffic Monitoring and Analysis*. Berlin, Heidelberg: Springer-Verlag., 2009, pp. 17–25.

[35] K. Jang, M. Han, S. Cho, H.-K. Ryu, J. Lee, Y. Lee, and S. Moon, "3g and 3.5g wireless network performance measured from moving cars and high-speed trains," in *MICNET '09 Proceedings of the 1st ACM workshop on Mobile internet through cellular networks*, Beijing, China, 2009.

[36] W. Eklof, "Adapting video quality to radio links with different characteristics," in *Master of Science Thesis*, Sweden, 2008.

[37] J. Navratil and R. L.Cotrell, "Abwe: A practical approach to available bandwidth estimation," in *Stanford Linear Accelerator Center (SLAC)*, 2003.

[38] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cotrell, "pathchirp: Efficient available bandwidth estimation for network paths," in *Passive and Active Measurement Workshop*, 2003.

[39] M. Li, M. Claypool, and R. Kinicki, "Wbest: a bandwidth estimation tool for ieee 802.11 wireless networks," in *In Proceedings of 33rd IEEE Conference on Local Computer Networks (LCN)*, Montreal, Quebec, Canada, October 2008.

[40] I. A. Fernandez, P.R.Alface, T.Gan, R.Lauwereins, and C. DeVleeschouwer, "Integrated h.264 region-of-interest detection, tracking and compression for surveillance scenes," in *PV 10: 18th International Packet Video Workshop*, Hong Kong, December 2010.

[41] X. Desurmont, A. Bastide, J. Czyz, C. Parisot, J.-F. Delaigle, and B. Macq, "A general purpose system for distributed surveillance and communication," in *Intelligent Distributed Video Surveillance Systems*. S.A Velastin and P Remagnino Eds, 2006.

[42] F. Chen and C. D. Vleeschouwer, "A resource allocation framework for summarizing team sport videos," in *IEEE International Conference on Image processing(ICIP), Cairo, Egipt*, 2009.

[43] J. Li, T. Wang, W. Hu, M. Sun, and Y. Zhang, "Soccer highlight detection using two-dependence bayesian network," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2006.

[44] T.Schierl, T. Wiegand, and M. Kampmann, "3GPP Compliant Adaptive Wireless Video Streaming Using H.264/AVC," in *IEEE International Conference on Image Processing. ICIP 2005.* .

[45] R. Gass and C. Diot, "An experimental performance comparison of 3g and wi-fi," in *11th Passive and Active Measurement Conference (PAM 2010)*, Zurich, 2010.

[46] G. Gommand, L.Schumacher, and G. Toma, "Performance Evaluation of Indoor Internet Access over a Test LTE Mini-Network," in *The 14th International Symposium on Wireless Personal Multimedia Communications, WPMC'11* , October 2011.

[47] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3g using wifi: Measurement, system design and implementation," in *MobiSys 2010*, San Francisco, USA, 2010.

Ivan Alen Fernandez received his M.Sc. Telecommunications Engineering degree from the University of Vigo (Spain), in 2009.

As a part of his studies, he developed his Master Thesis on video coding (H.264) and motion detection & tracking in the multimedia group of the research institute IMEC (Belgium) in 2008-2009. He was research assistant at the Université Catholique de Louvain (UCL), between 2009 and 2011. His main interest topics include video networking, security and cryptography. He has also worked at Vodafone Spain as trainee in 2008 in the Radio Networks Department and currently he is Quality Control Manager in critical software development (Daintel) for ICUs in Denmark.

Christophe De Vleeschouwer received the Electrical Engineering degree and the Ph. D. degree from the Université Catholique de Louvain (UCL) Louvain-la-Neuve, Belgium in 1995 and 1999 respectively.

He is currently a permanent Research Associate of the Belgian NSF and an assistant professor at UCL. He was a senior research engineer with the IMEC Multimedia Information Compression Systems group (1999-2000), and contributed to project with ERICSSON. He was also a post-doctoral Research Fellow at UC Berkeley (2001-2002), and at EPFL (2004). His main interests concern video and image processing for communication and networking applications, including content management and security issues. He is also enthusiastic about non-linear signal expansion techniques, and their use for signal analysis and signal interpretation. He is the co-author of more than 20 journal papers or book chapters. He did coordinate the FP7-216023 APIDIS European project (www.apidis.org), and several Walloon region projects, respectively dedicated to video analysis for autonomous content production, and to personalized and interactive mobile video streaming.

George Toma received his engineer degree at the Polytechnic University of Bucharest, Romania in automatic control, in 2007.

