PEDECIBA Informática

Instituto de Computación – Facultad de Ingeniería

Universidad de la República

Montevideo, Uruguay

Tesis de Doctorado

en Informática

Diseño centrado en calidad para la difusión Peer-to-Peer de video en vivo

Pablo Rodríguez Bocca

Tutores: Héctor Cancela (Universidad de la República, UDELAR, Uruguay)
Gerardo Rubino (IRISA - INRIA)

Julio 2008

Diseño centrado en calidad para la difusión Peer-to-Peer de video en vivo

Rodríguez Bocca, Pablo

ISSN 0797-6410

Tesis de Doctorado en Informática

Reporte Técnico RT 08 -05

PEDECIBA

Instituto de Computación – Facultad de Ingeniería

Universidad de la República.

Montevideo, Uruguay, Julio de 2008

Remerciements

Je remercie Alberto PARDO, Professeur, Universidad de la República, Uruguay, qui me fait l'honneur de présider ce jury.

Je remercie Edmundo DE SOUZA E SILVA, Professeur, Federal University of Rio de Janeiro, COPPE/ Computer Science Department, et Fernando PAGANINI, Professeur, Universidad ORT, Uruguay, d'avoir bien voulu accepter la charge de rapporteur.

Je remercie Eduardo GRAMPIN, Professeur, Universidad de la República, Uruguay, Daniel KOFMAN, Professeur, ENST/INFRES/RHD, et César VIHO, Professeur, INRIA Rennes, d'avoir bien voulu juger ce travail.

Je remercie tous les membres du projet GOL!P2P qui m'ont fourni un appui indispensable pendant la réalisatino de cette thèse, et je remercie spécialement Héctor CANCELA, Professeur, Universidad de la República, Uruguay, et Gerardo RUBINO, Directeur de Recherche, INRIA Rennes, qui ont dirigé ma thèse.

Contents

Part I

INTRODUCTION

Chapter 1

Global presentation of the thesis

This chaper presents the motivation, goals and contributions of the thesis. Finally, the organization of the document is outlined.

1.1 Motivation and goals

Video delivery applications are growing nowadays at increasing speed, as a consequence of the opening of video content producers to new business models, the larger bandwidth availability of the access network (on the Internet, on cellular networks, on private IP networks, etc.) and the explosion in the development of new hardware capable of reproducing and receiving video streams.

For example, it has been observed that the volume of video on the Internet doubles every year, while the demand is increased by a factor of three[1]. In the Internet context, video broadcast services are vastly deployed using Content Delivery Network (CDN) infrastructures, where a set of servers (located at strategic points around the Internet) cooperates transparently to deliver content to end users. However, since bandwidth is the most expensive resource on the Internet, and video delivery is one of the services that most demand it, live video services are limited yet in availability and diversity.

A method becoming popular these days consists in using the often idle capacity of the clients to share the distribution of the video with the servers, through the present mature Peer to Peer (P2P) systems. This approach also helps in avoiding local network congestions, because the servers (other clients) can be extremely close of the final client. On the negative side, the main drawback is that the clients (called peers in this context) connect and disconnect with high frequencies, in an autonomous and completely asynchronous way. This leads to the main challenge in P2P design: how to offer the quality needed by the clients in a highly varying environment.

Peers disconnections could cause the loss of the information they were sending to someone else. Live video distribution is very sensitive to losses, because of its real–time constraints. Moreover, in order to decrease bandwidth consumption, the encoding process takes away some of the natural video redundancy, making the streams still more vulnerable to missing data.

[1]See `http://www.researchandmarkets.com`.

Obviously, these (and other) factors affect the video quality perceived by final users, but it is not so obvious *how much* they affect it.

Standard network design uses indirect metrics such as loss rates, delays, reliability, etc. in order to measure and control the perceived quality in the network. The main target of our work is a quality-centric design of a peer-to-peer system for live video streaming. Therefore, it is important to be able to assess this perceived quality accurately and in real–time. There are two main approaches for measuring video quality, subjective tests and objective metrics, none of them adapt well to our design needs. In brief, subjective assessments consist in using a panel of human beings rating a series of short video sequences according to their own personal view about quality. They are time-consuming and inappropriate for real-time measurement. Objective assessments stand for the use of algorithms and formulas that measure the quality in a automatic, quantitative and repeatable way. The problem is that they usually do not correlate well with perceived quality. Moreover, they need the original signal to be computed. Mitigating the disadvantages of both approaches, hybrid methods have been developed. Pseudo-Subjective Quality Assessment (PSQA) is a recently proposed hybrid methodology for evaluating automatically and accurately the perceived quality at the client side, and it is widely used in our work for design issues.

File-sharing P2P distribution (for instance, based on Bittorrent-like protocols) uses a series of incentives and handshakes to exchange pieces of files between peers. Overheaded file searches and transfers from a peer to others can cause bottlenecks or delays unsuitable for real–time video streaming. To deal with this problem, this thesis proposes a multi-source approach where the stream is decomposed into several redundant flows sent by different peers to each other, in a tree-based overlay topology, with a very low signaling cost.

1.2 Contributions

The main objective of this work is to show the feasibility of a quality-centric design for video delivery networks. We provide a global methodology that allows to do the design by addressing the ultimate target, the perceived quality. We apply these ideas to the design of a P2P networks for live video streaming.

Our contributions concern the following topics:

- Quality of Experience.
- Multi-source Distribution using a P2P Approach.
- Efficient Search in Video Libraries.
- Quality-driven Dynamic Control of Video Delivery Networks.

Next subsections briefly describe the thesis' results around each of these points. The related publications are listed at the end of this chapter.

1.2.1 Quality of Experience

Our first contribution concerns the problem of video quality assessment, as the main component of the Quality-of-Experience (QoE) in video delivery networks.

We present a state of the art in video quality assessment, and an in-depth study of the PSQA methodology. PSQA builds a mapping between certain quality-affecting factors, and the quality as perceived by final users. This relation is learnt via a statistical tool, a Random Neural Network (RNN). The final output is a function able to mimic, somehow, the way that an average human assesses the quality of a video stream.

We improve the PSQA methodology in different ways. First, we study the effects of distribution failures on the perceived video quality, in particular the video frame loss effect, instead of the impact of packet losses (studied in all previous works). Studying the impact of losses at the frame level, allows to be independent of the kind of distribution failure, and our results can be applied to networked transmission errors (i.e. congestion), or server failures (for instance, in our context, a peer disconnection). Moreover, we show that the packet loss process does not correlate well with quality in a general context, because of its dependency of the protocol used. Instead, our frame loss factors have general applicability.

We also study the influence of video's motion on quality. The effect of different motion activity metrics is analyzed. We obtain and validate a very simple representation of motion, which is integrated in our PSQA mapping function to improve its accuracy.

PSQA allows to quantify *how* the frame losses affect the perceived quality. For instance, we show that, in some conditions, the user prefers sequences where losses are concentrated rather than spread. We show that, in a particular MPEG-4 encoding, P frames have a higher impact on quality than I frames. The PSQA technology will help us in the rest of our work, as we explain next. It is used in the following publications: [bj-hk07][euro-fgi07][globecom07][icc08] [ipom07audit][ipom07msource][lagos07][lanc07][pmccs07][submitted08a][submitted08b].

1.2.2 Multi-source Distribution using a P2P Approach

The second contribution of our work is the application of our video quality assessment methodology in network transmission design.

With the main target of a P2P distribution design, we develop a generic multi-source streaming technique, where the video stream is decomposed into different redundant flows that travel independently through the network. Controlling the number of flows, their rates, and the amount of redundancy transported by each of them, it is possible to choose a configuration as robust as desired in terms of perceived quality. Our method is particularly appropriate for the design of networks with high probability of distribution failures (such as P2P systems) or when a very strict level of quality is needed. In the P2P context, another important feature of the multi-source technique is that it leads to a very low signalling cost (overhead), in contrast with Bittorrent-like approaches.

We introduce the multi-source technique in [globecom07] (it is also partially presented in [bj-hk07]). In this paper we show the interest of the approach by analyzing the impact on quality of extreme cases.

To evaluate the different possible multi-source configurations, we develop analytical models for evaluating the loss processes as functions of failures in servers. Combining it with the PSQA assessment we show how to ensure a high QoE for end users ([lanc07] and [euro-fgi07]).

We then focus our analysis on how to define a configuration for a P2P network that maximizes the delivered QoE based on the heterogeneous peers' lifetimes. The main results concern the joint impact of different frame type losses on the QoE, always using the PSQA methodology, and how to identify an optimal parameter setting, in order to obtain the best possible quality level for a given peers' dynamics [icc08].

This optimal configuration is evaluated in a generic implementation of the multi-source streaming technique [ipom07msource]. It is generic because it allows different multi-source configurations, but also because it accepts different codecs (MPEG-2/4), transport protocols (HTTP, RTP,...) and container formats (OGG, ASF,...).

Previous analysis define a multi-source configuration that is applied at the server side of the network. We also study the protection strategy at the client side. Basically, a client can be protected from distribution failures, changing his buffer size. In [pmccs07] we model this problem, and we obtain an expression of the buffer size that, in particular conditions, ensures a pre-specified quality level.

1.2.3 Efficient Search in Video Libraries

The third main contribution of this work is on the subject of content search. In particular in search caching. Previous subsections concern the design problem of an efficient distribution of live video in a very dynamic environment. This is the main challenge for a broadcast-like service, where the contents (typically TV channels) are predefined. But a completely different situation occurs when the contents are provided by the clients of the network (instead of a broadcaster). In this case, the content itself exhibits a high dynamics. For instance, the Video on Demand (VoD) and MyTV services allow the clients to submit content.

The discovery of very dynamic content can not be solved with traditional techniques, like publications by video podcast or broadcatching. We present the state of the art and some models of content discovery in [jaiio04][jiio04][claio06]. In the P2P context, the design problem of an efficient content discovery is particulary relevant; we explore a solution using specialized cache techniques. We develop a mathematical programming model of the impact of cache expiration times on the total number of correct answers to queries in a content network, and on the bandwidth usage [inoc07].

In order to maximize the number of correct answers subject to bandwidth limitations, optimal values for the cache expiration times are computed in two particular scenarios. First, a file-sharing P2P system [lanc05], from where we expect a similar behavior in the VoD service. Second, the Domain Name System (DNS) [qest06], comparable with the dynamics observed in the MyTV service.

1.2.4 Quality-driven Dynamic Control of Video Delivery Networks

Our final contribution is on the subject of dynamic control of video delivery networks. We use the capacity of the PSQA technology to evaluate the perceived quality of the stream in real-time (that is, to provide the instantaneous value of quality), in order to control or simply to monitor the system. This is exactly the main design idea of our generic monitor suite presented in [ipom07audit]. We develop an effective monitoring and measuring tool that can be used by managers and administrators to assess the current streaming quality inside the network. This tool was designed as a generic implementation, in order to be able to use it with different architectures. Moreover, it can be associated with most common management systems since it is built over the SNMP standard. The tool is a free-software application that can be executed on several operating systems.

We also develop a preliminary design of a centralized tree-based overlay topology, for our P2P system. With very low signaling overhead, the overlay is built choosing which peer will serve which other node. The overlay is made in such a way to diminish the impact of peers disconnection on the quality. We model the overlay building as a mathematical programming problem, where the optimization objective is to maximize the global expected QoE of the network [lagos07]. We present different algorithms for solving this problem, applied to case studies based on real life data [submitted08b].

The tree-based overlay, the monitoring suite, and the optimal multi-source streaming configuration are integrated in a prototype, called GOL!P2P, that is presented in [submitted08a]. The prototype is independent of the operating systems, and it is a free-software application based on well proven technologies.

1.3 Document Outline

The remainder of this document is organized as follows. Chapter 2 introduces basic concepts needed to understand this work. After this common background, we have divided the main contents in three parts. Part II deals with the quality assessment methods. The main point is to obtain a methodology to measure the perceived video quality, methodology that will be used in Parts III and IV. Part III studies and analyzes, with a quantitative evaluation of quality, the main design ideas of our P2P system. The subject of Part IV is to show the main challenges associated with the real deployment of our system. Finally, we present conclusions and perspectives of our work in Chapter 12.

Part II QUALITY. Chapter 3 summarizes the state of the art in quality assessment for video applications. We discuss the main video quality assessment approaches available in the literature, and why they don't necessarily fulfill the current needs in terms of perceived quality assessment. Chapter 4 presents the PSQA hybrid methodology that we will use. In Chapter 5 we explore the sensivity of the perceived video quality with respect to several quality-affecting parameters. We obtain mapping functions of these factors into quality, that will be used in the rest of this work. In particular, we study the effects of frame losses (due to servers failures or network errors) and of video motion on the quality. A comparative analysis of our motion parameters with motion activity metrics presented in the literature is studied in Appendix A.

Part III PEER-TO-PEER. The main target of our work is a quality-centric design of a peer-to-peer prototype for live video streaming. Chapter 6 presents our multi-source streaming technique to deliver live video. In this chapter, we also focus on the consequences of the

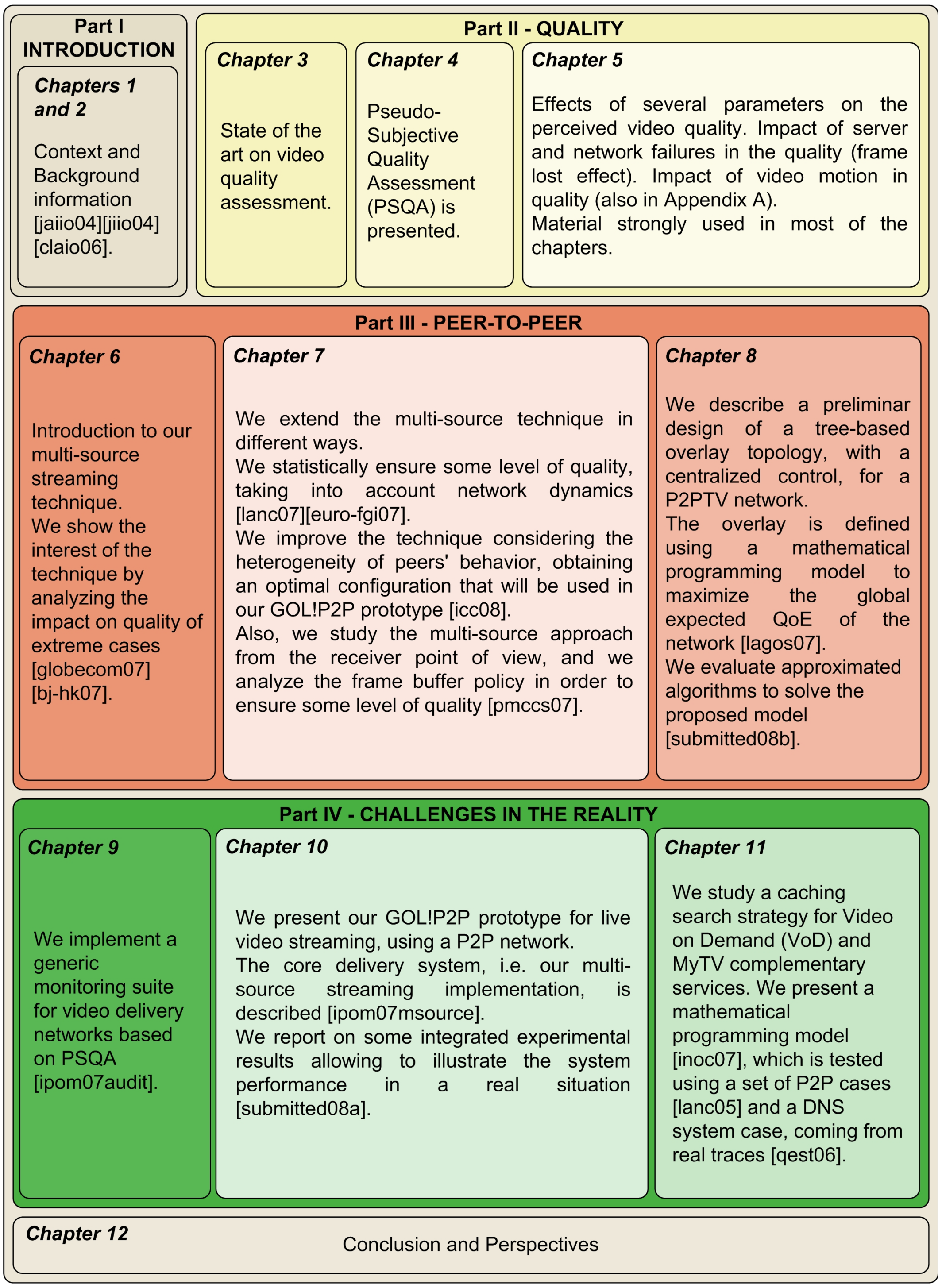

Part I
INTRODUCTION
Chapters 1 and 2
Context and Background information [jaiio04][jiio04] [claio06].
Part II - QUALITY
Chapter 3
State of the art on video quality assessment.
Chapter 4
Pseudo-Subjective Quality Assessment (PSQA) is presented.
Chapter 5
Effects of several parameters on the perceived video quality. Impact of server and network failures in the quality (frame lost effect). Impact of video motion in quality (also in Appendix A).
Material strongly used in most of the chapters.
Part III - PEER-TO-PEER
Chapter 6
Introduction to our multi-source streaming technique.
We show the interest of the technique by analyzing the impact on quality of extreme cases [globecom07] [bj-hk07].
Chapter 7
We extend the multi-source technique in different ways.
We statistically ensure some level of quality, taking into account network dynamics [lanc07][euro-fgi07].
We improve the technique considering the heterogeneity of peers' behavior, obtaining an optimal configuration that will be used in our GOL!P2P prototype [icc08].
Also, we study the multi-source approach from the receiver point of view, and we analyze the frame buffer policy in order to ensure some level of quality [pmccs07].
Chapter 8
We describe a preliminar design of a tree-based overlay topology, with a centralized control, for a P2PTV network.
The overlay is defined using a mathematical programming model to maximize the global expected QoE of the network [lagos07].
We evaluate approximated algorithms to solve the proposed model [submitted08b].
Part IV - CHALLENGES IN THE REALITY
Chapter 9
We implement a generic monitoring suite for video delivery networks based on PSQA [ipom07audit].
Chapter 10
We present our GOL!P2P prototype for live video streaming, using a P2P network.
The core delivery system, i.e. our multi-source streaming implementation, is described [ipom07msource].
We report on some integrated experimental results allowing to illustrate the system performance in a real situation [submitted08a].
Chapter 11
We study a caching search strategy for Video on Demand (VoD) and MyTV complementary services. We present a mathematical programming model [inoc07], which is tested using a set of P2P cases [lanc05] and a DNS system case, coming from real traces [qest06].
Chapter 12
Conclusion and Perspectives

Figure 1.1: Thesis outline map. It shows the four parts of the thesis, the size and the color of each chapter represent its relative contribution in the overall work (larger sizes and clearer colors correspond to more important contributions).

way the stream is decomposed on the resulting quality. Chapter 7 presents a deeper study of our proposed multi-source streaming technique. We show, using real data, how our approach allows to compensate efficiently the possible losses of frames due to peers leaving a P2P system. We improve the technique in different ways, especially considering the peers' dynamics heterogeneity. We obtain an optimal multi-source streaming technique that will be used in the sequel chapters. In this chapter we also study the multi-source approach from the receiver point of view, and we define a frame buffer policy in order to ensure some level of quality.
Chapter 8 explores a preliminary design of a tree-based overlay topology, with a centralized control, for a P2PTV network. We consider the problem of selecting which peer will serve which other one, i.e. the topology construction. A mathematical programming formulation models the problem, while a set of proposed approximated algorithms solves it.

Part IV CHALLENGES IN THE REALITY. Chapter 9 presents a monitoring suite for a generic video delivery network. It measures in real–time and automatically, the perceived quality at the client side by means of the PSQA methodology.
Chapter 10 presents our GOL!P2P prototype for robust delivery high quality live video. It explains the core delivery system, i.e. the multi-source streaming implementation, and it presents the global architecture and its main components.
In Chapter 11 we study the problem of content discovery for the VoD and MyTV complementary services. In particular, a mathematical programming model of the caching search strategy for these services is analyzed.

Figure 1.1 shows a outline map of the thesis, combined with the contributions of each chapter (larger size chapters and clearer color chapters imply more important contributions).

1.4 Publications

This section contains the list of papers published as a result of this PhD thesis work.

[bj-hk07] Rodríguez-Bocca, P., *Exploring Quality Issues in Multi-Source Video Streaming*, in: *2nd Beijing-Hong Kong International Doctoral Forum: Web, Network, and Media Computing (BJ-HK Phd Forum'07). Best Paper Award in the area of Network Multimedia Applications and Systems.*, Hong Kong, SAR, 2007.

[claio06] Rodríguez-Bocca, P. and H. Cancela, *Mecanismos de descubrimiento en las Redes de Contenido*, in: *XIII Congreso Latino-Iberoamericano de Investigación Operativa (CLAIO'06)*, Universidad de la República, Montevideo, Uruguay, 2006.

[euro-fgi07] Rodríguez-Bocca, P., H. Cancela and G. Rubino, *Modeling Quality of Experience in Multi-source Video Streaming*, in: *4th Euro-FGI workshop on "New trends in modelling, quantitative methods and measurements" (Euro-FGI WP IA.7.1)*, Ghent, Belgium, 2007.

[globecom07] Rodríguez-Bocca, P., H. Cancela and G. Rubino, *Perceptual quality in P2P video streaming policies*, in: *50th IEEE Global Telecommunications Conference (GLOBECOM'07)*, Washington DC, United States, 2007.

[icc08] da Silva, A. P. C., P. Rodríguez-Bocca and G. Rubino, *Optimal quality–of–experience design for a p2p multi-source video streaming*, in: *IEEE International Conference on Communications (ICC 2008)*, Beijing, China, 2008.

[inoc07] Rodríguez-Bocca, P. and H. Cancela, *Modeling cache expiration dates policies in content networks*, in: *International Network Optimization Conference (INOC'07)*, Spa, Belgium., 2007.

[ipom07audit] Vera, D. D., P. Rodríguez-Bocca and G. Rubino, *QoE Monitoring Platform for Video Delivery Networks*, in: *7th IEEE International Workshop on IP Operations and Management (IPOM'07)*, San José, California, United States, 2007.

[ipom07msource] Rodríguez-Bocca, P., G. Rubino and L. Stábile, *Multi-Source Video Streaming Suite*, in: *7th IEEE International Workshop on IP Operations and Management (IPOM'07)*, San José, California, United States, 2007.

[jaiio04] Rodríguez-Bocca, P. and H. Cancela, *Redes de contenido: un panorama de sus características y principales aplicaciones*, in: *33th Argentine Conference on Computer Science and Operational Research, 5th Argentine Symposium on Computing Technology (JAIIO'04)*, Córdoba - Argentina, 2004.

[jiio04] Rodríguez-Bocca, P. and H. Cancela, *Introducción a las Redes de Contenido*, in: *Jornadas de Informática e Investigación Operativa (JIIO'04)*, Montevideo, Uruguay, 2004.

[lagos07] Cancela, H., F. R. Amoza, P. Rodríguez-Bocca, G. Rubino and A. Sabiguero, *A robust P2P streaming architecture and its application to a high quality live-video service*, in: *4th Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS'07)*, Puerto Varas, Chile, 2007.

[lanc05] Rodríguez-Bocca, P. and H. Cancela, *A mathematical programming formulation of optimal cache expiration dates in content networks*, in: *Proceedings of the 3rd international IFIP/ACM Latin American conference on Networking (LANC'05)* (2005), pp. 87–95.

[lanc07] Rodríguez-Bocca, P., H. Cancela and G. Rubino, *Video Quality Assurance in Multi-Source Streaming Techniques*, in: *IFIP/ACM Latin America Networking Conference (LANC'07). Best paper award by ACM-SIGCOMM*, San José, Costa Rica, 2007.

[pmccs07] da Silva, A. P. C., P. Rodríguez-Bocca and G. Rubino, *Coupling QoE with dependability through models with failures*, in: *8th International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS'07)*, Edinburgh, Scotland, 2007.

[qest06] Cancela, H. and P. Rodríguez-Bocca, *Optimization of cache expiration dates in content networks*, in: *Proceedings of the 3rd international conference on the Quantitative Evaluation of Systems (QEST'06)* (2006), pp. 269–278.

[submitted08a] Rodríguez-Bocca, P. and G. Rubino, *A QoE-based approach for optimal design of P2P video delivering systems*, submitted.

[submitted08b] Martínez, M., A. Morón, F. Robledo, P. Rodríguez-Bocca, H. Cancela and G. Rubino, *A GRASP algorithm using RNN for solving dynamics in a P2P live video streaming network*, submitted.

Chapter 2

Background

Some background information is needed to understand this work. This chapter briefly recalls the main needed concepts.

2.1 Digital Video Overview.

Digital video is a video representation using digital signals. It refers to the capturing, manipulation, distributing and storage of video in digital formats, allowing it to be more flexible and rapidly manipulated or displayed by a computer.

Digital video enabled a revolution in the associated research areas. Video compression received an important research attention from the late 1980's. It enabled a variety of applications including video broadcast over digital cable, satellite and terrestrial, video-conferencing, digital recording on tapes and DVD's, etc. Video communication over best-effort packet networks started in the mid 1990's, with the growth of the Internet. In the Internet, the packet losses, the time-varying bandwidth, and the delay and jitter, are the main difficulties faced by video delivery.

2.1.1 Compression and Standard Coding of digital video

Video compression is the technique designed to eliminate the similarities or redundancies that exist in a video signal. Roughly speaking, a digital video signal is a temporal sequence of digital pictures (or frames). Consecutive pictures in the sequence have temporal redundancy since they typically show some movement over the same objects. Inside a picture there is spatial and color redundancy, where the values of nearby pixels are correlated. Besides eliminating the redundancy, the most used compression techniques are lossy, where they encode only perception relevant information, reducing the irrelevant data and increasing the compression ratio.

A compression method is completely specified by the encoder and decoder systems, jointly called CODEC (enCOder/DECoder). Main codecs today are based on the standards proposed by the Moving Pictures Expert Group (MPEG).

MPEG-2 is an umbrella of international compression standards [167], developed by the

MPEG group. Several of its parts were developed in a joint collaborative team with ITU-T (for instance, the specific codecs H.261 [176] and H.263 [177]). After the success of MPEG-2, the MPEG-4 was standardized by the same teams. The key parts to be aware of are MPEG-4 part 2 [169] (used by codecs such as Xvid [354]) and MPEG-4 part 10 [168] (known also as MPEG-4 AVC/H.264). In addition to standard codecs, there are some proprietary ones. The main and most used examples are RealVideo [268] and Windows Media Video (in the process of standardized).

Standard codecs and proprietary codecs use the same compression principles, and therefore by understanding one of them we can gain a basic understanding of the whole compression area.

Concerning the MPEG-2 and MPEG-4 specifications, there are important differences between them. Perhaps the most basic one is that MPEG-2 is a pixel-oriented specification while MPEG-4 is an object-oriented one. But there are also many common features. One of them is the compression method they use. In both specifications the idea behind compression is the same: the numerous temporal redundancies between frames are exploited by means of using motion estimation and compensation. A key concept in MPEG-2 is the frame, or picture, the transmission unit, in some sense; the corresponding concept in MPEG-4 is the the video object plane (VOP). Both concepts represent a coded picture. There are three main frame types: the Intra frames (I-frame), the Predicted frames (P-frame), and the Bidirectional or Interpolated frames (B-frame). An I-frame codes a full image; this means that an I-frame can be decoded into an image independently of any other frame in the stream. A P-frame is a frame that is predicted (using motion compensation), based on another previously decoded P or I frame. A B-frame is a frame that is predicted based on past as well as future P or I frames.

Figure 2.1 shows the inter-dependencies of the three main frame types. These frame types

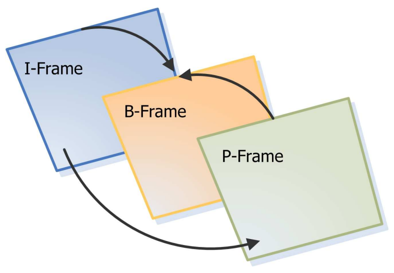


Figure 2.1: Frame type dependencies. An I-frame represents an image. A P-frame is a frame that is predicted based on another previously decoded P or I frame. A B-frame is a frame that is predicted based on past as well as future P or I frames.

are defined in MPEG-2. They also exist in MPEG-4, although in this case there are other types of frames; however, in MPEG-4 the most important one are the three mentioned types.

The video sequence is divided in sets of frames, called Group of Pictures (GOP). A GOP contains a small number of frames that can be decoded alone (without reference to frames outside the group). Typically a GOP has only one I-frame at the beginning.

Usually, an error in the I-frame propagates until the next I-frame (because the inter-dependencies of the other frames in the GOP). Errors in the P-frames propagate until the next I-frame or P-frame. B-frames do not propagate errors. The more I-frames the video has, the more robust to failures it is. However, having more I-frames increases the video size. In order to save bandwidth, videos prepared for Internet broadcast often have only one I-frame per GOP. In MPEG-2, the GOP size is around 20 frames, while in MPEG-4, it rises to 250 frames. It is possible because the MPEG-4 coding has more flexibility and expressiveness then its predecessor, allowing a better compression with greater use of P and B frames.

2.1.2 Digital video applications and Streaming Issues

The digital video can be stored, for future use, or transmitted, for a remote playback. There exist a diverse range of video applications, which have very different properties and constraints.

For instance, a video communication application can be for one-to-one communication (like video calls or video-on-demand), for one-to-many communication (like video conferencing and multiplayer games), or for one-to-all communication (like broadcast TV). The communication can be interactive or bi-directional (like video conferencing), or non-interactive (like broadcast TV or video-on-demand). The video can be encoded in real–time (live TV), or pre-encoded (like video-on-demand).

The specific application constraints strongly influence the design of the system. A summary of the most successful video delivery networks are presented in section 2.2. Next, we briefly present the mechanisms to deliver video over packet networks.

Video streaming stands for the simultaneous delivery and playback of video over packet networks [19]. Each frame must be delivered and decoded by its playback time, therefore the sequence of frames has an associated sequence of deliver/decode/playback deadlines, known as real–time constraints.

Basically, in a streaming mechanism, the video is split into pieces (in a particular container format), which are sequentially transmitted over the network (with a transport protocol), and the receiver decodes and reproduces the video as the components of the stream arrive.

Nowadays there are many different *container* formats such as MPEG-TS, MPEG-PS, AVI, OGM and so on. Many of these formats focus on some specific application type. For example MPEG-PS is suitable for transmission in a relatively error-free environment, while MPEG-TS (Transport Streams) is suitable for transmission in which there may be potential packet losses or corruption by noise. In this thesis we used MPEG-TS as our container format.

The MPEG-TS specification basically addresses two main features, the multiplexing of digital video and audio and the output synchronization. It also defines rules for conditional access management and multiple program transport streams (i.e., sending more than one program at a time). The audio or video encoder output is called an Elementary Stream (ES). In a further step, elementary streams are packetized into smaller units, generating the Packetized Elementary Stream (PES). Then, these PES are packetized, in turn, into still smaller packets called transport stream (TS) packets. At this level is where the audio and video multiplexing is done, creating an unique stream called transport stream (TS). Finally this transport stream is encapsulated into some transport protocol packets in order to send it through the network. Nowadays there are various transport protocols suitable for the streaming transport over a network; some of them are: User Datagram Protocol (UDP), Real-time Transport Protocol (RTP) and Hypertext Transfer Protocol (HTTP)[1]. Each one of these protocols has its own advantages in specific situations, together with some disadvantages when using them in certain types of scenarios. UDP [142] is a non-connection-oriented protocol, appropriate for live streaming, for instance. RTP [146] was specifically designed to stream media over networks. It was built on the top of UDP and, as its name suggests, it is designed for real-time streaming. Finally, HTTP [148] is a connection-oriented protocol that guarantees correct delivery of each bit in the media stream. Actually 40% of the Internet streaming is transported over HTTP [79], and in most cases proprietary streaming protocols are used. An HTTP stream has the enormous advantage of bypassing the firewalls protections. This makes it very popular in the Internet users community. HTTP is the protocol typically used in Internet TV systems.

2.2 Video Delivery Networks

A **Video Delivery Network** (VDN) is a system that delivers video transparently to end users. The area of Video Delivery Networks is growing nowadays, because of the larger bandwidth available on the access networks (on the Internet, cellular networks, private IP networks, etc.), the new business models proposed by video content producers, and the capability of new hardware to receive video streams.

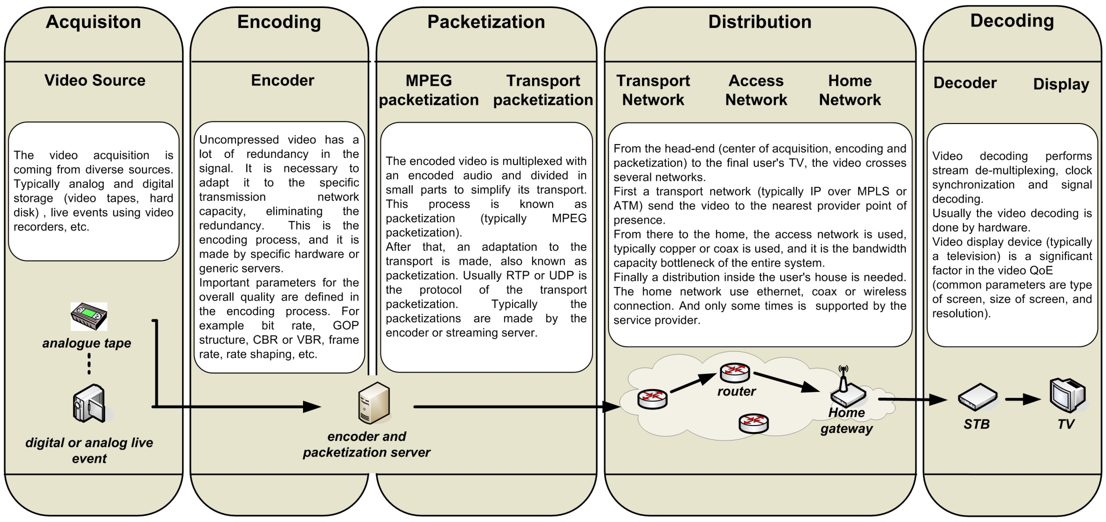
Service providers have developed many types of services using video streaming, for instance: broadcast TV (or live TV), video on demand (VoD), video conferencing, security monitoring, etc. In this work we concentrate on VoD and live TV services. Also, different video formats and resolutions are being used, depending on the nature of the service and the network constraints. From the network point of view, VDNs have many different architectures, and there are specific technological challenges associated with each of them. The most successful networks today can be classified into the following categories: Digital Cable and Satellite, Internet Television, P2PTV, IPTV, and Mobile TV.

We describe next the main characteristics of the most important Video Delivery Network (VDN) architectures, with special focus on the main factors that affect the overall Quality of Experience (QoE).

2.2.1 A Simple Network Model.

A generic high level end-to-end video services delivery network model is shown in Figure 2.2. The data layer of the network is composed of five stages: acquisition, encoding packetization, distribution, and decoding. There are many possibles sources for video acquisition. Some examples are analog and digital storage (video tapes, hard disk) or live events using video recorders. Uncompressed video has a large amount of redundancy in the signal. This redundancy is eliminated (or reduced) when adapting the signal to the specific transmission network capacity by the encoding process, a task typically done by specific hardware or generic servers.

[1]There are other important proprietary streaming protocols, like Microsoft Media Server (MMS), RealMedia, etc.



The video acquisition is coming from diverse sources. Typically analog and digital storage (video tapes, hard disk) , live events using video recorders, etc.

Uncompressed video has a lot of redundancy in the signal. It is necessary to adapt it to the specific transmission network capacity, eliminating the redundancy. This is the encoding process, and it is made by specific hardware or generic servers.
Important parameters for the overall quality are defined in the encoding process. For example bit rate, GOP structure, CBR or VBR, frame rate, rate shaping, etc.

The encoded video is multiplexed with an encoded audio and divided in small parts to simplify its transport. This process is known as packetization (typically MPEG packetization).
After that, an adaptation to the transport is made, also known as packetization. Usually RTP or UDP is the protocol of the transport packetization. Typically the packetizations are made by the encoder or streaming server.

From the head-end (center of acquisition, encoding and packetization) to the final user's TV, the video crosses several networks.
First a transport network (typically IP over MPLS or ATM) send the video to the nearest provider point of presence.
From there to the home, the access network is used, typically copper or coax is used, and it is the bandwidth capacity bottleneck of the entire system.
Finally a distribution inside the user's house is needed. The home network use ethernet, coax or wireless connection. And only some times is supported by the service provider.

Video decoding performs stream de-multiplexing, clock synchronization and signal decoding.
Usually the video decoding is done by hardware.
Video display device (typically a television) is a significant factor in the video QoE
(common parameters are type of screen, size of screen, and resolution).

Figure 2.2: Video Delivery Network model architecture. Data layer of the network is composed of five stages: acquisition, encoding, packetization, distribution, and decoding.

Important parameters for the overall quality are defined in this encoding process. For example, the bit rate used and the choice between constant and variable rates (CBR or VBR), the GOP (Group of Pictures, see below) structure, the frame rate possible, rate shaping, etc. The encoded video signal is multiplexed with an associated encoded audio flow, and it is divided into small pieces to simplify its distribution. This process is known as packetization (typically, MPEG packetization). Then, an adaptation to the transport layer is performed, also known as packetization. Usually, RTP or UDP is the protocol used for the transport packetization. Packetizations are normally performed by the encoder or streaming server. From the head-end (center of acquisition, encoding and packetization) to the final user's terminal, the video flow travels through several networks. First, a transport network (typically IP over MPLS or ATM) sends the video to the nearest provider point of presence. From there to the home, the access network is used, typically based copper or coaxial, and this is the bandwidth capacity bottleneck of the entire system. Finally, a distribution inside the user's house is needed. The home network uses Ethernet, coaxial or wireless connections. This distribution can occasionally be supported by the service provider. Video decoding performs stream de-multiplexing, clock synchronization and signal decoding. Usually the video decoding process is hardware-based. The video display device (typically a television set) is an important components of the final video quality; some examples of the associated factors are the type of screen, its size, or its resolution.

Internet Television. On the Internet, the majority of VDN have a traditional CDN (Content Delivery Network) structure [72, 346], where a set of datacenters absorbs all the load, that is, concentrates the task of distributing the content to the customers. This is, for instance, the case of msnTV [225], YouTube [360], Jumptv [187], myTVPal [228], etc., all working with specific video contents[2].