He is currently a PhD student at FUNDP - The University of Namur, Belgium. He was a research assistant at the same university between 2007 and 2010. His research topics include adaptive streaming techniques, quality assessment of streaming sessions and performance evaluation of wireless access networks.

Laurent Schumacher received his M.Sc. EE. Degree from the Faculté Polytechnique de Mons, Belgium, in 1993 and his Ph.D. degree from the Université catholique de Louvain, Belgium, in 1999. Since 2003, he has been a professor at FUNDP - The University of Namur, Belgium, after a post-doctoral stay at Aalborg Universitet, Denmark. His current research interests include performance evaluation of cellular systems (LTE and beyond) and SIP/IMS signalling. Prof. Schumacher is a member of the IEEE Computer Society and the ACM.

Bibliography

[1] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, July 2003.

[2] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network, "Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs (Release 6)," in *Tech. Rep. 3GPP TS 26.234 v6.9.0*, October 2006.

[3] Alan Clark and Geoff Hunt, "RTCP XR Report Block for QoE Metrics Reporting," ''http://tools.ietf.org/id/draft-ietf-avt-rtcp-xr-qoe-00.txt'', Expired April 2008.

[4] Ronny Henkes, "Adaptation automatique de la qualité d'un streaming vidéo par les protocoles RTP/RTCP/RTSP," M.Sc. Thesis, FUNDP Belgium, ''http://www.fundp.ac.be/recherche/publications/page_view/70394/'', 2010.

[5] Eklof William, "Adapting Video Quality to Radio Links with Different Characteristics," M.Sc. Thesis, KTH Sweden, ''http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/081208-William_Ekloef-with-cover.pdf'', 2008.

[6] T. Schierl and T. Wiegand, "H.264/AVC Rate Adaptation for Internet Streaming," in *14th International Packet Video Workshop (PV)*, Irvine, CA, USA, December 2004.

[7] N.Baldo, U. Horn, M.Kampmann, and F. Hartung, "RTCP Feedback Based Transmission Rate Control for 3G Wireless Multimedia Streaming," in *The 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications(PIMRC 2004)*, September 2004.

[8] International Telecommunication Union, "Telephone Network and ISDN, Quality of Service, Network Management and Traffic Engineering," ITU-T Recommendation E.800, 1994.

[9] Wikipedia.org [online], "Quality of Service," ''http://en.wikipedia.org/wiki/Quality_of_service'', last visit: May 2012.

[10] Cisco DocWiki [online], "Internetworking Technology Handbook," ''http://docwiki.cisco.com/wiki/Quality_of_Service_Networking'', last visit: May 2012.

[11] Wikipedia.org [online], "Throughput," ''http://en.wikipedia.org/wiki/Throughput'', last visit: May 2012.

[12] Geoff Huston, *Internet Performance Survival Guide - QoS Strategies for Multiservice Networks*, John Wiley and Sons, Inc, 2000.

[13] G. Almes, S. Kalidindi, and M. Zekauskas, "A Round-trip Delay Metric for IPPM," RFC 2681, 1999.

[14] L.J. van Beek, P. Kerofsky, "Server-Side Playout Delay Management for Video Streaming ," in *ICIP2006, IEEE International Conference on Image Processing*, Atlanta, GA, United States, 2006, pp. 3077–3080, ACM.

[15] Hua-xia Rui, Chong-rong Li, and Sheng-ke Qiu, "Evaluation of Packet Loss Impairment on Streaming Video," *Journal of Zhejiang University - Science A*, vol. 7, pp. 131–136, 2006, 10.1631/jzus.2006.AS0131.

[16] DCCP Active WG, Ingemar Johansson, "DCCP Status Pages, IETF-73 DCCP minutes," [online] ''http://tools.ietf.org/wg/dccp/minutes?item=minutes73.html'', last visit: June 2011.

[17] V. Paxon and M. Allman, "Computing TCP's Retransmission Timer," RFC 2988, November 2000.

[18] V. Jacobson, "Congestion avoidance and control," in *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, New York, NY, USA, 1988, pp. 314–329, ACM.

[19] International Telecommunication Union, "Vocabulary for Performance and Quality of Service"," ITU-T Recommendation P.10/G.100 Amendment 1, 2007.

[20] TeliaSonera International Carrier, "Content Delivery Networks: Ensuring Quality of Experience in Streaming Media Applications," ''www.teliasoneraic.com/NewsandEvents/Whitepapers/CDN/index.htm'', last visit: August 2011.

[21] Yun Q. Shi and Huifang Sun, *Image and Video Compression for Multimedia Engineering, Fundamentals, Algorithms and Standards*, CRC Press, 2000.

[22] Institute for Telecommunication Sciences - ITS, "Why Develop New Metrics for Digital Video Systems?," ''http://www.its.bldrdoc.gov/n3/video/new_metrics/'', last visit: November 2011.

[23] International Telecommunication Union, "Methodology for the subjective assessment of the quality of television pictures," ITU-T Recommendation ITU-R BT.500-11, 2002.

[24] Francesca De Simone, Matteo Naccari, Marco Tagliasacchi, Frederic Dufaux, Stefano Tubaro, and Touradj Ebrahimi, "Subjective Quality Assessment of H.264/AVC Video Streaming with Packet Losses," *EURASIP Journal on Image and Video Processing*, 2011.

[25] Stefan Winkler and Frederic Dufaux, "Video Quality Evaluation for Mobile Applications," Proc. SPIE 5150, 593, 2003.

[26] Van Peteghem H. and Schumacher L., "Description of an IPv6 Linux-based UTRAN Testbed," in *2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom)*, Barcelona, Spain, 2006.

[27] MSU Graphics and Media Lab (Video Group) [online], "MSU Video Quality Measurement Tool," ''http://compression.ru/video/quality_measure/info_en.html'', last visit: December 2011.

[28] Kye-Hwan Lee, Son Tran Trong, Bong-Gyun Lee, and Young-Tak, "QoS-Guaranteed IPTV Service Provisioning in IEEE 802.11e WLAN-based Home Network," in *Network Operations and Management Symposium Workshops (NOMS)*, 2008.

[29] Quan Huynh-Thu and Mohammed Ghanbari, "Impact of Jitter and Jerkiness on Perceived Video Quality," in *Proceedings of the Second International Workshop on Video Processing and Quality Metrics (VPQM06) for Consumer Electronics* , 2006.

[30] M. Ries, O. Nemethova, and M. Rupp, "Motion Based Reference-Free Quality Estimation for H.264/AVC Video Streaming ," in *2nd International Symposium on Wireless Pervasive Computing, ISWPC '07*, 2007.

[31] M. Ries, O. Nemethova, and M. Rupp, "Video Quality Estimation for Mobile H.264/AVCVideo Streaming," *Journal of Communications*, vol. 3, January 2008.

[32] Ricardo Rafael Pastrana-Vidal, "Vers une Métrique Perceptuelle de Qualité Audiovisuelle dans un Contexte à Service Non Garanti," *Ph.D. thesis, Université de Bourgogne*, 2005, ''http://books.google.be/books?id=osp6OwAACAAJ''.

[33] Pastrana-Vidal R. and Gicquel J., "Automatic Quality Assessment of Video Fluidity Impairments Using a No-Reference Metric," in *Proceedings of the Second International Workshop on Video Processing and Quality Metrics for Consumer Electronics WVPQM*, 2006, ''http://enpub.fulton.asu.edu/resp/vpqm2006/papers06/311.pdf''.

[34] Pastrana-Vidal R. and Gicquel J., "A No-Reference Video Quality Metric Based on a Human Assessment Model," in *Proceedings of the Third International Workshop on Video Processing and Quality Metrics for Consumer Electronics WVPQM2007*, 2007, ''http://enpub.fulton.asu.edu/resp/vpqm2007/papers/403.pdf''.

[35] Yeol-Kook Yoo Xingang Liu, "Real-Time Reference-Free Video Quality Measurement for Multimedia Communication," in *International Conference on Embedded Software and Systems*, 2008.

[36] T.Friedman, R. Caceres, and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)," RFC 3611, November 2003.

[37] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," RFC 2326, April 1998.

[38] Alan Clark and Geoff Hunt, "RTCP XR Report Block for Measurement Identity," ''http://tools.ietf.org/html/draft-ietf-avt-rtcp-xr-meas-identity-02'', November 2009.

[39] Mohammed Ebrahim Al-Mualla, C. Nishan Canagarajah, and David R. Bull, "Video Coding for Mobile Communications: Efficiency, Complexity, and Resilience," Academic Press, ISBN 0-12-053079-1, 2002.

[40] P. Frossard O. Verscheure and M. Hamdi, "MPEG-2 video Services over Packet Networks: Joint Effect of Encoding Rate and Data Loss on User-Oriented QoS," in *8th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV'98, United Kingdom*, 1998.

[41] Y. Li, A. Markopoulou, J. Apostolopoulos, and N. Bambos, "Content aware playout and packet scheduling for video streaming over wireless links," *IEEE Trans. Multimedia*, vol. 10, pp. 885–895, August 2008.