P2PTV. Another method becoming popular in the Internet these days consists in using the often idle capacity of the clients to share the distribution of the video with the servers, through the now mature Peer to Peer (P2P) systems. Today, the most successful P2PTV networks are iMP [27] (from BBC), Joost [185], PPlive [260] and TVUnetwork [326], PPstream [261], SopCast [312], TVAnts [325]. These are virtual networks developed at the application level over the Internet infrastructure. The nodes in the network, called *peers*, offer their resources (bandwidth, processing power, storing capacity) to the other nodes, basically because they all share common interests (through the considered application). As a consequence, as the number of customers increases, the same happens with the global resources of the P2P network. This is what we call *scalability* in the Internet. Clearly, using a P2P infrastructure for video distribution appears to be a good idea, due to the high requirements in terms of bandwidth of these applications, and we observe that streaming services in VoD (Video on Demand) have actually similar characteristics than popular P2P systems for file sharing and distribution. However, real-time video streaming (live TV) has different and strong constraints that imply a series of specific technical problems because of the before-mentioned P2P dynamics.

IPTV. IPTV is a network architecture where digital broadcast television and VoD services are delivered using the Internet Protocol over a dedicated and closed network infrastructure. IPTV is typically supplied by a broadband operator together with VoIP and Internet access (the set of services is called “Triple Play”). IPTV is not TV that is broadcasted over the public Internet, as in the previously mentioned Internet TV or P2PTV, it is another competitor in this market. Given the architecture complexity of IPTV and the typical lack of experience in video of traditional telecommunication companies, in general the IPTV solutions are provided by an integrator. Well known names in the market are Alcatel [9], Siemens [309], and Cisco [59].

MobileTV. MobileTV is a name used to describe a video delivery service (live and VoD video) to mobile phone devices. MobileTV is typically supplied by a mobile operator. One difference with the previous described systems is that MobileTV has two possible architectures at the access network: either it uses the two-way cellular access network (W-CDMA, CDMA2000, GSM, GPRS, etc.) or it relies on a dedicated digital broadcast network (DVB-H, DMB, TDtv, ISDB-T, etc.). It is expected that the cellular network will be used for unicast delivery (i.e. VoD) and the broadcast network will be used for multicast services (live TV). The implementations start to comply with the IP Multimedia Subsystem (IMS) architecture [1], and, as in the IPTV case, this market is dominated today by the main integrators.

Digital Cable and Satellite. The difference between IPTV and Digital Cable concentrates at the distribution stage. Digital Cable provides television to users via radio frequency signals transmitted through coaxial cables and/or optical fiber distribution networks. It may be either a one-way or a bidirectional distribution system. In the DVB-C standard, Digital Cable

[2]In general, the technologies used are derived from those associated with Web applications, thus based on the Hypertext Transfer Protocol (HTTP) [184, 203, 311].

uses a spectrum with more than 100 carriers of 6 MHz channels, typically used to carry 7-12 standard digital channels (MPEG-2 MP@ML[3] streams of 3-5 Mbps). The modulation is the Quadrature amplitude modulation (QAM) technology, in the 64-QAM and 256-QAM versions. The abbreviation CATV originally meant "Community Antenna Television"; however, CATV is now usually understood to mean "Cable TV". The Digital Satellite television uses a satellite to distribute the signal to the users. It typically uses the Quadrature Phase-shift keying (QPSK) digital modulation, and the standards MPEG-2 or MPEG-4 AVC, and DVB-S (or DVB-S2).

2.2.2 Quality-of-Experience (QoE) in a VDN.

A common challenge of any Video Delivery Network deployment is the necessity of ensuring that the service provides the minimum quality expected by the users. Quality of Experience (QoE) is the overall performance of a system from the users' perspective [4]. QoE is basically a subjective measure of end-to-end performance at the service level, from the point of view of the users. As such, it is also an indicator of how well the system meets its targets. To identify factors playing an important role on QoE, some specific quantitative aspects must be considered [87]. For video delivery services, the most important one is the perceptual video quality measure.

In the previous subsection, we described the most successful classes of VDN nowadays. Depending on the service provided, each type of VDN has its own specific protocols and components. First, observe that there are no significant differences on the video sources of the network. Obviously, the quality of the original video (analog or digital) greatly affects the encoding efficiency and the overall QoE. Certainly, better quality is expected in dedicated networks as in the IPTV, Digital Cable and Satellite cases (for example, these networks typically use MPEG-2 or MPEG-4 video sources, in comparison with P2PTV that uses home technology). Video encoding is accomplished using video codecs appropriate to the particular network. In the Internet environment (Internet TV and P2PTV), the video encoders are implemented with dedicated servers, or simply with home PCs. In the dedicated networks (IPTV, MobileTV, cable...) a hardware appliance encoder is preferred.

The encoder parameters (like the encoder standard used, the bit rate, the GOP structure, CBR/VBR, etc.) are defined taking into account the transmission method and the bandwidth requirements. Since encoders and decoders are basically software-based, there is a large freedom for the specific designs, in the Internet world. The standard case for the applications is the use of proprietary codecs (sometimes MPEG-4 – xvid, H.264, sometimes SMPTE VC-1 – Windows Media 9).

[3]The MPEG-2 specification defines profiles and levels to allow specific applications to support only a subset of the standard. While the profile defines features (such as the compression algorithm), the level defines the quantitative capabilities (such as maximum bit rate) in this profile. At least the main profile (MP) and the main level (ML) (for instance the DVD quality) are accepted in the digital cable and satellite.

[4]Sometimes QoE is confused with QoS (Quality of Service). QoS is related to objective measures of performance at the network level and from the network point of view. QoS also refers to the protocols and architectures that enable to provide differentiated service levels, as well as managing the effects of congestion on application performance.

In IPTV and digital cable and satellite, MPEG-2 implementations are being smoothly replaced by new encoders such as H.264 and VC-1. In MobileTV, following the 3GPP recommendations, MPEG-4 Part 2 is used. The bit rate affects the quality substantially. Internet TV and P2PTV typically use broadband home connections (of about 500 Kbps). In MobileTV, the access network limits the bandwidth available to about 100 Kbps in 2G networks and at most 2 Mbps in the case of 3G networks. In IPTV and digital Cable and Satellite a minimum of 12 to 24 Mbps is recommended.

The MPEG packetization used to multiplex video and audio, and to divide it into small pieces in order to simplify the distribution. Usually MPEG-TS is used for this task, for single channel distribution (Single Program Transport Streams) as in IPTV, or for multiple channel distribution (Multiple Program Transport Stream) as in Digital Cable. In Internet TV and P2PTV an important amount of standards and proprietary protocols are used. After the packetization phase, the transport packetization is performed, typically by the encoder or by streaming server. Again, there are many standards and proprietary protocols for this processes in the Internet.

The distribution characterizes the different VDN architectures considered. The distribution introduces some delay, jitter, and loss in the stream. In live TV and VoD the most important factor is the packet loss because the video quality and the overall QoE degrades severely with it (in Chapter 5 we study the effects of several factors on the perceived video quality, and the most important ones are referred to the distribution). Fixed access of IPTV and digital cable (e.g., ADSL2+, cable modems, Ethernet, etc.) can control the QoS of the distribution with intelligent priority-marking algorithms, except in the home network where wireless is used (Wi-Fi), degradating potentially the quality. Digital Satellite television and mobileTV depend significantly on the variable environment conditions. In P2PTV the access network is an overlay network of peers that give (part of) their bandwidth to the others. The main problem here is that peers connect and disconnect with high frequency, in an autonomous and completely asynchronous way. This means that the main challenge in P2P design is to offer the quality needed by the clients in such a highly varying environment, trying to minimize packages losses. A set of methods are being used in access networks to mitigate packet losses. Some examples are Forward Error Correction (FEC), retransmission, selective discard, and in particular situations as in P2PTV, the multiple source or the multiple path streaming.

Video decoding is done by a dedicated hardware (Set-Top-Box for IPTV, Cable, Satellite and mobileTV), except in the Internet where a personal computer is typically used. Finally, as stated above, the television terminal is another important component of the QoE, with, as main associated factors, the type of screen, its size, or its resolution.

The components of a video delivery network are summarized in Figure 2.3 and its protocols are shown in Figure 2.4.

2.2.3 Summary

A Video Delivery Network (VDN) is a system that delivers video transparently to end users. VDNs have different architectures: Digital Cable and Satellite, Internet Television, P2PTV, IPTV, and Mobile TV. No matter how well-designed a network is or how rigorous its QoS controls are, there is always the possibility of failures in its components, of congestion, or of

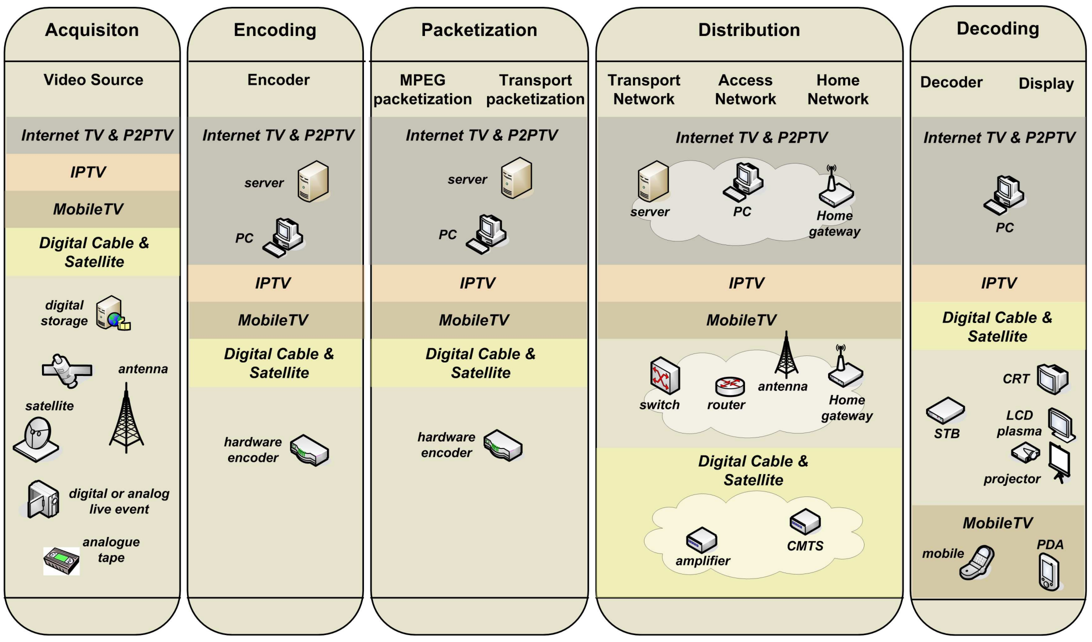


Figure 2.3: Components of a Video Delivery Network.

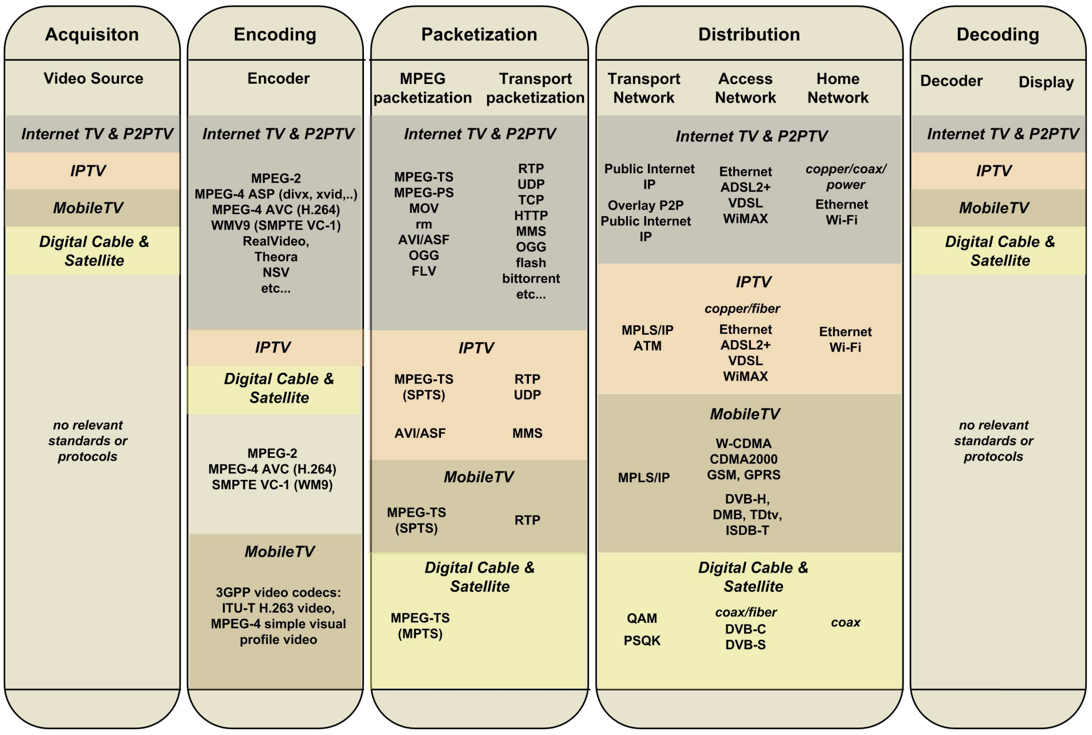


Figure 2.4: Protocols of Global architecture.

errors creeping into the video stream. Therefore, it is necessary to monitor and to measure the overall quality in a Video Delivery Network, and especially the perceived quality, representing the user's point of view. In next section we study with more detail the Video Delivery Network used in the Internet (Internet TV and P2PTV). In Chapter 9 we detail a general video delivery network monitoring suite (applicable to all the VDN architectures), that measures, in real–time and automatically, the perceived quality at the client side. In Chapter 10 we present our GOL!P2P prototype, a P2P architecture for real–time (live) video distribution.

2.3 Peer-to-Peer Network Architecture for Video Delivery on Internet

For the specific context of the Internet, the presented VDN architectures (Internet TV and P2PTV) are enclosed inside the general concept of Content Networks. After introducing them in general, we present, in the particular case of video flows, the Content Delivery Networks (CDN) and the Peer-to-Peer networks (P2P). Finally, we focus on the state of the art of the Peer-to-Peer network design for video delivery.

2.3.1 Introduction, the Content Networks

A **Content Network** is a network where the addressing and the routing of the information is based on the content description, instead of on its physical or logical location [199, 200, 216]. Content networks are usually virtual networks built over the IP infrastructure of Internet or of a corporative network, and use mechanisms to allow accessing a content when there is no fixed single link between the content and the host (or hosts) where this content is located. Even more, the content is usually subject to re-allocations, replications, and even deletions from the different nodes of the network. In the last years many different kinds of content networks have been developed and deployed in widely varying contexts, they include (see Figure 2.5): the domain name system [39, 143, 144, 242], peer-to-peer networks [30, 60, 82, 88, 89, 115, 190, 230, 305], content delivery networks [7, 58, 247–250], collaborative networks [33, 34, 121, 240], cooperative Web caching [94, 98, 181, 262, 269, 313], subscribe-publish networks [23, 45], content-based sensor networks [92, 129, 130], backup networks [85, 198, 287, 288], distributed computing [69, 96, 112, 305], instant messaging [141, 180], multiplayer games [84], and search engines [116, 356].

The ability of Content Networks to take into account different application requirements (like redundancy and latency) and to gracefully scale with the number of users have been a main factor in this growth [251, 262, 272]. Other motivations for their deployment are the possibility to share resources (for example in P2P networks), and the simple design to avoid centralized, single failure points or bottlenecks.

In [274, 276, 277, 279] we present a taxonomy of Content Networks, based on their architectures. Some of the characteristics studied were the decentralization of the network, the content aggregation and the content placement. We also analyze their behavior, in terms of performance and scalability.

Figure 2.5: Example of current Content Networks (CNs).

Depending on the specific application of the content network, and mainly on the content dynamics in the network, different architecture designs are used. The design areas of a content network are:

- **Content Discovery.** The objective of a content network is to transparently present the contents to the end users, i.e. present a consistent global view of the contents. In order to reach this objective, the network must be able to answer each content query with the most complete possible set of nodes where this content can be found; this corresponds to discovering the content location in the most effective and efficient possible way.

  The content discovery is introduced in Section 2.4, and studied in depth in Chapters 8 and 11. In the P2P Section 2.3.3, we reference it in our particular context.

- **Content Distribution.** Knowing the nodes where each specific content is to be found, a client of the network retrieves the content of these sources.

  The task of delivering the content is call content distribution. Depending on the kind of content, different restrictions must be considered. Some unique contents (like chats or multiplayer games) have to be exclusively distributed from the source, file sharing is typically distributed from all the replicated sources in a best effort form, video streaming has real–time restrictions, etc.

  The content distribution has been intensively studied in each kind of content network, in order to improve the efficiency of the distribution, or the availability of the content. It is also studied in te context of P2P networks as a mechanism with incentives to share, and to avoid or mitigate the free-riders (peers that only consume the content but not serve them to other ones).

In our work we focus on the video distribution in order to improve the quality-of-experience (QoE). See 2.3.3.3 for the special P2P case. Our *multi-source streaming technique* is presented in chapters 6 and 7.

- **Network Dynamics.** The *network effect* is defined as the phenomenon whereby a product or service becomes more valuable as more customers use it, thus encouraging even increasing numbers of customers [235, 236]. Usually, content networks have a very high network effect potential, specially P2P networks, and it is considered in depth, for example with game theory.

  If two networks offer the same kind of content, they are in competition. Also the network dynamics is studied in the competence context.

- **Others.** Other design factors are commonly studied, like security, anonymity, interoperability, etc.

These last years have seen an explosion on the design and deployment of different kinds of content networks, in most cases without a clear understanding of the interaction between network components neither of the tuning of the network architecture and parameters to ensure robustness and scalability and to improve performance. This in turn has led to a still small (but growing) number of empirical studies (based on large number of observations of a given network activity) [186, 272, 300, 302, 361], and of analytical models which can be fitted to the observations in order to better understand and possibly to predict different aspects of network behavior [99, 251, 262, 358, 359].

2.3.2 Content Networks for Video Delivery on Internet

In Section 2.2 we introduce two main architectures for the Internet-based Video Delivery Networks: Internet TV and P2PTV. In the present section we will explain them in depth. Depending on their number of simultaneous clients, Internet TV is deployed using a simple *Server Farm* or a scalable *Content Delivery Network* (CDN). On the other hand, P2PTV is based on *Peer-to-Peer* (P2P) systems.

Different architectures are needed for the same service because of scale and cost reasons (the deployment of this service on Internet is very expensive, mainly due to the bandwidth costs).

2.3.2.1 Scale and Cost Problem of Live-video Delivery on Internet

Following [12], the scale and cost problem results from the fact that only one-to-one communications are available on Internet, which implies sending a different copy of the video to each client.

The bandwidth is the most expensive resource on Internet, and video delivery is one of the services that most demand it. In particular, the *cost* of live video delivery is proportional to the number of simultaneous clients, and nowadays it is the first limitation for the expansion of availability and of diversity of video content on Internet. Moreover, live-video delivery is hard to *scale* without a strict planning because the streaming of popular videos cause high congestion in most Internet sectors. Congestion implies high number of losses in the distribution, and therefore a significant perceived quality degradation by clients.

But point-to-point communication is not a limitation of the Internet Protocol (IP) standard, it is a traditional Internet Service Provider (ISP) business rule based on lack of standards maturity and revenue preservation[5]. In IP, one-to-one communication is called *unicast*, and the standard also allows one-to-all communication (*broadcast*), and one-to-many communication (*multicast*). While in IP broadcast, data is sent (only once by the source) to all possible destinations, in IP multicast, data is sent (only once by the source) to all requesters that have registered their interest in it from that source. IP broadcast and IP multicast are limited to dedicated networks (corporate or enterprise networks). Applications that use IP multicast include IPTV architectures (see Section 2.2), videoconferencing, distance learning, corporate communications, software distribution, routing protocols, stock quotes, news, etc. IP Multicast [153, 154] is able to minimize the bandwidth consumption in the network. A requester must join a multicast group (by using the Internet Group Management Protocol (IGMP) [156, 162]) to receive a particular multicast content. Routers build distribution trees for each multicast group (by using Protocol Independent Multicast (PIM) [158, 159, 161]) which replicate and distribute multicast content to all requesters that are in this group.