[42] M. Karczewicz and R. Kurceren, "The SP- and SI-frames design for H.264/AVC," in *IEEE Trans. on Circuits and Systems for Video Technology,*, July 2003, vol. 13, pp. 637–644.

[43] Alex Zambelli, "IIS Smooth Streaming Technical Overview," `''http://download.microsoft.com/download/4/2/4/4247C3AA-7105-4764-A8F9-321CB6C765EB/IIS_Smooth_Streaming_Technical_Overview.pdf''`, 2009, "Microsoft Corporation".

[44] Huang A. Ma K.J, Man Li and Bartovš R., "Video Rate Adaptation in Mobile Devices via HTTP Progressive Download of Stitched Media Files," *Communications Letters, IEEE*, vol. 15, pp. 320–322, 2011.

[45] International Telecommunication Union, "Advanced Video Coding for Generic Audiovisual Services," Amendament 3: Scalable Video Codec, 2005, "ITU-T Rec.H.264 and ISO/IEC 14496-10 AVC, v3".

[46] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, pp. 1103–1120, Sep. 2007.

[47] Heiko Schwarz, Patrick Ndjiki-Nya, and Thomas Wiegand, "Coding Efficiency Improvement for SVC Broadcast in the Context of the Emerging DVB Standardization," in *17th European Signal Processing Conference (EUSIPCO 2009)*, 2009.

[48] Inc. RealNetworks, "Minimize Rebuffering and Maximize Media Encoding Rate Delivered: Helix Mobile Server Rate Control for Mobile Networks," 2010.

[49] D. Bansal and H. Balakrishnan, "Binomial Congestion Control Algorithms," in *IEEE INFOCOM*, April 2001.

[50] D-M. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," *Computer Networks and ISDN Systems*, vol. 17, pp. 1–14, 1989.

[51] Danny De Vleeschauwer and David Robinson, "TCP: From Data to Streaming Video," `http://www2.alcatel-lucent.com/blogs/techzine/2011/tcp-from-data-to-streaming-video/`, March last visit: May 2012, Alcatel-Lucent.

[52] Song Chong Seong-jun Bae, "TCP-friendly Wireless Multimedia Flow Control Using ECN Marking ," in *IEEE Global Telecommunications Conference, GLOBECOM'02*, 2002, GLOBECOM '02.

[53] H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, and F. Khafizov, "TCP over Second (2.5g) and Third (3g) Generation Wireless Networks," feb 2003, IETF RFC3481.

[54] Eckehard Steinbach Mark Kalman and Bernd Girod, "Adaptive Media Playout for Low-delay Video Streaming Over Error-prone Channels," *IEEE Transaction on Circuits and systems for video technology*, vol. 14, pp. 841–851, 2004.

[55] Jiri Navratil and R. Les Cottrell, "ABwE: A Practical Approach to Available Bandwidth Estimation," in *in Passive and Active Measurement (PAM) Workshop 2003 Proceedings, La Jolla*, 2003.

[56] Vinay Ribeiro, Rudolf Riedi, Richard Baraniuk, Jiri Navratil, and Les Cotrell, "pathChirp: Efficient Available Bandwidth Estimation for Network Paths," in *Passive and Active Measurement Workshop*, 2003.

[57] Mingzhe Li, Mark Claypool, and Robert Kinicki, "Wbest: a Bandwidth Estimation Tool for IEEE 802.11 Wireless Networks," in *Proceedings of 33rd IEEE Conference on Local Computer Networks (LCN)*, Montreal, Quebec, Canada, October 2008.

[58] C. Mairal and M. Agueh, "Smooth and Scalable Wireless JPEG 2000 Images and Video Streaming with Dynamic Bandwidth Estimation," in *Advances in Multimedia (MMEDIA)*, Athens, Greece, June 2010.

[59] Dimitrios Koutsonikolas and Y. Charlie Hu, "On the feasibility of bandwidth estimation in 1x EVDO networks," in *Proceedings of the 1st ACM workshop on Mobile internet through cellular networks*, New York, NY, USA, 2009, MICNET '09, pp. 31–36, ACM.

[60] A.N.M. Zaheduzzaman Sarker, "A Study over Adaptive Real Time Video over LTE," Ph.D Thesis, Luleå University of Technology, 2007.

[61] Silvia [online], "Adaptive HTTP streaming for open codecs," ''http://blog.gingertech.net/2010/10/09/adaptive-http-streaming-for-open-codecs/'', last visit: May 2012.

[62] Xinghua Sun, Piamrat Kandaraj, and Viho Cesar, "QoE-based dynamic resource allocation for multimedia traffic in IEEE 802.11 wireless networks ," in *2011 IEEE International Conference on Multimedia and Expo (ICME)*, July 2011.

[63] Zoran Despotovic Mohammed Shehada, Srisakul Thakolsri and Wolfgang Kellerer, "QoE-based Cross-Layer Optimization for Video Delivery in Long Term Evolution Mobile Networks," in *The 14th International Symposium on Wireless Personal Multimedia Communications (WPMC'11)* , October 2011.

[64] George Toma, Laurent Schumacher, and Christophe De Vleeschouwer, "Offering streaming rate adaptation to common media players," in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME), 2011*, July 2011.

[65] Marcos Paredes Farrera, Martin Fleury, Ken Guild, and Mohammed Ghanbari, "Measurement and Analysis Study of Congestion Detection for Internet Video Streaming," *Journal of Communications*, vol. 5, no. 2, pp. 169–177, February 2010.

[66] Joachim Fabini, Wolfgang Karner, Lukas Wallentin, and Thomas Baumgartner, "The Illusion of Being Deterministic -Application-Level Considerations on Delay in 3G HSPA Networks," in *Proceedings of the 8th International IFIP-TC 6 Networking Conference (NETWORKING '09)*, Berlin, Heidelberg, 2009, pp. 301–312, Springer-Verlag.

[67] Peter Romirer-Maierhofer, Fabio Ricciato, Alessandro D'Alconzo, Robert Franzan, and Wolfgang Karner, "Network-Wide Measurements of TCP RTT in 3G," in *Proceedings of the First International Workshop on Traffic Monitoring and Analysis (TMA '09)*, Berlin, Heidelberg, 2009, pp. 17–25, Springer-Verlag.

[68] Laurent Schumacher, Gille Gomand, and George Toma, "Performance Evaluation of Indoor Internet Access over a Test LTE Mini-

Network," in *The 14th International Symposium on Wireless Personal Multimedia Communications (WPMC'11)* , Ocotber 2011.

[69] Keon Jang, Mongnam Han, Soohyun Cho, Hyung-Keon Ryu, Jaewha Lee, Yeoungseok Lee, and Sue Moon, "3G and 3.5G Wireless Network Performance Measured from Moving Cars and High-Speed Trains," in *Proceedings of the 1st ACM workshop on Mobile Internet through Cellular Networks (MICNET '09)*, Beijing, China, 2009.

[70] Arun Venkataramani Aruna Balasubramanian, Ratul Mahajan, "Augmenting Mobile 3G Using WiFi: Measurement, System design and Implementation," in *MobiSys 2010*, San Francisco, USA, 2010.

[71] Ali C. Begen Saamer Akhshabi and Constantine Dovrolis, "An Experimental Evaluation of Rate-adaptation Algorithms in Adaptive Streaming Over HTTP," in *Proceedings of the second annual ACM conference on Multimedia systems, MMSys'11*, Febryuary 2011.

[72] Aditya Mavlankar, Piyush Agrawal, Derek Pang, Sherif Halawa, Ngai-Man Cheung, and Bernd Girod, "An Interactive Region-of-Interest Video Streaming System for Online Lecture Viewing," in *Proc. International Packet Video Workshop - PV2010, Hong Kong, China*, December 2010.

[73] I.A. Fernandez, F. Lavigne, X. Desurmont, and C. DeVleeschouwer, "Worthy Visual Content on Mobile Through Interactive Video Streaming," in *Proceedings of 2010 IEEE International Conference on Multimedia and Expo (ICME 2010)*, Singapore, July 2010.

[74] E. Bomcke and C. De Vleeschouwer, "An Interactive Video Streaming Architecture for H.264/AVC Compliant Players," in *IEEE International Conference on Multimedia and Expo(ICME), New-York, USA*, 2009.

[75] Richard Gass and Christophe Diot, "An Experimental Performance Comparison of 3G and Wi-Fi," in *Proceedings of 33rd IEEE Conference on Local Computer Networks (LCN)*, Beijing, China, 2009.

[76] Whiteaker Jon, Schneider Fabian, and Teixeira Renata, "Explaining packet delays under virtualization," *SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. 38–44, January 2011.

[77] S. Wenger, M.M. Hannuksela, T. Stockhammer, M. Westerlund, and D. Singer, "RTP Payload Format for H.264 Video," RFC 3984, Feb. 2005.

[78] Ivan Alen Fernandez, Christophe De Vleeschouwer, George Toma, and Laurent Schumacher, "An Interactive Video Streaming Architecture Featuring Bitrate Adaptation," *Journal of Communications*, vol. 7, 2012, No. 4.

[79] S. Casner, "Session Description Protocol (sdp) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth," RFC 3556, July 2003.