2.3.2.2 Content Delivery Networks (CDN): global scalability

Without IP multicasting on Internet, video delivery infrastructure usually begins with a server farm, using unicast communication. The farm allows high availability and load balancing between its servers. It has an available (and expensive) bandwidth, B Kbps, that is shared by the set of servers (usually uniformly). If the stream consumes bw Kbps, the total number of simultaneous clients has to be less than B/bw. It is possible to increase the number of clients, growing the available bandwidth B (increasing the costs) or reducing the video quality (reducing bw). For example if $bw = 500$ Kbps, and we have four thousands clients (a small number for a global system), we need more than $B = 2$ Gbps (that is a very high value for most business models). Actual servers can stream at approximately 800 Mbps, and can be considered as extremely dependable. Nevertheless, when the number of clients increases, the server farm needs to be upgraded to several server farms, or to a Content Delivery Network to avoid bottlenecks at the farm, and to reduce latency to end users.

Current successful Internet-based Video Delivery Networks have a traditional Content Delivery Network (CDN) structure. A Content Delivery Network (CDN) [140, 264, 332] is a system of servers located at strategic points around the Internet that cooperate transparently to deliver content to end users. Basically, all nodes in a CDN can serve the same content. The preference for a particular node to serve a particular client, is decided using (possible proprietary) protocols between servers to identify which is the best one to do the task. Each particular

[5]There are some exceptions to this rule. MBONE [91] (short for "Multicast Backbone") was a virtual IP network created to multicast video from the Internet Engineering Task Force (IETF) meetings. MBONE was used by several universities and research institutions. Recently, the U.S. Abilene Network [3] has been created by the Internet2 community and natively support IP multicast. The BBC multicast initiative [28] is another example of current efforts.

CDN has its own rules to determine the best server, usually based on network proximity, network congestion, etc.

The CDN approach was originally developed to improve the World Wide Web service. It reduces client latency and network traffic by redirecting client requests to a node closest to that client. In addition, it improves the availability of the service, avoiding single points of failures and distributed denial of service. Today, there is a wide range of academic knowledge that discusses different aspects of the use of CDN to WWW replication (for instance, see the books [140, 264, 332] and the surveys [184, 203, 311]). Some of the most important aspects that differentiate each CDN approach are: the request routing mechanism that determine which node will serve each client, where to replicate the content (i.e. where to place a node of the CDN in Internet, and in which node to replicate each content depending on its popularity), how to maintain the consistency between the servers considering content changes, and how the CDN adapts to the content, server and client dynamics. Current successful CDNs are the following: Akamai, with tens of thousands of replica servers placed all over the Internet [7, 81]; CDNetworks, with a big deployment in Asia [50]; Globule, an open-source collaborative CDN [14, 256–258]; CoDeeN [97], an academic testbed built on top of PlanetLab [259]; etc. [247–250].

Considering that video delivery is one of the most bandwidth-demanding applications running on the Internet, it is natural to think that CDNs will extend their service to video delivery as well. Nowadays, all the commercial CDNs deployments include streaming, specially for video on demand, because technically it is very similar to file downloading. Also particular CDNs are developed for media delivery service, for audio streaming [166, 202, 227, 265, 317, 339], and video streaming [49, 306, 335]. The most successful Internet TV deployments use a CDN architecture; this is, for instance, the case of YouTube [360], msnTV [225], Jumptv [187], etc. Behind the commercial CDNs, there are some academic studies [72, 193, 346].

From the scale and cost point of view, the CDN architecture solves the scalability problem of the server farm (avoiding bottlenecks); but the cost problem persists, because in a CDN a set of servers absorbs all the load (i.e. concentrates the task of distributing the content to the customers). Therefore, as in the server farm solution, the cost is proportional to the total number of simultaneous clients. In addition the cost of operating and maintaining the servers, placed worldwide, has to be considered. This operational cost usually limits CDN application to large commercial sites only [187, 225, 360].

Moreover, a CDN solves the scalability problem partially, for the following reasons:

- The available bandwidth is not dynamic; therefore, a CDN does not scale well with very popular content that has flash crowd behavior [314].
- The CDN avoids bottlenecks if the CDN servers are close to the clients. For example, this is easy for a worldwide content, but impossible if some content is very popular inside a network. For example, football (soccer) matches are one of the most popular live contents in the world, but, for instance, the "Uruguay-Brasil" match will have lot of fans inside Uruguay, and it would not help much (to these fans) to have many servers located outside Uruguay, because they will share the same congested link to the clients.

2.3.2.3 Peer-to-Peer Networks (P2P): cutting off the costs

From the scalability point of view, if we want to deliver video of different popularities, it seems to be necessary to have the server as close as possible of the final client. From the cost point of view, it would be ideal to distribute the load of the video delivery with the interested ones (the clients).

A method becoming popular these days consists of using the often idle capacity of the clients to share the distribution of the video with the servers, through the present mature Peer to Peer (P2P) systems. **P2PTV** are virtual networks developed at the application level, where the nodes in the network, called peers, offer their resources (bandwidth, processing power, storing capacity) to the other nodes, basically because they all share common interests (through the considered application). As a consequence, as the number of customers increases, the same happens with the global resources of the network. Moreover, the peers that serve another peer can be chosen with a proximity network criterion, avoiding bottlenecks, even more efficiently than with CDN servers. To base the scalability on the *network effect* property seems a good idea due to the high costs in terms of bandwidth of video distribution [320]. On the negative side, the main drawback is that peers connect and disconnect with high frequencies, in an autonomous and completely asynchronous way. When a node leaves the network (we will refer to this situation as a *failure*), it causes the loss of the information it was sending to someone else. The resources of the network as a whole are then highly dynamic, and this leads to a serious constraint at the control level, because the network must be robust face to these fluctuations. This is the main challenge in P2P design: to offer the quality needed by the clients in a highly varying environment.

Some commercial P2PTV networks for video distribution are available. The most successful are BBC: iMP [27] based on the VeriSign Kontiki software [192], Joost [185], PPlive [260], PPstream [261], SopCast [312], TVAnts [325], and TVUnetwork [326].

Besides the commercial networks some academic studies are SpreadIt [80], CoopNet [246] and SplitStream [48].

The challenges on a P2P network design are different according to the type of video service deployed. Video on Demand (VoD), for instance, has similar characteristics as popular P2P systems for file sharing and distribution. However, real–time video streaming (live TV) has different and strong constraints that imply a series of specific technical problems difficult to satisfy in a very dynamic environment. In the next subsection we provide a short introduction to P2P technology and discuss current approaches to live TV service.

2.3.3 Peer-to-Peer (P2P) Networks

Peer-to-Peer networking are the class of systems and applications that use distributed resources to accomplish some critical function in a decentralized form. The resources can be: processing, storage (particular content) and / or transmission (bandwidth). The critical functions are: distributed computation, shared information, collaboration between peers, or any type of services that can be offered by a system with these characteristics.

A series of definitions co-exist with regard to P2P networks (see [216] for a discussion). For the rest of this document we follow the definition given by Clay Shirky [308]: a **Peer-to-Peer (P2P)** Network is a Content Network that takes advantage of the free resources available in the Internet edges (i.e. the final users).

P2P networks are a particular case of Content Networks, the difference appears in some characteristics of the architecture and of the nodes of the network:

- First, P2P networks are composed by peers with a high level of *autonomy*. In particular, nodes should have the possibility of connecting and disconnecting to the network at any time in a very flexible way. In addition, in general every node determines the quantity of resources that it wants to share.

- Considering that the nodes can enter freely to the system to share their resources, an incentive is necessary for this participation. This incentive is in general achieved because in a P2P network, the peers share a common aim or a common need. Therefore, there is an implicit *desire to share* resources naturally arising in this type of systems.

- Another consequence of the autonomy is the *anonymity*. Often, the sources and the receivers of a content do not want to be known (for example for legal reasons when the content can be subject to copyright), or in other cases simply because this knowledge is completely unnecessary.

The first classification of the P2P networks that appears in the literature is on the basis of the type of application of the network [215, 216]:

- **content sharing**: file delivery, multimedia distribution, distributed storage, caching, information management (discover, aggregate, filter, mining, organize, etc.), etc;

- **collaboration**: communication (chat, instant messaging), co-review/edit/author/create, gaming, social networks, etc;

- **distributed computing**: Internet/Intranet distributed computing, etc. Its applicability includes the extraterrestrial life search, the genome project, market analysis, financial risk analysis, demographic analysis, etc.

An excellent survey about this classification of Peer-to-Peer systems is [216].

A classification based on the application is not sufficient to understand the different types of designs, architectures and algorithms that coexist in the P2P world. In our previous work [274] we present a taxonomy of Peer-to-Peer networks, based on their architectures. Similar works are [15, 32, 209]. Next, we resume our taxonomy based on two main characteristics, self-organization and decentralization.

2.3.3.1 Overlay Network: the control and routing layer

In a P2P network there are two types of exchanged data: the content itself (i.e. files, videos, etc.) and the control and routing messages (i.e. peer' connections/disconnections, content publications, searches, etc.). Each P2P network has its own method to exchange each type of data. As in Content Networks, the design problem behind the content exchange is called *content distribution*, and the design problem behind the control messages exchange is called *content discovery*.

With respect to the *content discovery*, a P2P network has to transparently present the contents to the end users, i.e. to present a consistent global view of the contents. In order to reach this objective, some control and routing messages have to be exchanged between peers to discover the content location in the most effective and efficient possible way.

With the control information exchanged a *knowledge network* is constructed. The knowledge is the information about the location where (in each peer) specific content is to be found. The knowledge network is usually called overlay network, and is defined as follows. The **Overlay Network** is a directed graph. The nodes are the peers. If a participating peer knows the location of another peer, then there is a directed arc from the former node to the latter.

Self-organization. There is a common P2P classification based on how the overlay is constructed [47, 211]. The self-organization of a P2P network can be structured or unstructured.

- **Unstructured.** The nodes of an unstructured overlay topology are organized arbitrarily (non deterministically), using flood (broadcast or random walks) for the content discovery. The content searches are propagated among the peers following the overlay topology. The participating nodes are maintained together using keep-alive messages. Usually these networks do not scale well because of the high amount of signalling traffic. On the other hand, they present a very high level of anonymity. The most known example of these networks is Gnutella [115].

- **Structured.** A structured organization network uses a globally consistent protocol to route a search of other node or content. In general, they consist in some variant of a Distributed Hash Table (DHT) technology. These networks construct a structured overlay where each node maintains a specific set of contents (or a set of content location indexes). Therefore the content searches are deterministic and efficient. In structured overlay networks, the changes of available peers and available contents are published, a difference with respect to unstructured overlay network where peers must look for changes. In general the protocols of these networks ensure to discover a specific content crossing at most a quantity of nodes logarithmic in the size of the network. In very dynamics networks it is possible that it is not efficient enough when maintaining the information about changes, compared to unstructured approaches. Examples are Kademlia [214], CAN [266], Chord [316], Pastry [286, 289] and Tapestry [364].

Decentralization. There are three possible roles of a peer in a P2P network. The nodes that have contents are source nodes, the nodes that demand these contents are requester nodes, and the nodes that exchange control messages to present a global view of content locations in the network are router nodes. A single computer may be source, requester, and router node, at the same time.

In Content Networks, the decentralization is determined by the capacity of the network to work with several nodes of the same type (sources, requesters and routers). Considering the peers' desire to share in P2P networks, always there are always many sources in these networks, limiting the concept of decentralization to the requesters and (principally) to the routers.

From the decentralization point of view, P2P networks are classified in the following types [359]:

- **Pure.** It expresses faithfully the philosophy of a P2P network, where all the nodes play equal roles (source, requester and router) and have the same responsibilities. It is a completely distributed network, which implies that the network can work with an arbitrary quantity of nodes of each class, for example the requesters can ask for a content to any node, and this node has to know how to route to some source. Examples are Gnutella [115] and Freenet [60, 322].
- **Hybrid.** The centralized model receives the name of hybrid, because it is not a pure P2P system. In this case there is a central node (or a server farm) that routes all the control messages in the network. The requesters ask for a content to the central server, who informs the sources of this content. The sources distribute the content in a P2P basis to the requesters. An hybrid network can have one requester or many requesters. For example Napster [43, 230] has a unique router node, many sources and many requesters, whereas Seti@home [305, 347] has a unique router node, many sources of processing resources, and only one requester of this processing power.
- **Hierarchical.** Between the pure model and the hybrid model, the hierarchical model has several router nodes. In file sharing applications, hierarchical networks are called super-peer networks. In these networks there are source/requester/router nodes called super-peers and exclusively clients nodes (source/requester) called peers. The super-peers form a backbone network of control messages, and present a global view of the content in the network to themselves and to the peers (with less resources than super-peers).

  This model has the potential of combining the efficiency of the content searches of the centralized hybrid systems with the autonomy, load balancing and lack of an unique point of failure of the completely distributed pure systems [358, 359]. The most known example of this type of networks is KaZaA [190].

Usually pure networks do not scale well, because of the peers' heterogeneity in connection time and shared resources [186, 272, 300, 302, 361]. By definition there are no structured hybrid networks. When the unstructured networks grew in quantity of nodes, arose the need of a model who was contemplating better the peers' heterogeneity (in connection time and shared resources), i.e. the hierarchical model. Today, the hierarchical approach is quite frequent.

Table 2.1 resumes our taxonomy of P2P architectures (with respect to two main characteristics: the self-organization, and the decentralization) with several P2P networks examples. Observe that, as far as we know, there are no structured hierarchical architectures (we suppose because of its recent deployment).

2.3.3.2 Structured Overlays for Live Video Streaming

The most popular P2P systems are those designed for file sharing. Important names are KaZaA [190], Bittorrent [30], or eMule [89], all using almost no dedicated infrastructure. The

Table 2.1: Taxonomy of P2P networks architectures with several P2P networks examples.

		Decentralization		
		Pure	Hierarchical	Hybrid
Self-organization	Unstructured	Gnutella, FreeHaven	KazaA, eMule, openNap	Napster, Seti@home, Avaki, Genome@home
	Structured	FreeNet, Kademlia, SplitStream(Pastry), OceanStore(Tapestry)		

common feature of this type of system is the basic underlying hypothesis that peers remain connected for some time to the network. For instance, [299] reports a median session duration of about one hour both for the famous Napster [230] and Gnutella [115]. Based on this assumption, many structured P2P networks have been developed for file sharing. Using the idea of Distributed Hash Tables (DHT), the most successful ones are Kademlia [214], CAN [266], Chord [316], Pastry [286, 289] and Tapestry [364].

Live video P2P networks have to satisfy harder constraints, because video reproduction has real–time restrictions (file sharing don't)[6], and the nodes only remain connected a few minutes [314]. For designing purposes, in our work we follow a video delivery reference service (presented in Section 2.5) where the clients are connected fifty minutes on the average.

Short surveys about structured overlay P2P networks for live video distribution are [12, 303, 321]. We use here some of the concepts presented by them.

Tree-based Overlays. The first idea to use the idle resources available at the final users is to make a chain of streaming proxies. A brodcaster (initial publisher of the live signal) sends the stream once to a selected peer. This peer selects another peer and relays the stream, and so on. A long chain of peers relaying the stream one to each other theorically covers the distribution needs, but it is impractical because of the peers' dynamic and peers' resource heterogeneity. When a peer disconnects from the network (a very frequently event), the whole tail of the chain loses the stream while the child of the leaving node remains disconnected. Moreover, if a peer does not have enough upload bandwidth to stream the signal, it can not participate in the network.

The idea of tree-based distribution arise to diminish the length of the chain, and therefore the probability of predecessor disconnections.

Tree-based overlays implement a tree distribution graph, rooted at the broadcaster, where the peers relay the stream to, possible more than one, other peers. In a tree, the depth (longest chain from the root) can be logarithmic with respect to the number of nodes in the network

[6]A peer disconnection from the network can generate some losses in other peers that receive the content from them, until the structured network is reconstructed. But this is perceived completely different by final users, depending on the kind of content. Live streaming is especially sensitive to losses, whereas in file downloads, losses will only (slightly) increase the delay, with a very limited impact on the quality as experienced by the user.

(instead of linear as in the chain). In addition, in a tree distribution there are many leaves that can receive the stream without the need of an upload bandwith capability. On the other hand, in a tree distribution some peers need to send the stream more than once, implying greater resources contribution at the peers side.

Earliest tree-based overlay systems were proposed to overcome the limited deployment of IP multicast on Internet. Called Application Level Multicast (ALM), they have faster reconstruction algorithms, ideal for high dynamic scenarios. ALMs are proposed for live streaming (e.g., ALMI [255], End System Multicast [90, 135], Scattercast [52]), and they report success in controlled environments [134]. Each ALM based system has its own protocol for building and maintaining the multicast tree. For example, both NICE [25] and Zigzag [324] adopt hierarchical distribution trees and therefore scale to a large number of peers. Narada [136] targets small scale multisource-multirequester applications. Narada maintains and optimizes a mesh of peers first and then construct a tree on top, that yields good performance but imposes significant maintenance overhead. SpreadIt [80] constructs a distribution tree rooted at the source for a live media streaming session. A new requester joins by traversing the tree starting at the root, until it reaches a node with sufficient remaining capacity.

Specific multicast applications for live streaming improve the overlay designs with codec-specific knowledge. Usually, in these networks, the broadcaster encodes the multimedia stream into many sub-streams using Multiple Description Coding (MDC) [16, 118, 191, 204, 304]. This allows to construct multiple trees distribution (one tree for each description or sub-stream), providing more flexibility in resources allocation and more resilience to failures. CoopNet [119, 243–246] supports both live and on-demand streaming. It constructs multiple distribution trees (using MDC) spanning all participants. It implements a hybrid P2P system, with a centralized algorithm to build the trees: when the video server is overloaded, clients are redirected to other nodes, thereby creating a distribution tree routed at the server. SplitStream [48] provides a cooperative infrastructure that can be used to distribute large files (e.g., software updates) as well as streaming media, in a completely decentralized form. SplitStream is built on top of Scribe [46], a scalable publish-subscribe system that employs Pastry [286] as the lookup substrate. The content in SplitStream is divided into several stripes, each distributed by a separate tree, this allows to freely manage the bandwidth contribution of each peer. Different from these systems, CollectCast [126, 127] is not intended for multicast. As a complementary P2P service, CollectCast is proposed for media streaming from multiple sources to one requester.

Like commercial CDNs deployments, some companies offer content providers a live video delivery service using a tree-based multicast overlay infrastructure. For instance, Allcast [11], and PeerCast [252].

Tree-based structured overlays for live video streaming have efficient tree construction algorithms in contrast with the overheaded file sharing distribution protocols. But the tree-based approach perform well when the peers have low dynamic, in other case the tree is continuously broken and reconstructed, and it equates in overhead with the file sharing protocols.

Mesh-based Overlays. Most used file sharing applications implement a mesh distribution graph. A list of the peers currently downloading each file is maintained by a central Web server (called the tracker) in Bittorrent [30], or by a distributed number of servers (usually by the Lugdunum software [210]) in eMule [89], or by peers with high availability (called super-peers) in KaZaA [190]. Using this list, each peer knows, at any time, a subset of other peers that are downloading the same file (called the peer swarm). The file is distributed in pieces (called chunks). Every peer knows which chunks are owned by the peers in its swarm, and explicitly pulls the chunks that it requires. In Bittorrent, peers request the chunks which fewest number of peers have (*rarest-first* download policy), and peers upload chunks first to the peer requesters that are uploading data to them at the fastest rate (*tit-for-tat* upload policy). Other networks have similar incentive policies to avoid free-rider peers.

Mesh-based overlays are starting to be used for live video distribution. File sharing applications cannot be used directly for live video distribution, because the first chunks of the file are not downloaded first, and therefore the file cannot be played until the download has been completed. But not only the *rarest-first* download policy has to be changed for use in live streaming: pull mechanisms involve overhead, that implies large buffers at the client side (to increase the probability of finding a chunk), and therefore very high delays.

On the positive side, mesh-based overlays offer good resilience to peer failures, and they out-perform conventional tree-based overlays because every peer is a source (no matter its upload bandwidth capabilities). In essence, without considering protocols details, mesh-based distribution can be seen as a distributed tree-based mechanism for each chunk.

Besides a few academic studies [78, 321, 336, 351, 352], some commercial mesh-based networks for live video distribution are available. The most successful are PPlive [260] with more than $200,000$ concurrent users on a single channel [303] (reaching an aggregate bit-rate of 100 Gbps), SopCast [312] with more than $100,000$ concurrent users reported by its developers, CoolStreaming [68] with more than $25,000$ concurrent users on a single channel [362], PPstream [261], TVAnts [325], and TVUnetwork [326].

PPLive [260] is the most popular P2PTV software, especially for Asian users. Born as a student project in the Huazhong University of Science and Technology of China, PPLive uses a proprietary protocol, and its source-code is not available. With reverse engineering, the works [10, 128] show that it uses a mesh-based approach. Their protocol is very similar to the Bittorrent protocol, with a channel selection bootstrap from a Web tracker, and peer download/upload video in chunks from/to multiple peers. It uses two kinds of video encoding formats, Window Media Video (VC-1) and Real Video (RMVB), with average bit rates from 250 Kbps to 400 Kbps, and it uses the user native media player to display the video. The distribution engine (Bittorrent-like protocol) does not use the *rarest-first* download policy, because the streaming must satisfy real–time restrictions. Also the *tit-for-tat* upload policy is not applied: when a client joins a channel, the other peers in the swarm send chunks forcefully to minimize the playback startup delay. PPLive has two buffers, one on the distribution engine (to ensure the chunk arrival in time), and the media player buffer. After a user selects a channel, he has to download the list of peers, contact them, etc. This bootstrap delay is around $10 \sim 15$ seconds. After that, approximately $10 \sim 15$ seconds are needed to fill the buffers, and the playback starts. Therefore, the total startup delay is $20 \sim 30$ seconds approximately (this varies according to the channel's popularity, and for less popular channels the total startup delays can increase up to 2 minutes). The peering selection (i.e the selection of peers to pull the chunks that a peer requires) uses a greedy algorithm completely independent of the underlying Internet network, which implies higher costs to ISPs and more probable congestions.

SopCast [312], created at the Fudan University of China, is very similar to the PPLive project. Following the same reverse engineering procedure, [10, 128] show that SopCast uses a mesh-based approach, with a proprietary protocol very close to the PPLive protocol. A complete performance study of this network is [303].

CoolStreaming [68] (also known as DONet) is the predecessor of PPLive and SopCast, now forced to close down due to copyright issues. Also using a proprietary protocol, and closed source-code, as PPLive and SopCast, the CoolStreaming authors published information about its distribution engine [353, 362, 363]. The video stream is divided into chunks with equal size. A difference of CoolStreaming, with respect to PPLive and SopCast, is that the video stream is decomposed into six sub-streams by a rotating selection of video chunks[7]. Another difference is that CoolStreaming uses a locality-aware strategy for the peering selection. Nowadays, CoolStreaming is the base technology for Roxbeam [290], supported by SoftBank Japan; they have live channels jointly with Yahoo Japan.

Anysee [206] is a mesh-based overlay, but the peers do not pull chunks from its peers list indiscriminately. In AnySee, peers maintain a mesh-based overlay that match with the underlying physical topology. After a bootstrapping, each peer periodically floods messages to select a list of logical neighbors that are close to him. The streaming delay from the video source to the node is evaluated for each peer in the list, and a set of active streaming path is constructed. Each peer maintains one active streaming paths set and one backup streaming path set. When one active streaming path is failing (due to its poor bandwidth or peer's disconnection), a new streaming path is selected from the backup set. When the number of backup streaming paths is less than a threshold, an inter-overlay optimization algorithm is called to find appropriate streaming paths (flooding messages in the mesh overlay). Due to the inter-overlay optimization, some experiments [206] show that Anysee has smaller buffers than the chunk-based schemes (with average delay between source and destination less than 30 seconds).

Some companies offer content providers a live video delivery service using a P2P mesh infrastructure. The most successfull ones are Abacast [2], Rawflow [183, 267] and Octoshape [12, 238]. They report about more than 97% of bandwidth saving, if the clients have an average upload capacity as large as the live stream.

In summary, tree-based overlays have efficient transmission, because the data goes from parent to children (and does not flood the network). But these systems have low stability, because of their sensibility to the nodes' failures. On the other hand, peers in a mesh-based overlay have to maintain a buffer with large size for data sharing, increasing the global delay and the player startup. There are some proposals [137] of tree-mesh based hybrid overlay networks, that try to exploit the advantages of the two approaches. Table 2.2 resumes the P2P architectures studied in this section. Observe that there is no open-source architecture between

[7]Chapters 6 and 7 are dedicated to improve this technique. In a context of peers' failure, we optimally divide the video stream in sub-streams to guarantee a perceived video quality level.

them.

Table 2.2: P2P network examples of architectures & business models.

		Structured Self-organization	
		Tree	Mesh
Business Model	Advertising, Infomediary, or Undefined	SpreadIt, CoopNet, SplitStream, CollectCast	PPLive, SopCast, CoolStreaming, AnySee
	Subscription, Brokerage	AllCast	AbaCast, Rawflow, Octoshape

2.3.3.3 Content Delivery: the data download from the client point of view

In most file sharing P2P systems, the data is downloaded using Bittorrent-like protocols, where each peer maintains a list of neighbors from where he downloads/uploads the file in chunks. Video on demand can be distributed in a similar way. These methods imply that a client receives the data from multiple sources simultaneously.

Live video streaming can be distributed, in a P2P network, from multiple sources or from one single source[8]. No matter if the network is based on a tree overlay or a mesh overlay, when multiple sources are used, the clients receive several redundant flows from different peers, allowing them to reconstruct the original stream. This way, if some of the pieces don't arrive because of failures of some nodes, the client doesn't loose, in general, the whole stream. Some keywords associated with this approach are Multiple Description Coding (MDC) [16, 118, 191, 204, 304] and Network Coding [6, 55, 56] (we explain it in depth later, in Subsection 6.1.2).

Therefore, it is possible to see the whole system from the final user point of view. For instance, Figure 2.6 presents a client that reconstructs the stream from three redundant sources. The client can also act as a server (not shown in the picture), and send partially or fully the received video to other peers. In a Peer-to-Peer network these servers are also peers that frequently disconnect from the network or change their bandwidth capabilities.

Studing the system from the client point of view allows us to quantify the global performance (including the final target: the perceived video quality), independently of the underlying particular distribution technique. Part III of this document studies the design of a Peer-to-Peer distribution system from this perspective. This approach also allows for a large flexibility of the system, modulated by the dynamics of the network. In particular, it is possible to increase the number of substreams composing the original flow and/or the amount of redundant information sent through the network; this is used as a tool to deal with the problem of nodes leaving the network and causing partial signal losses to some clients. We can also control the way the load is distributed among the different substreams.

[8]Some networks use a single source because it is the simplest way to manage the complexity of the synchronism and the real-time restrictions.

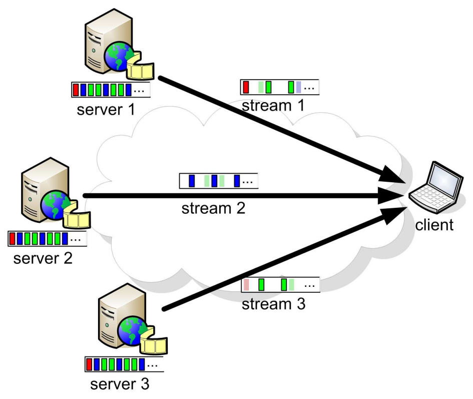


Figure 2.6: Client point of view of a streaming method with three servers.

2.3.4 Introduction to Network Robustness: an hybrid P2P approach

As said before, is possible to combine the advantages of Content Delivery Networks (CDN) architectures and Peer-to-Peer (P2P) networks in a hybrid system. The main idea is to extend the resources available in the CDN with the efficiency of P2P delivery, keeping the high quality, security, and centralized control of the CDN.

Suppose we have a live video delivery service with a streaming encoding rate of $bw = 500$ Kbps. We expect to have $N = 100.000$ simultaneous users in some popular events. Therefore, we need a total bandwidth of 50 Gbps in the network to support this service. Today, the average upstream bandwidth of residential users is $\overline{BW^{out}} = 400$ Kbps[9]. Suppose that each user provides 90% of its upload bandwidth to the network. With a hybrid system, we need a bandwidth capacity of $Nbw - 0.9N\overline{BW^{out}} = 14$ Gbps. Instead, to serve the same number of users with a traditional CDN architecture, we have to increase our bandwidth capacity in an additional 36 Gbps, or we have to degrade the streaming quality to 140 Kbps per user.

As in the P2PTV networks, in the hybrid approach the bandwidth is dynamically allocated, reducing cost and avoiding local congestion or bottlenecks (for contents that are popular in a local area).

Our GOL!P2P prototype for live video streaming (presented on Chapter 10) is a **hybrid** P2P system, with central control and distributed delivery. The central control maintains a simple tree-based **structured** P2P overlay network, while a **multi–source streaming technique** is used for a P2P delivery.

The robustness of this solution with respect to the resources dynamics is the most important design element if we want to guarantee some level of video quality.

In the previous section we introduced the robustness concept from the client point of view.

[9]Speed-test results from the popular `http://broadbandreports.com` reports an average residential U.S. upstream bandwidth of 371 Kbps (and a downstream bandwidth of 1.9 Mbps) on May, 2007. Speed-tests are TCP-based bandwidth measurements that are conducted from well-provisioned servers distributed on Internet. Users voluntarily test their connection speeds, and the results are summarized, over a week, based on DNS domain names or ISPs.

In this section we extend the robustness concept with the network point of view. This introduction must help to understand the rest of the document. Our design mitigates the impact of the peer' disconnection in two levels. In the long term the system places the peers most committed to the network (those with largest lifetime and lowest bandwidth fluctuation) near to the broadcaster to obtain a more robust overlay topology. This dynamics is formalized as a mathematical optimization problem and solved in Chapter 8. In the short term, and known which peers will serve the streaming to a client, we study the best streaming technique that mitigates the peers' disconnection (Chapter 7). So, both much in the long and short terms, the explicit aim is the same: to improve the perceived average video quality of end users.

2.3.5 Summary: Conclusions, Results and Discussion

In this section we presented the Content Delivery Networks (CDN) and the peer-to-peer (P2P) networks as particular classes of Content Networks. We showed the advantages and disadvantages of each architecture when they are used for video delivery on the Internet. We presented the state of the art of P2P network design, deepening in the video delivery case, i.e., the P2PTV networks. Two main designs coexist for P2PTV networks: tree-based overlays and mesh-based overlays. Without considering the distribution design, two factors are inalterable. First, it is possible to see the video distribution from the client point of view, where the client receives the video from a set of sources, with some redundancy level, and with some failure probability. Second, it is possible to combine the good properties of the CDN architecture with the efficient P2P distribution in an hybrid system. The obtained hybrid system bring a service more scalable and robust than each architecture isolated. Our P2PTV prototype, GOL!P2P, follows this idea, with a centralized control and a tree-based overlay video distribution.

2.4 Search Caching: an approach for discovery scalability in Content Networks

In the previous section (Section 2.3) we presented an overview of the content delivery mechanisms used in Content Networks, with an in depth study about the live video delivery. Besides of content delivery design, other very important architectural design area is the content discovery.

As we have previously defined, in a content network the addressing and routing are based on the content description, instead of on its location. This means that every content network is actually a *knowledge network*, where the knowledge is the information about the location of the nodes where each specific content is to be found. This is *meta-information*, in the sense of being the information about the information contents itself. The knowledge network is usually called *overlay topology*.

An objective of the network is to be able to found the most complete possible set of nodes where a content is lodged. This corresponds to discovering the content location in the most effective and efficient possible way. There are two main strategies to discover the meta-information, namely *publication* and *search*. By *publication* we mean the process by which a network node unrequestedly sends meta-information it possesses to the remaining nodes. By *search* we mean the process by which a node asks the remaining ones to send it the meta-information they possess. By analogy with logistics, we can say that publication is an "information push" strategy, and search an "information pull" strategy. As both nodes and contents are continuously going in and out of the network, the task of maintaining updated the network meta-information is very difficult and represents an important communication cost. Both publishing and search can contribute towards this task, but their relative efficiency varies, so that there is a tradeoff between their frequency and modality of application. As we say, in any content network there are three kind of nodes:

- *source nodes*, that offer some content to the rest of the nodes;
- *requester nodes*, which want to obtain some content from the network;
- *router nodes*, which exchange control messages with other nodes, in order to coordinate the discovery of the content in the network, and to help to the requesters to find sources that could answer to they orders.

A single computer may be source, requester, and router node; and typically they are source and requester at the same time. The scalable networks have an specific kind of router nodes, the *cache node*. Cache nodes are used to hold the available meta-information; as this information is continuously getting outdated, the cache nodes must decide when to discard it, which means increasing communication overhead for the sake of improving the quality of the answers. In the simple network architecture, source nodes connect to cache (also called aggregation) nodes and "publish" their contents. Request nodes query to aggregation nodes and ask for content locations; afterwards, for the delivery, they connect directly to source nodes. Router nodes connect the aggregation nodes to present a global view of content locations in network.

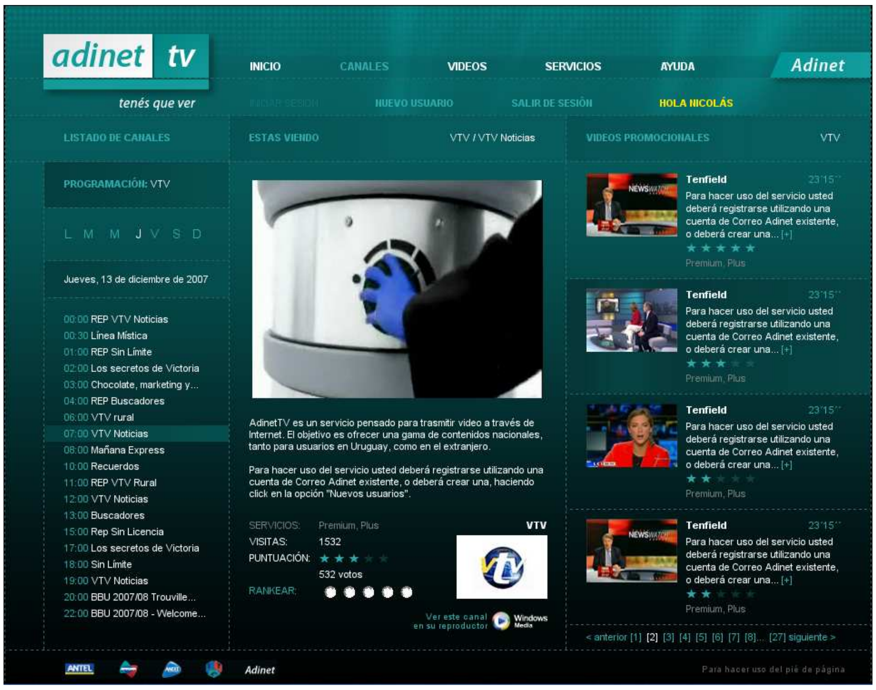
The content discovery is study in depth in chapters 8 and 11. In Chapter 8 we present the overlay topology used in our GOL!P2P prototype for the live video streaming service. In Chapter 11, we develop a simplified model of a content network, and in particular of the number of correct answers to a query as a function of the information expiration times used at the cache nodes. This model gives rise to a mathematical programming formulation, which is used to find the expiration times maximizing the correct answers to the queries received from two additional services, planned in our prototype, the VoD service and the MyTV service.

2.5 A Video Delivery Reference Service

AdinetTV[10] is a live video service of a medium-size ISP, with focused content for Uruguayan people. Figure 2.7 shows how adinetTV looks like.

AdinetTV has a Content Delivery Network (CDN) architecture, with tens of servers and a total upload bandwidth of 2 Gbps. The streaming rate goes from 100 Kbps to 350 Kbps depending on the number of simultaneous users (therefore, the network accepts from 20.000 to 5.500 simultaneous users). For the moment, AdinetTV has 60.000 subscribed users. The frequency of user connection varies in a wide range. Figure 2.8 shows the number of connection per most active users in a 4 months period. The users connect to the system from Uruguay (35%

[10] `http://www.adinettv.com.uy.`



adinet tv

tenés que ver

INICIO CANALES VIDEOS SERVICIOS AYUDA Adinet

INICIAR SESIÓN NUEVO USUARIO SALIR DE SESIÓN HOLA NICOLÁS

LISTADO DE CANALES

PROGRAMACIÓN: VTV

L M M J V S D

Jueves, 13 de diciembre de 2007

00:00 REP VTV Noticias
00:30 Línea Mística
01:00 REP Sin Límite
02:00 Los secretos de Victoria
03:00 Chocolate, marketing y...
04:00 REP Buscadores
06:00 VTV rural
07:00 VTV Noticias
08:00 Mañana Express
10:00 Recuerdos
11:00 REP VTV Rural
12:00 VTV Noticias
13:00 Buscadores
15:00 Rep Sin Licencia
17:00 Los secretos de Victoria
18:00 Sin Límite
19:00 VTV Noticias
20:00 BBU 2007/08 Trouville...
22:00 BBU 2007/08 - Welcome...

ESTAS VIENDO VTV / VTV Noticias

AdinetTV es un servicio pensado para trasmitir video a través de Internet. El objetivo es ofrecer una gama de contenidos nacionales, tanto para usuarios en Uruguay, como en el extranjero.

Para hacer uso del servicio usted deberá registrarse utilizando una cuenta de Correo Adinet existente, o deberá crear una, haciendo click en la opción "Nuevos usuarios".

SERVICIOS: Premium, Plus
VISITAS: 1532
PUNTUACIÓN: ★★★
532 votos
RANKEAR:

VTV

Ver este canal en su reproductor Windows Media

VIDEOS PROMOCIONALES VTV

Tenfield 23'15''
Para hacer uso del servicio usted deberá registrarse utilizando una cuenta de Correo Adinet existente, o deberá crear una... [+]
★★★★★
Premium, Plus

Tenfield 23'15''
Para hacer uso del servicio usted deberá registrarse utilizando una cuenta de Correo Adinet existente, o deberá crear una... [+]
★★★
Premium, Plus

Tenfield 23'15''
Para hacer uso del servicio usted deberá registrarse utilizando una cuenta de Correo Adinet existente, o deberá crear una... [+]
★★
Premium, Plus

Tenfield 23'15''
Para hacer uso del servicio usted deberá registrarse utilizando una cuenta de Correo Adinet existente, o deberá crear una... [+]
★★
Premium, Plus

< anterior [1] [2] [3] [4] [5] [6] [7] [8]... [27] siguiente >

ANTEL Adinet

Para hacer uso del pié de página

Figure 2.7: AdinetTV: our video delivery reference service to design GOL!P2P.

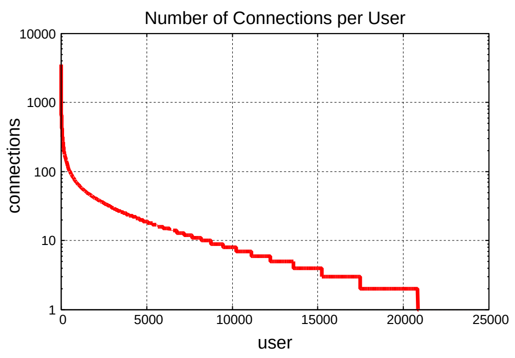
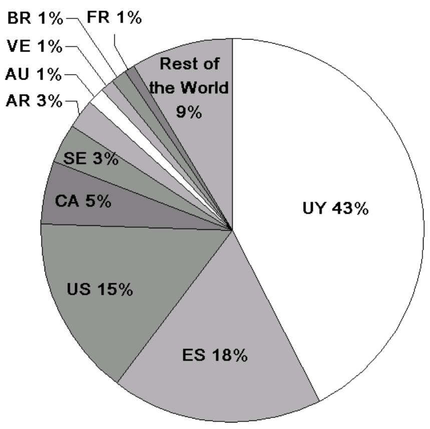
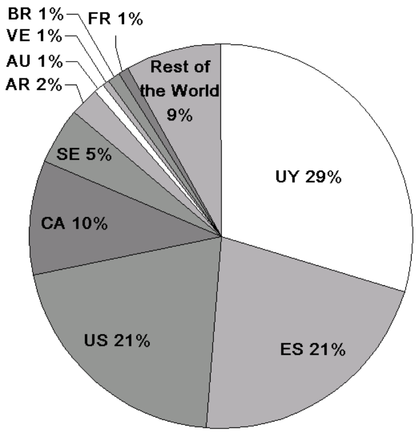


Figure 2.8: Number of connection per user, in a 4 months period. Real data of a live-video delivery reference service from a medium-size ISP. Only the 25.000 more active users are shown.

approximately), and the rest from all around the world (principally where large Uruguayan communities are located, like Spain and USA). Figure 2.9(a) shows the percentual number of connections from different contries, while Figure 2.9(b) shows the percentual consumed time from each country. During popular events, like football matches, AdinetTV has peaks of



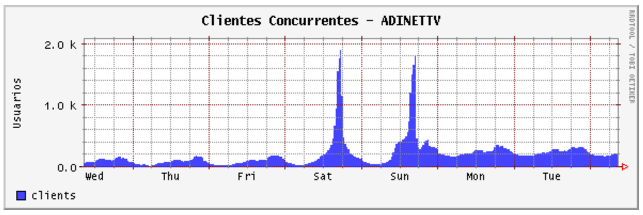
(a) Percentual number of connections.



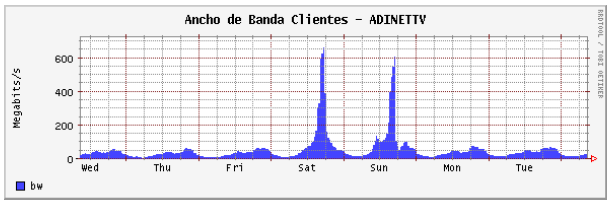
(b) Percentual accumulated connection-time.

Figure 2.9: Geographical use of AdinetTV.

4.000 simultaneous users. The Figure 2.10(a) shows the simultaneous users in a week, and the Figure 2.10(b) shows its bandwidth consumption.



(a) Simultaneous Users.



(b) Bandwidth consumption.

Figure 2.10: Statistical use of AdinetTV.

We want to extend AdinetTV with a P2P delivery system, improving it in scalability (more simultaneous users), and in quality (larger streaming rates).

We analyze the streaming workload from AdinetTV, to better understand the design requirements for building a live streaming system. In the rest of the work, we use the workload of this reference service to evaluate the performance of our design. We focus our analysis on the characteristic that are most likely to impact design, such as peer dynamics. We collected 4 months of AdinetTV streaming server logs, in order to estimate the distribution of connection/disconnection frequencies and session life-times. Figure 2.11 shows the peers' life-time distribution in our video delivery reference service, for a given number of users x, the curve gives $F(x) = y$ meaning that x users have connection time $\geq y$. Since in the proposed architecture we suppose that the servers are P2P nodes, it is reasonable to assume that the mean-life of the users will also correspond to the expected sojourn time of the servers in the network.

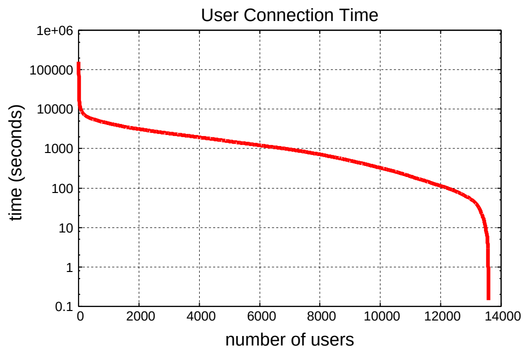


Figure 2.11: User life-time distribution. Real data of a live-video delivery reference service from a medium-size ISP. For a given number of users x, the curve gives $F(x) = y$ meaning that x users have connection time $\geq y$.

2.6 Summary

In this chapter, a digital video overview is presented, with the needed depth in video compression and streaming technologies; in order to understand the rest of this work. We also gave a video delivery network classification, with some level of details in the factors that impact the quality-of-experience (QoE) in each kind of network. We presented the state of the art of Content Delivery Networks (CDN) and Peer-to-Peer (P2P) networks, zooming into the video delivery techniques on the Internet. It is possible to combine the good properties of the CDN architecture with the efficient P2P distribution in an hybrid system. The obtained hybrid system offers a service more scalable and robust than in each architecture alone. Our P2PTV prototype, GOL!P2P, follows this idea, with a centralized control and a tree-based overlay video distribution.

Part II

QUALITY

Chapter 3

Video Quality Assessment: State of the art

In this chapter we present the issues associated with the assessment of the quality of video streams. We discuss the evaluation mechanisms currently available in the literature, and why they don't necessarily fulfill the current needs in terms of perceived quality assessment.

There are two main approaches for measuring the video quality: *subjective assessment* and *objective assessment*. In brief, subjective assessments consist of a panel of human beings rating a series of short video sequences according to their own personal view about quality. Objective assessments stand for the use of algorithms and formulas that measure the quality in a automatic, quantitative and repeatable way.

The rest of the chapter is organized as follows. In Section 3.1, we describe several subjective quality methods and we compare the most important aspects of them. In Section 3.2, we provide a general description of objective quality metrics. Mitigating the disadvantages of both approaches, hybrid methods have been developed. In Section 3.3 we present the precedents of the PSQA methodology, that will be explained in depth in Chapter 4. Finally we provide conclusions of this chapter in Section 3.4

3.1 Subjective Quality Assessment

There is no better indicator of video quality than the one given by human observers, because the video is created to be watched by human beings. Unfortunately, the quality given by an observer depends on his/her own experience. That is why we call this view the perceived quality of the video.

Perceived video quality is, by definition, a subjective concept. The mechanism used for assessing it is called *subjective testing*. There are several types of subjective assessment techniques, which depend of the quality aspects that are evaluated and the kind of application. Two main classes of subjective assessment can be found, namely *qualitative assessment*, and *quantitative assessment*.

Qualitative assessment [36, 37] produces descriptions of an observer quality perception, and usually it does not translate well into numeric scales. This approach is useful in several contexts, for instance, to describe how different observers react to variations of the perceived quality, and to evaluate the interest in proposing pricing schemes for a multimedia service. These methods are suitable to explain the “sociologic” aspects of the quality rather than to be used in an engineering design.

On the other hand, quantitative assessment provides a more concise approach to quality assessment, and is more suitable for our work. It consists of building a panel with real human subjects which will evaluate a series of short video sequences according to their own personal view about quality. An alternative is to use a (smaller) panel of experts. In the first case, we will get the quality of the sequences as seen by an average observer. In the second case, we can have a more pessimistic (or optimistic, if useful) evaluation. The output of these tests is typically given as a Mean Opinion Score (MOS) [171]. Obviously, these tests are very time-consuming and expensive in manpower, which makes them hard to repeat often. And, of course, they cannot be a part of an automatic process (for example, for analyzing a live stream in real time, for controlling purposes).

There are standard methods for conducting subjective video quality evaluations, such as the ITU-R BT.500-11 [171]. Some variants included in this standard are[1]:

- Single Stimulus (SS),
- Double Stimulus Impairment Scale (DSIS),
- Double Stimulus Continuous Quality Scale (DSCQS),
- Single Stimulus Continuous Quality Evaluation (SSCQE),
- Simultaneous Double Stimulus for Continuous Evaluation (SDSCE)
- and Stimulus Comparison Adjectival Categorical Judgement (SCACJ).

The differences between them are minimal and mainly depend on the particular application considered. They concern, for instance, the fact that in the tests the observer is shown pre-evaluated sequences for reference (which in turn, can be explicit or hidden), the quality evaluation scale (and the fact that it is discrete or continuous), the sequence length (usually around ten seconds), the number of videos per trial (once, twice in succession or twice simultaneously), the possibility to change the previously given values or not, the number of observers per display, the kind of display, etc.

In the following we shortly present and discuss some of these methods.

3.1.1 Single Stimulus (SS)

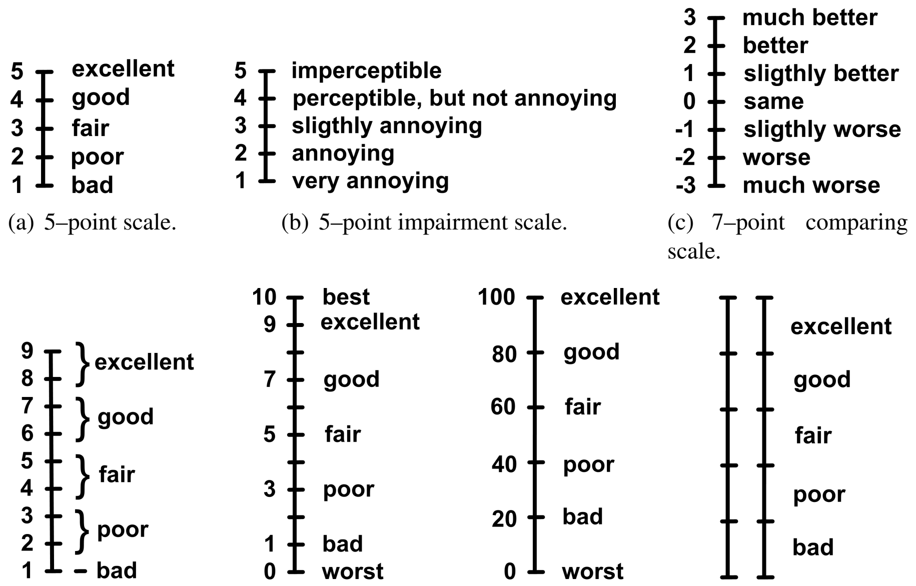
The Single Stimulus (SS) method of ITU-R BT.500-11 is also called Absolute Category Rating (ACR) in ITU-T Recommendation P.910 [174].

This method is designed to make an absolute quality assessment of audiovisual sequences. The duration of each sequence should be about 10 seconds. A gray scene (not longer than

[1]There are a set of ITU standards for other applications, such as ITU-T P.800 [173] for voice and ITU-T P.920 [175] for interactive multimedia.

500 ms) should be used at the beginning and at the end of each sequence to make it more natural. Also, the termination of a sequence should not cause an incomplete scene, and audio and video must be perfectly synchronized. After viewing a sequence, subjects are asked to rate the quality in a period that should be less than or equal to 10 seconds; this phase is called the voting time. The test sequences are presented one at a time and are rated independently on an absolute scale. There are different scales suggested in the standard, depending on the accuracy needed; the five–point scale (Figure 3.1(a)) is the most used scale in audio tests, whereas in video tests a more discriminative measure is required and a nine-point scale or eleven-point scale may be used as shown in Figures 3.1(d) and 3.1(e).

Depending on the accuracy needed, the number of subjects required by the standard vary from 4 to 40. They should not be experts in video quality and should not have work experience in this topic. Also they have to have normal vision. In our work we did a set of subjective tests based in the SS method with the eleven–point scale. For more details about how to carry out this subjective test see section 4.4.



(a) 5–point scale. (b) 5–point impairment scale. (c) 7–point comparing scale.

(d) 9–point scale. (e) 11–point scale. (f) continuous scale. (g) double continuous scale.

Figure 3.1: ITU standard scales for subjective test methods.

3.1.2 Double Stimulus Impairment Scale (DSIS)

The Double Stimulus Impairment Scale o Degradation Category Rating (DSIS) method of ITU-R BT.500-11 is also called Degradation Category Rating (DCR) in ITU-T Recommendation P.910 [174].

This method is designed to measure the quality degradation caused by some encoding or transmission scheme. The sequences are shown consequently in pairs: the first one is the reference sequence, the second one is the impaired sequence. The subjects are informed about which one is the reference high quality video, and are asked to rate the quality of the latter with respect to the former using the scale shown in Figure 3.1(b) (the scales used in SS method can also be used by replacing the quality adjectives by the corresponding impairment adjectives). The time between the reference and the impaired sequence should be less than 2 seconds, and the voting time should be less than or equal to 10 seconds.

3.1.3 Double Stimulus Continuous Quality Scale (DSCQS)

The Double Stimulus Continuous Quality Scale (DSCQS) is designed to measure the quality when the sequences only explore a sub-range of the full available range. In this situation a double stimulus is needed to help the subjects in comparing different qualities. Sequences are played in pairs simultaneously[2], one of the videos is the reference and the other one is the impaired. Subjects do not know which is the reference one, and the order is changed randomly from one pair to the next one. For each pair, subjects have to mark the quality of each sequence in a continuous vertical scale (Figure 3.1(g)).

3.1.4 Other Works in Subjective Video Quality Measures

Three other methods are presented in ITU-R BT.500-11. They are: Single Stimulus Continuous Quality Evaluation (SSCQE), Simultaneous Double Stimulus for Continuous Evaluation (SDSCE), and Stimulus Comparison Adjectival Categorical Judgement (SCACJ).

SSCQE is designed to measure the quality in a variable context, for example when an adaptative compression is used. Previously described methods measure the quality of a whole sequence, whereas SSCQE uses a continuous assessment. Where each subject constantly measures the quality with the Figure 3.1(f) scale, the average score is obtained estimating the probability distribution of each assessment.

Similar to DSCQS, SDSCE method measures two sequences simultaneously. The subjects are requested to check the impairment between the two sequences constantly, using the 3.1(f) scale (where 100 points mean identical sequences).

Similar to the DSCQS, in SCACJ two sequences are played simultaneously, and after playback the subject is asked to give his opinion using the comparison scale of Figure 3.1(c).

The ITU-R draft document 6Q/57-E [170] specifies the Subjective Assessment Methodology for VIdeo Quality (SAMVIQ). SAMVIQ is designed to evaluate the quality of multimedia in the Internet, especially video formats and streaming techniques. SAMVIQ allows particular codecs and resolutions that previous Recommendations (ITU-R BT.500) do not consider because were developed for TV application.

Sequences are present in multi-stimulus form. The subjects can choose the tests order and correct their votes. Hidden references can be used among the sequences, not only at the beginning of the test.

[2]Sometimes for simplification, the two sequences are showed in the same playback windows, given the possibility to freely switch between the two videos.

In the “Architecture and Transport” working group of the DSL Forum [86], there is a guide to implementing a test plan and to the use of video quality subjective testing with MOS Scoring in the IPTV context (working progress document WT-126 “Triple-play Services Quality of Experience (QoE) Requirements and Mechanisms” [87]). In this area, accurate video-quality measurement and monitoring is today an important requirement of industry. In chapter 9 we present an architecture to measure automatically and in real time the QoE in Video Delivery Networks (and particulary in IPTV systems).

Coming from the academic or the industry sector, some groups study specifically the assessment of subjective video quality. Some of the most important ones are the Video Quality Experts Group (VQEG) [333], the Institute für Rundfunktechnik [164], and the Graphics & Media laboratory of Computer Science department of MSU [120]. Part of this Section is inspired by their works.

3.1.5 Subjective Test Comparison

The presented *Subjective* quality assessment methods principally differ in their objectives. Single Stimulus (SS) is designed to make an absolute quality assessment of audiovisual content. To measure the quality degradation caused by some encoding or transmission scheme the Double Stimulus Impairment Scale (DSIS) is more suitable. The Double Stimulus Continuous Quality Scale (DSCQS) can measure the quality when the sequences only explore a sub-range of the full available range. Single Stimulus Continuous Quality Evaluation (SSCQE) is designed to measure the quality in a variable context, for example when an adaptative compression scheme is used. Simultaneous Double Stimulus for Continuous Evaluation (SDSCE) checks the impairment between two sequences, and Stimulus Comparison Adjectival Categorical Judgement (SCACJ) compares two sequences that are played simultaneously. Closer to our application there are: the Subjective Assessment Methodology for VIdeo Quality (SAMVIQ) designed to evaluate the quality of multimedia in the Internet, and the work-in progress document WT-126 “Triple-play Services Quality of Experience (QoE) Requirements and Mechanisms” of the DSL Forum, to measure QoE in IPTV networks. Based on [194], Table 3.1 summarizes the main differences between subjective test methods.

All the presented subjective quality assessment methods measure directly the perceived quality from the user’s perspective. However, subjective tests are very time-consuming and expensive in manpower, which makes them hard to repeat often. Furthermore, given their nature, they are obviously not suitable for real-time operation (which is our final goal in this work). Moreover, the standards (except the draft SAMVIQ [170]) are designed for broadcast TV application, and they do not fit well with multimedia streaming in a best–effort network, like the Internet. In this sense, they do not fit well to our application because: they were designed for applications which normally have better quality levels [344, 345], with the use of short samples, separating audio and video, it is not possible to capture the relative importance of video and audio in the overall quality [125, 327], and they have numerous strict parameters ranging from room size to equipment calibration that make the test more expensive.

Table 3.1: Subjective Test Comparison.

Parameter	SS	DSIS	DSCQS	SSCQE	SDSCE	SAMVIQ
ITU-R Standard	BT.500	BT.500	BT.500	BT.500	BT.500	BT.700
Scale	discrete	discrete	continuous	continuous	continuous	continuous
(Figure)	3.1(a), 3.1(d) or 3.1(e)	3.1(b)	3.1(g)	3.1(g)	3.1(g)	3.1(g)
Sequence Length	10 seg	10 seg	10 seg	$\geq$ 5 min	10 seg	10 seg
Reference	no	explicit	hidden	no	explicit	explicit and hidden
High anchor	no	no	yes	no	no	hidden
Low anchor	no	no	yes	no	no	yes
2 simultaneous stimulus	no	no	no	no	yes	no
Presentation of sequences	once	once or twice in succession	twice in in succession	once	once	several concurrent (multi stimuli)
Possibility to change the vote before proceeding	no	no	no	no	no	yes

3.2 Objective Quality Assessment

Considering the bad properties of subjective testing (principally those associated with the fact that they rely on the work of panels of users), researchers and engineers had naturally looked for automatic procedures, called *objective methods*. *Objective* metrics are algorithms and formulas (generally signal processing algorithms) that measure, in a certain way, the quality of a stream. With a few exceptions, objective metrics propose different ways of comparing the received sample with the original one[3], typically by computing a sort of distance between both signals. So, it is not possible to use them in an real–time test environment, because the received and the original video are needed at the same time in the same place. But the most important problem with these quality metrics is that they often provide assessments that do not correlate well with human perception, and thus their use as a replacement of subjective tests is limited.

The most commonly used objective measures for video are:

- Peek Signal to Noise Ratio (PSNR),
- ITS' Video Quality Metric (VQM) [21, 338],
- EPFL's Moving Picture Quality Metric (MPQM),
- Color Moving Picture Quality Metric (CMPQM) [328, 329],
- Normalization Video Fidelity Metric (NVFM) [329].

They differ in complexity, from the simplest one (PSNR) to the most sophisticated one based on the Human Vision System (HVS) (CMPQM or NVFM). Based on previous work [218, 271], in what follows we present some of the most common objective tests.

[3]In audio, the E–model [172] standard specify an objective quality measurement that does not need the original signal; we doesn't know a equivalent procedure for video.

3.2.1 Peak Signal–to–Noise Ratio (PSNR)

The most common and simple objective video quality assessment is the Peak Signal–to–Noise Ratio (PSNR). We can define the Mean Square Error (MSE) between a original video sequence o and the distorted sequence d as:

$$\text{MSE} = \frac{1}{K.M.N} \sum_{k=1}^{K} \sum_{m=1}^{M} \sum_{n=1}^{N} [o_k(m,n) - d_k(m,n)]$$

where each video sequence has K frames of $M \times N$ pixels each, and $o_k(m,n)$ and $d_k(m,n)$ are the luminance pixels in position (m,n) in the k^{th} frame of each sequences.

The PSNR is the logarithmic ratio between the maximum value of a signal and the background noise (MSE). If the maximal luminance value in the frame is L (when the pixels are represented using 8 bits per sample, $L = 255$) then:

$$\text{PSNR} = 10.\log_{10} \frac{L^2}{\text{MSE}}$$

An extension to color video sequences has been proposed, by considering also the chrominance. The first advantage of PSNR is that it is easy to compute. However, it is not appropriate in our QoE context since it does not correlate well with perceptual quality measures [26].

3.2.2 ITS' Video Quality Metric (VQM)

ITS' Video Quality Metric (VQM) [21, 338] is developed by the "Institute for Telecommunication Sciences" (ITS). First it extracts some features from both the original and the distorted sequences. The features are an objective measure that characterizes perceptual changes of the sequence, analyzing spatial, temporal, and chrominance information. Then, a set of quality parameters are computed comparing the original and distorted features. Using these parameters, a classification assigns a global quality measure. The classification is a linear combination calculated using functions that model human visual masking. These impairments are then statistically pooled to obtain a single quality measure for the total sequence.

Summarizing, VQM makes a comparison between the original and distorted sequences based only on a set of features extracted independently from each video. It is useful when is not possible to have the original and received sequences at the same time, for instance in a network. But it still need some information about both sequences.

VQM is accepted as an objective video quality standard by ANSI, and some studies shows a good correlation with subjective tests for low bitrate encodings (while it not performs well for encodings with high bitrates [26]).

3.2.3 Moving Picture Quality Metric (MPQM) and Color Moving Picture Quality Metric (CMPQM)

Moving Picture Quality Metric (MPQM) [26, 329] and its color extension, Color Moving Picture Quality Metric (CMPQM) [328, 329], were developed by researchers working at the "École Polytechnique Fédérale de Lausanne" (EPFL). They are the most used objective metrics based on the Human Vision System (HVS) model. Stimuli of the same amplitude are perceived different when they are included in flat spatial areas or in areas including edges. In general, Stimuli with different spatial and temporal frequencies are not perceived in the same way by the human eye. The HVS models these and other aspects of human visual perception, and it is included on the MPQM metric (and on the CMPQM metric) to improve his (their) performance and robustness. In particular, MPQM incorporates the most important human perception phenomenon: contrast sensitivity and masking. These phenomenons accounts for the fact that a minimal threshold is needed to detect a signal change, and the threshold depends on the contrast of the foreground/background relation.

MPQM considers a set of distorted perceptual components obtained from the original sequence and their difference with the distorted one. Each perceptual component is computed using signal processing filters, and they measure in some way the perceptual differences between the original sequence and the distorted sequence. Each component has a sensitivity in the perceptual quality, considering its weight, a global distortion E is computed (as an important improvement of the MSE in the PSNR metric). Finally a Masked PSNR is defined:

$$\mathrm{MPSNR} = 10\log_{10}\frac{L^2}{E^2}$$

CMPQM is an extension to MPQM that also use the chrominance values. At first step it transforms the original and the distorted sequences to the linear opponent-color space (B/W, R/G, B/Y). The computation follows in a very similar way then for the original MPQM.

The authors of these proposals show that both metrics, MPQM and CMPQM, correlate well with subjective assessment in particular scenarios [329], specially for high bit rate codifications. However, in more general situation, the correlation is more variable.

3.2.4 Normalization Video Fidelity Metric (NVFM)

Normalization Video Fidelity Metric (NVFM) [329], also developed by EPFL, is based on a visibility prediction followed by a normalization stage. As MPQM, the prediction is made in the pixel domain, using space and time linear transformations, but is applied to the original sequence and in the distorted sequence (instead to its difference as in MPQM). The perceptual components obtained in the prediction stage are normalized based on the ratio between the excitatory and inhibitory of a inter–channel masking that consider the sensitivity weight. Finally, the measure metric is computed as the squared vector sum of the difference of the normalized responses.

3.2.5 Other Works in Objective Video Quality Measures and Comparison

Structural Similarity Index (SSIM) [342, 343] is a structural distortion based technique. All the previously described methods are error based. Instead, HVS is not oriented towards that extracting structural information from the viewing field. Therefore, a measurement of structural distortion should be a good approximation of perceived image distortion. Only studied by its authors, it is not clear yet the correlation with subjective tests of this approach.

Noise Quality Measure (NQM) [77, 341] models the error source with a linear frequency distortion and additive noise injection; the two sources are considered independent. A distortion measure (DM) is used for the effect of the frequency distortion, and a noise quality measure (NQM) is used for the effect of the additive noise. A global perceptual quality based on the two measures (NQM and DM) is not defined.

Table 3.2 summarizes the considered objective metrics. Observe that all objective video quality metrics use the original video sequence (and the distorted video sequence). Therefore, it is not possible to use them in an real–time test environment, because the received and the original video are needed at the same time in the same place. Also in some applications the complex computations involved are a limitation to their use. But the most important disadvantage of these metrics is that they often provide assessments that do not correlate well with human perception. As IRT [164] says: “despite efforts to develop objective measuring methods, the results repeatedly fail to reflect quality as perceived by the human eye, which is uninterested in purely logical approaches. Subjective tests are therefore recommended for checking the quality of digital videos”.

Table 3.2: Objective Metric Comparison.

Parameter	PSNR	VQM	MPQMS	CMPQM	NVFM	SSIM
Use of Original Sequence	yes	yes	yes	yes	yes	yes
Chrominance Consideration	no	yes	no	yes	yes	yes
Mathematical complexity	simple	very complex	complex	complex	complex	complex
Correlation with Subjective Methods	poor	good	varying	varying	varying	unknown

3.3 An Hybrid Quality Assessment approach: Pseudo-Subjective Quality Assessment

In next chapter we present a hybrid approach between subjective and objective evaluation called Pseudo Subjective Quality Assessment (PSQA) [221]. It is a technique allowing to approximate the value obtained from a subjective test in an automatically way. Therefore, it puts together the advantages of both approaches: it allows an automatic and simple measurement at a very low cost, and it does not need the original sequence. As a consequence, it is ideal for real–time applications (and for our project).

The idea is to have several distorted samples evaluated subjectively by a panel of human observers, and then to use the results of this evaluation to train a specific learning tool (in PSQA the best results come from the Random Neural Networks one [100]) in order to capture the relation between the parameters that cause the distortion and the perceived quality. It is necessary to understand that, in general, the distorted sequences used in the test phase are generated for a given application, and therefore a new PSQA module must be generated for every new application.

The PSQA assessment approach is proposed in [218]. The authors studied its applicability to different contexts, their work principally concentrates in speech quality assessment [219, 220], but also he studied the video quality [221, 222], specially for the standard video-conference codec H.263 [177]. In video, that is our focus, the methodology produces good evaluations for a wide range variation of all the quality affecting parameters; the published works present evaluations with correlation coefficients with values coming from human observers up to 0.97. To use the new technique, he designed a new rate control mechanism [220]. This mechanism combines an automated real–time multimedia quality assessment with a TCP–friendly rate controller, with the objective of deliver the best multimedia quality and save bandwidth consumption.

A previous inspiring work [237] used Artificial Neural Networks to estimate audio quality from signal factors (PSQA' authors work approach analyze application and network–level factors instead).

Following the first study, the PSQA's performance was studied in–depth in the Voice over IP context [271]. This work helped to validate the methodology itself and to compare it with other quality assessment techniques [67, 223, 224, 295]. The authors studied one–way and interactive VoIP applications, in wired and wireless networks. They also used PSQA to evaluate different redundant schemes, specially Forward Error Correction (FEC) [293]. With respect to the use of PSQA in control application, they presented a very simple priority scheme for packets in a wireless link [294, 296].

The PSQA methodology was also applied to control a DiffServ IP network in order to improve the performance of multimedia services [323]. This study worked at router's level in two main research areas: Active Queue Management (AQM) for Assured Forwarding and DiffServ-aware video streaming.

The main target of our work is a peer–to–peer prototype for live video streaming. In the next chapter we extend PSQA to use it in this context. We improve the methodology in different ways. First, we study the effects of several parameters on the perceived video quality, in particular the frame loss effect, instead of packet lost (studied in all previous works). Second, the influence of source' motion in the quality is considered. And third, we analyze the impact of server failures in perceived quality. We use the PSQA quality assessment: to optimize a multi-source streaming technique with respect to server failures behavior (Chapter 6), to implement a robust structured overlay peer-to-peer based on quality guarantees (Chapter 8), and to measure and to monitor the quality in a general Video Delivery Network (VDN) automatically (Chapter 9).

3.4 Summary

In this chapter we presented the state of the art in video quality assessment. Several subjective methods and objective metrics are explained and compared. It is clear from this summary that current quality assessment methods (subjective and objective) are not able to measure the quality in real–time (by definition, in the subjective case, and due to the need of the original video, in the objective one), disabling its use to monitoring and to controlling purposes.

Subjective tests are expensive and not suitable for real–time operation because they need a test campaign. Also, objective metrics are not suitable for real–time applications because they need the original an the distorted sequences at the same time to compute the metric. Moreover, they often provide assessments that do not correlate well with human perception.

The hybrid quality assessment approach, Pseudo-Subjective Quality Assessment (PSQA), adapts very well to our needs to measure in real–time the subjective video quality of a live video distribution system. The PSQA methodology will be explained in–depth in next chapter.

Chapter 4

PSQA – Pseudo–Subjective Quality Assessment

The main target of our work is a quality-centric design of a peer–to–peer prototype for live video streaming. To measure and to control the network we use the Pseudo–Subjective Quality Assessment (PSQA) methodology[1]. In this chapter we present the PSQA methodology (Sections 4.3, 4.4, and 4.5). In Chapter 5, we explore the sensivity of the perceived video quality with respect to several quality–affecting parameters and obtain some mapping function of quality that will be used in the rest of this work.

4.1 Overview of the PSQA Methodology

Let us briefly describe the way PSQA works. We start by choosing the parameters we think will have an impact on quality. This depends on the application considered, the type of network, etc. Then, we must build a testbed allowing us to send a video sequence while freely controlling simultaneously the whole set of chosen parameters. This can be a non-trivial task, especially if we use a fine representation of the loss process. We then choose some representative video sequences (again, depending on the type of network and application), and we send them using the testbed, by changing the values of the different selected parameters. We obtain many copies of each original sequence, each associated with a combination of values for the parameters, obviously with variable quality. The received sequences must be evaluated by a panel of human observers using some subjective testing procedure. Each human observer evaluates many sequences and each sequence is evaluated by all the observers (as specified by an appropriate test subjective norm). After this subjective evaluation, we perform a statistical filtering process to this evaluation data, to detect (and eliminate, if necessary) the bad observers in the panel (a bad observer is defined as being in strong disagreement with the majority). All these concepts have well defined statistical meanings. At that stage enters the training process, which allows learning the mapping $Q()$ from the values of the set of parameters into perceived

[1]Current video quality assessment methods are not able to measure the quality in real–time disabling its use to monitoring and to controlling purposes (see previous chapter for details).

quality. To train, we use the Random Neural Networks learning tool [100], that shows the best success results [292].

Figure 4.1 represents graphically the whole process.

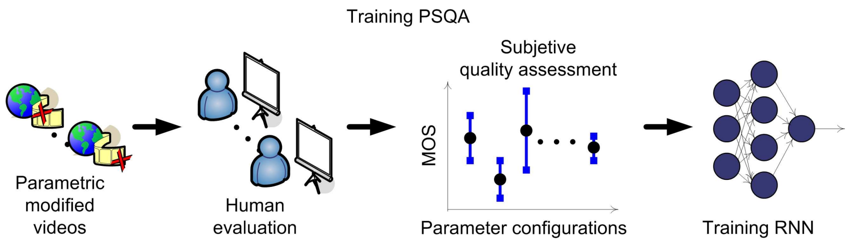


Figure 4.1: Training PSQA method.

After the training phase, PSQA is very easy to use: we need to evaluate the values of the chosen parameters at time t (that is, to measure them), and then to put them into the function $Q()$ to obtain the *instantaneous* perceived quality at t. See Figure 4.2 where we represent PSQA in operation.

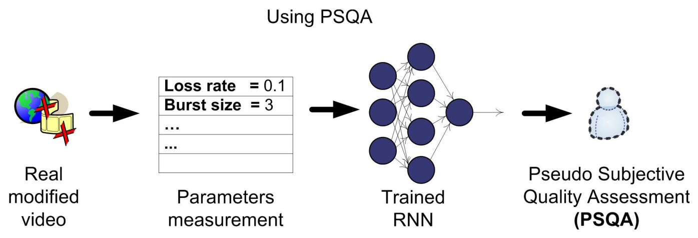


Figure 4.2: Using PSQA method.

Summarizing, we can divide the PSQA methodology in three main stages:

- (i) the election of the parameters that we think will have an impact on quality (Section 4.3);
- (ii) the evaluation of these parameters by means of subjective tests campaign (Section 4.4);
- (iii) and finally the training process, which allows to learn the mapping from the values of the set of parameters into perceived quality obtained in the tests campaign (Section 4.5).

Let us now introduce some notation before describing the methodology in–depth.

4.2 A Formal Notation

Following previous work, we use the following formal notation to reference the principal aspects of PSQA.

The set of *a priori* quality-affecting parameters are denoted by $\mathcal{P} = \{\pi_1, \cdots, \pi_P\}$. The parameter π_i can take a discrete set of possible values $\{p_{i1}, \cdots, p_{iH_i}\}$, with $p_{i1} < p_{i2} < \cdots < p_{iH_i}$. A configuration $\vec{v}$ vector is defined as a vector of possible values for each parameter: $\vec{v} = (v_1, \cdots, v_p)$, where $v_i \in \{p_{i1}, \cdots, p_{iH_i}\}$.

The total number of possible configurations is then $\prod_{i=1}^{P} H_i$. The possible configuration space is usually large, we select a subset of configurations for the testing phase, $\mathcal{S} = \{\vec{v_1}, \cdots, \vec{v_S}\}$, where $\vec{v_s} = (v_{s1}, \cdots, v_{sP})$ and v_{sp} being the value of parameter π_p in configuration $\vec{v_s}$.

In a testbed environment, we generate a distorted video sequence σ_s, applying the $\vec{v_s}$ parameters conditions to an original sequence σ. Therefore, we build a test set $\mathcal{T} = \{\sigma_1, \cdots, \sigma_S\}$ of distorted video sequences, that correspond to the result of sending S-times σ through the source-network system for the selected configurations $\mathcal{S}$.

The test set $\mathcal{T}$ is subjectively evaluated by a panel of human users following some subjective method (see section 3.1). The average perceptual quality score, Mean Opinion Score (MOS), corresponding to sequence σ_s is denoted by $\overline{q_s}$, where $\overline{q_s} \in \Psi$, and Ψ is the scale range of the test (one option of the list in Figure 3.1).

The final goal of the PSQA methodology is to obtain a perceptual quality function of the parameters $Q : \mathcal{P} \longmapsto \Psi$, such that

(i) for any sample $\sigma_s \in \mathcal{T}$, $Q(v_{s1}, \cdots, v_{sP}) \approx \overline{q_s}$,

(ii) for *any other* vector of parameter values $(v_1, \cdots, v_P)$, $Q(v_1, \cdots, v_P)$ is close to the MOS that would be associated in a subjective evaluation with a media sample for which the selected parameters had those specific values $v_1, \cdots, v_P$.

To approximate this function, we use an Random Neural Network (RNN) as will be described in 4.5.

4.3 Stage I: Quality–affecting factors and Distorted Video Database Generation

In the first stage, we select the quality–affecting factors that we think have an impact on quality, we also generate a distorted video database varying these parameters to be used on next stage. Figure 4.3 represents graphically the stage I.

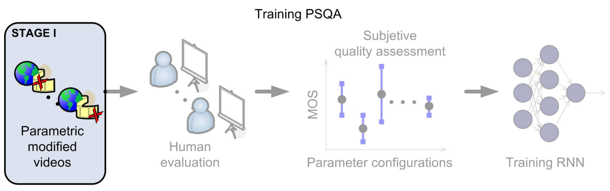


Figure 4.3: Stage I: PSQA method.

4.3.1 Quality–affecting Parameters Selection

There are lot of factors that influence the perceptual video quality. As observed before, they depend, among other elements, on the specific application, the networking technology, etc. The most important task in the methodology is to correctly identify the parameters that have most impact in the quality. It is thus recommended to do some preliminary in-house tests to verify the choice.

The global quality of the method is very influenced by the parameters chosen, because the rest of the methodology application has to be repeated if a missing factor is detected later (it includes repeating the time–consuming subjective test campaign). The accuracy and the robustness of the measure are diminished when unrelevant parameters are chosen, because the learning tool will not be able to mimic subjective perception since important information is missing.

In this situation, it is logical to think that is important to incorporate the larger number of possible parameters. But as we will see, this implies more value combinations to be watched and evaluated in the test campaign, and there are empirical restrictions in the number of sequences that can be used (because the excessive time and manpower needed in the test campaign)[2].

Following this reasoning, the number of possible values that a parameter can take has to be carefully selected: they must have the needed granularity, but keeping the number of configuration low enough, in order to have manageable size tests.

It is possible to classify the quality–affecting factors into four categories, depending on the factor source:

- **Environment parameters.** Environment parameters are for example: the ambient noise level, the lighting of the room, the fidelity of the display (monitor or TV) and the speakers used, the computation power capabilities of the media player (the computer), etc. The environment parameters are usually uncontrollable, and difficult to measure in a testing session. Therefore, we will usually not consider these factors in the developed methodology.

- **Source parameters.** The original source video signal has an obvious strong impact on the global perceived quality. For example the sound level, or the luminance level, and the average motion, have an impact in the quality, especially in conjuntion with other factors, like very low bit rate encoding and/or packet losses in the network.

[2]In all our studies (and previous work) we consider less than ten parameters.

The video source parameters like the nature of the scene (e.g. amount of motion, color, contrast, image size, etc.) that depends on the characteristics of the particular sequence that is being transmitted can also have an impact on the human perception of the video quality.

The encoding or compression parameters are the most important source factors. Some of these parameters are: the type of the codec used (MPEG-2, MPEG-4 Part 2 or Part 10, etc.), the sampling rate, the number of bits per sample, the bit rate, the frame rate, the encoded frame types, the number of layers in the case of layered codecs, etc.

The sender can implement some quality–improving techniques (the receiver also has to interpret the added or modified data). The most common one are: Forward Error Correction (FEC), interleaving, layered coding, and, for us, our multi-source streaming proposal. Basically, these improving techniques are designed to mitigate the effect on the quality of the packet losses in the network, and the server failures in the multi-source case. Therefore if some parameter of these improving techniques is used, it is also necessary to include another parameter measuring the failure factor (i.e. the losses).

- **Distribution (or network) parameters.** The quality–of–service (QoS) measures at the network are main components in network design and management, in general. Typically, they include packet loss, delay, jitter, and bandwidth factors, but it is not clear how much these factors affect the global perceived quality, and therefore which one and how they have to be parameterized. Even more, there are different sensitivities of the quality with respect to the specific multimedia application. For instance, if real-time distribution is needed, the packet loss rate is the most important network parameter, and the retransmission and buffering receiver factors take an important role. If there is interactivity (for example in a video call), delay and jitter have an important role also, adding echo, crosstalk and lost of audio/video synchronization that can be mitigated with receiver techniques too.

  Internet is a best–effort network, meaning that there is no QoS guarantees to the users. Nevertheless, some techniques are applied to mitigate the network distribution effect on quality. For instance, Random Early Detection (RED) is an active queue management algorithm, applied into the routers of the network, that partially avoids congestion. In dedicated IP networks, a set of techniques are often developed to allow traffic engineering, like Differentiated Services (DiffServ) and Multi Protocol Label Switching (MPLS). All this network improving quality techniques are parameterized and can be taken into account.

  The transport–layer protocols (i.e. Real–time Transport Protocol (RTP)) and application–layer signaling protocols (i.e. RTCP, H.323 and SIP) are important for quality also, because they can provide, in different ways, retransmission, extra synchronization information, congestion control algorithms, etc.

  Overlay networking is becoming popular these days. These are virtual networks developed at the application level over the Internet infrastructure. In particular, Peer–to–Peer networks are becoming more and more popular today (they already generate most of the traffic in the Internet). For instance, P2P systems are very used for file sharing and distribution; some known examples are Bittorrent, KaZaA, eMule, etc. They use the often idle capacity of the clients to share the distribution load. As a consequence, as the number of customers increases, the same happens with the global resources of the network. Their main technical problem is that peers connect and disconnect with high frequencies, in an autonomous and completely asynchronous way. This means that the resources of the network as a whole are also highly variable, and thus, that the network must be robust face to these fluctuations. The main idea that has been considered to deal with these problems is to build the systems using some redundancy in the signals: it is possible to increase the number of sources and/or the amount of redundant information sent through the network. This can be used as a tool to deal with the problem of nodes leaving the network (we will refer to this situation as a node failure), causing partial signal losses to some clients, and therefore a degradation of quality. In section 2.2 we study the failure factors in the distribution in a Video Delivery Network (VDN), and in 2.3 some particularities of Peer–to–Peer network for video delivery.

- **Receiver parameters.** Besides the quality–improving techniques implemented by the sender (with the understanding of the receiver), there are a set of quality–improving procedures which can be implemented at the receiver alone.

  Some examples are: buffering, loss concealment (insertion, interpolation and regeneration of lost data), and congestion control improvements in UDP streams.

Another relevant classification for the quality–affecting factors is with respect to the abstract layer where the measurement is made:

- **Network level parameters.** In IP networks, network level parameters are the typical quality–of–service (QoS) measures: packet loss, delay, jitter, bandwidth, etc.

- **Application level parameters.** Some factors of the application influence the quality. The encoding used is a main one; other examples are the buffering technique, the streaming protocol (that uses mechanisms to mitigate the packet losses and the network congestion, etc), etc. A common stack protocol of video streaming is: RTP/UDP/IP. At the encoding output, the video is an Elementary Stream; after that it is divided into packets (Packetized Elementary stream); then, it is multiplexed with an audio stream (Transport Stream). Finally, this Transport Stream is streaming over RTP. Therefore, also at the application level, there are different levels where it is possible to find relevant quality–affecting parameters.

In proprietary IP network deployments, to measure network level parameters is an easy task for operators and managers, and to measure application level ones implies an interference in the client service. On the other hand, the network level parameters are less correlated with the quality than the application level ones. But, less correlation does not mean less precision in the quality assessment, it usually means less robustness with respect to some change in the environment factors. For example, we know that losses degradate quickly the quality, but it is not the same to measure packet losses than frame losses. For example, at the application level, it is possible to count the late frames and to discriminate the loss according to the frame type. Depending on the streaming stack protocol, the packet losses generate different frame losses (see section 5.1), and therefore different perceived quality impact. If we want to measure the losses at the network level, a streaming stack protocol has to be fixed before applying the PSQA methodology, losing generality. In a Peer–to–Peer system the designer can easily implement measures at the application level, because he develops the P2P application. Therefore, in our context, we can apply the PSQA methodology without the details of the streaming technique, measuring the losses at the application level.

4.3.2 Distorted Video Database Generation and Testbed Development

After selected the quality–affecting parameters to be taken into account, (i) we need to validate this selection, (ii) then, we must generate a video database of distorted video sequences to be used in the next stage (the subjective test).

There are two ways of reaching these aims: experimentation and emulation. The experimental approach needs some manipulation of the video delivery service, and of course the capability of measuring the parameters. The emulation approach needs a realistic model, assuming some simplifications (i.e. introducing some measure errors). Considering that the experimentation is time expensive and that we are in a validation stage, we recommend to develop a testbed to emulate the parameter configurations, capable of allowing us to send a video sequence while freely controlling simultaneously the whole set of chosen parameters.

To construct the database, we need to choose some representative parameters configuration of the whole set of possible ones. Then, we send the selected sequences using the testbed. We obtain a set of distorted videos, each associated with a different combination of values for the parameters. After assigning a quality value to these samples (at the subjective test stage), we will use them to train the learning tool. The distorted video database can not be very large because it has to be evaluated in a subjective test campaign, but on the other hand, it has to be enough to train and validate the learning tool.

4.4 Stage II: Subjective Quality Assessment

In the second stage of the PSQA methodology, we evaluate the parameters chosen in previous stage by means of a subjective test campaign. Figure 4.4 represents graphically the stage II.

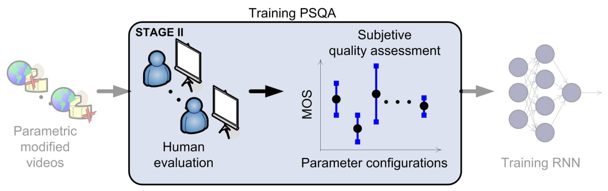


Figure 4.4: Stage II: PSQA method.

4.4.1 A Subjective Test Campaign

For the subjective test campaign we can use some of the methods explained in Section3.1. In particular, we follow the Single Stimulus (SS) method of ITU-R BT.500-11 [171]. Briefly, some important considerations are:

- **With respect to the video database and its presentation.** The ITU recommends that the duration of each video sequence should be less than 10 seconds, without abrupt termination or incomplete meaning. Between sequences, a gray scene (not longer than 500ms) can be used for a more natural presentation.

  The sequence test evaluation should be divided into multiple sessions, with each session not longer than 30 minutes. Each session should start with some *warm–up* sequences (about four or five), which should not be taken into account in the results

  Some sequences with perfect quality have to be used in each session as references. The reliability of subjects should be qualitatively evaluated by checking their votes for the references.

- **With respect to the subjects.** For statistical reasons, the number of subjects required to carry out the test can vary from 4 to 40, although most tests require from 10 to 20 subjects. Subjects should not be experts in video quality evaluation or its theory, and they must have normal visual acuity.

  Some written instructions should be given to the subjects before carrying out the test. Also it should be necessary to show preliminary trials, to help in the explanation and to familiarize them with the voting scale and the task they will perform. They should not receive any information on the quality–affecting factors that are being considered in the database.

4.4.2 Mean Opinion Score (MOS) Calculation and Related Statistical Analysis

Following our formal notation (Section 4.2), the test set $\mathcal{T} = \{\sigma_1, \cdots, \sigma_S\}$ of distorted video sequences is subjectively evaluated by a human panel. Let us denote by N the number of subjects in the panel, and by q_{is} the vote of sequence σ_s made by subject i. The average perceptual quality score or Mean Opinion Score (MOS), corresponding to sequence σ_s is denoted by $\overline{q_s}$, that is

$$\overline{q_s} = \frac{1}{N} \sum_{i=1}^{N} q_{is} \tag{4.1}$$

The standard deviation of this sequence σ_s is denoted by δ_s, using

$$\delta_s = \sqrt{\sum_{i=1}^{N} \frac{(q_{is} - \overline{q_s})^2}{N-1}}$$

It is possible that some subjects do not follow the voting instructions, or do not pay enough attention during the test; these subjects degradate the global precision of the PSQA method and usually they are statistical filtered in this stage.

We follow the ITU-R BT.500-11 recommendation [171] for our statistical filtering. First, we use the β_2 test to determine whether the votes have a normal distribution or not. Let

$$\beta_{2s} = \frac{m_{4s}}{m_{2s}^2} \text{ where } m_{xs} = \frac{1}{N}\sum_{i=1}^{N}(q_{is} - \overline{q_s})^x.$$

If $2 \leq \beta_{2s} \leq 4$ then the votes distribution $(q_{is})_{i=1,\cdots,N}$ may be assumed to be normal.

Knowing that the votes have a normal distribution, it is possible to reject bad observers, defining a bad observer as a subject such that its votes are systematicly far from the mean votes. for each subject i we define two counters P_i and Q_i following the Algorithm 1.

Algorithm 1 Statistical Filtering of β_2 Test. Update rules for P and Q.

```
$P_i \leftarrow 0$
$Q_i \leftarrow 0$
for $s = 1$ to $S$ do
  if $2 \leq \beta_{2s} \leq 4$ then
    if $q_{is} \geq \overline{q_s} + 2 \times \delta_s$ then
      $P_i \leftarrow P_i + 1$
    end if
    if $q_{is} \leq \overline{q_s} - 2 \times \delta_s$ then
      $Q_i \leftarrow Q_i + 1$
    end if
  else
    if $q_{is} \geq \overline{q_s} + \sqrt{20} \times \delta_s$ then
      $P_i \leftarrow P_i + 1$
    end if
    if $q_{is} \leq \overline{q_s} - \sqrt{20} \times \delta_s$ then
      $Q_i \leftarrow Q_i + 1$
    end if
  end if
end for
```

We can then eliminate any subject i such that

$$\frac{P_i + Q_i}{S} > 0.05 \text{ and } \left|\frac{P_i - Q_i}{P_i + Q_i}\right| < 0.3$$

The β_2 test should only be run once. After this filtering, a final Mean Opinion Score (MOS) should be recomputed on the resulting subject set, using Equation 4.1. Also an average root mean square error $\overline{\delta}$ of a subject of the panel can be computed to know the variability of the panel with respect to their quality assessment:

$$\overline{\delta} = \sum_{i=1}^{N} \frac{\sqrt{\sum_{s=1}^{S} \frac{(q_{is} - \overline{q_s})^2}{S}}}{N} \tag{4.2}$$

4.5 Stage III: Learning of the quality behavior with a Random Neural Networks(RNN)

The result of stage II is a quality evaluation for each video sequence in the test set $\mathcal{T}$. Each distorted video sequence is based on some parameters conditions. Therefore, the final goal of the PSQA methodology is to obtain a perceptual quality function of the parameters such that they smoothly approximate the empirical test values. To build this function, we use a Random Neural Network (RNN). In what follows we present this learning tool. Figure 4.5 represents graphically the stage III.

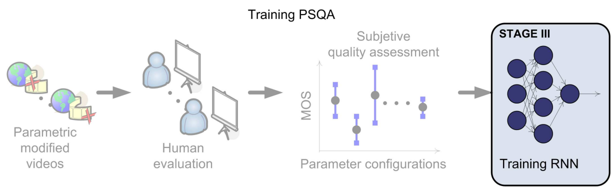


Figure 4.5: Stage III: PSQA method.

4.5.1 Random Neural Networks (RNN) overview

Random Neural Networks (RNN), also called G-networks, were proposed in [100, 101, 103] as an new type of Neural networks. They are also open Markovian queuing network with "positive" and "negative" customers (see below). Since the proposal, RNNs have been successfully applied in many areas. Its author used it in networking applications several times [70, 71, 104–106, 108, 111]. For a pretty complete survey of its applicability see [24].

The random neural network (RNN) is a simplified model of a biological nervous system. The RNN is formed by a set of neurons which exchange signals in the form of spikes, like the natural pulsed behavior. Each neuron's state is a non-negative integer called its potential, which increases by 1 when a positive signal (an excitation) arrives to it, and decreases by 1 when a negative signal (an inhibition) arrives. The signals can originate outside the network, or they can come from other neurons. Usually, the signals can not flow in an arbitrary way: a topology is designed that specify which neurons receive signals from which ones, and which neurons receive signals from the environment.

In our PSQA methodology, we use a particular RNN architecture, where each neuron behaves as a $./M/1$ queue with respect to positive signals. This means that positive signals are interpreted as customers, these customers arrive to the neurons, and are served in a FIFO order. The service rate at neuron i is denoted by μ_i. A neuron i receives positive customers from the environment according to a Poisson process with rate λ_i^+ (no negative customers arrive from the environment).

The potential of a neuron is the number of positive customers in its waiting queue. When a neuron receives a positive customer, either from another neuron or from the environment, its potential is increased by 1 (i.e. increase the waiting queue). If a neuron with a strictly positive potential receives a negative customer its potential decreases by 1 (i.e. a negative customer arrive kills the last customer at the queue (if any), and kills itself). After leaving neuron (queue) i, a customer either leaves the network with probability d_i, goes to queue j as a positive customer with probability p^+_{ij} or as a negative customer with probability p^-_{ij}. So, if there are M neurons in the network, for all $i = 1, \cdots, M$:

$$d_i + \sum_{j=1}^{M}(p^+_{ij} + p^-_{ij}) = 1 \tag{4.3}$$

At any time, the network state is specified by the potential (positive customers waiting queue) of its neurons: $\vec{N}_t = (N^1_t, \cdots, N^M_t)$, where N^i_t is the potential of neuron i at time t (see Figure 4.6 for a graphical representation of a neuron in a RNN topology).

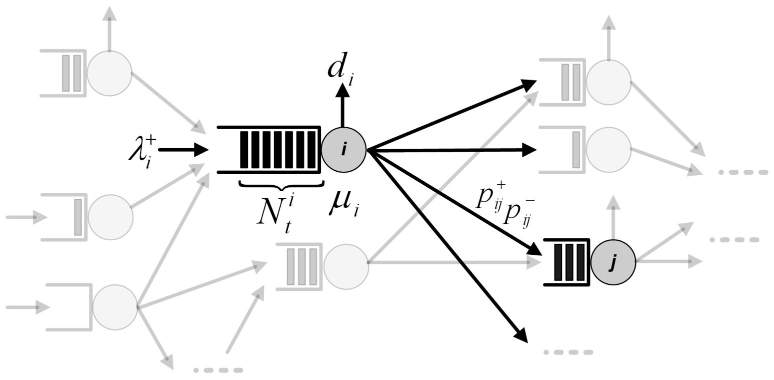


Figure 4.6: Neuron detail in a RNN.

As shown in [101, 103], if this Markov model is stable, we have, in steady-state, the following joint probability distribution:

$$\Pr(\vec{N}_t = (k_1, \cdots, k_M)) = \prod_{i=1}^{M}(1 - \varrho_i)\varrho_i^{k_i}.$$

Equilibrium factors ϱ's are the loads in the network (i.e. ϱ_i is the asymptotic probability that queue i is not empty).

The loads are obtained by solving the following non-linear system of equations, built in terms of the individual service rates, environment arrivals and routing probabilities:

$$\varrho_i = \frac{\lambda^+_i + \sum_{j=1}^{M} \varrho_j w^+_{ji}}{\mu_i + \sum_{j=1}^{M} \varrho_i w^-_{ij}}, \; i = 1, \cdots, M \tag{4.4}$$

The numbers $w^+_{ij} = \mu_i p^+_{ij}$ and $w^-_{ij} = \mu_i p^-_{ij}$ are called *weights* (as in the standard neural network terminology).

It can be proved that th network is stable if and only if this non-linear system (Equation 4.4) has a solution where for all $i \in \{1, \cdots, M\}$, $\varrho_i < 1$.

4.5.2 Using RNN as a function approximator: a learning tool

A common use of the RNN is as an approximator for bounded and continuous functions, in our context, a learning tool. Knowing the values of a set of input (i.e. the mapping between inputs and outputs), the RNN *learns* how to evaluate the function for any input. The function input is the vector $\vec{\lambda} = (\lambda_1^+, \cdots, \lambda_M^+)$, i.e. the environment customers arrival rate. And the function output is the vector $\vec{\varrho} = (\varrho_1, \cdots, \varrho_M)$, i.e. the stationary loads of the RNN[3]. As we will see next, we usually consider that only some neurons have environment customers arrivals, and see the output load only in some neurons (not in the whole vector $\vec{\varrho}$).

The result of stage II is a quality evaluation (MOS), $\overline{q_s}$, for each video sequence σ_s of the test set $\mathcal{T} = \{\sigma_1, \cdots, \sigma_S\}$. The distorted video sequence σ_s is generated in the testbed, with the parameters conditions $\vec{v_s} = (v_{s1}, \cdots, v_{sP})$. At stage III, a perceptual quality function is approximated using a Random Neural Network (RNN). This function, $Q : \mathcal{P} \longmapsto \Psi$, from the parameters to the perceived quality, approximates smoothly the values: $(\vec{v_s}, \overline{q_s})_{s=1,\cdots,S}$. Therefore, our RNN has:

- A single scalar output value, which corresponds to the perceived quality for any given parameter configuration. It means that we will only consider one output neuron, o, and so the perceived quality will be given by its load ϱ_o.
- It must have P input neurons, one for each quality–affecting parameter, and the rate of the arrival flow (of positive customers) at neuron p is $\lambda_i^+ = v_p$.

We need that for any sample $\sigma_s \in \mathcal{T}$, the function approximate the empirical quality, $Q(v_{s1}, \cdots, v_{sP}) \approx \overline{q_s}$. It means that if the arrival rate of each p neuron (of the P input neurons) is $\lambda_i^+ = v_{sp}$, then the occupation rate of output neuron o is $\varrho_o \approx \overline{q_s}$. The process that allows this adjusting is known as *training*, and it implies the calibration of the all weights of the neurons.

It is possible to consider the output ϱ_o as a function of the set of weights (denoted here by $\vec{w}$) and of the arrival rates ($\vec{\lambda^+}$): $\varrho_o(\vec{w}, \vec{\lambda^+})$, and then minimize the mean square error (MSE) for the know mapping values, defined by:

$$\text{MSE} = \frac{1}{S}\sum_{s=1}^{S}(\varrho_o(\vec{w}, \vec{v_s}) - \overline{q_s})^2. \tag{4.5}$$

The optimal weights values $\vec{w}_0$:

$$\vec{w}_0 = \text{argmin}_{\vec{w} \geq \vec{0}} \ \frac{1}{2}\sum_{s=1}^{S}(\varrho_o(\vec{w}, \vec{v_s}) - \overline{q_s})^2$$

[3]Do not confuse the output of the RNN as a learning tool (the vector $\vec{\varrho}$), with the output flow of customers of the network. In open queuing network models, usually the output process is the flows of customers going out of the system (the mean rate of customers leaving the network from neuron i is $\varrho_i \mu_i d_i$).

can be found using a gradient–descent algorithm (as proposed by Gelenbe [102]), or with more sophisticated techniques, as the Levenberg–Marquardt method or the *quasi–Newton* optimization proposed by [207]. In this work we use the simple gradient descent algorithm to train the RNN.

4.5.2.1 Training and Validating

There are three major learning algorithm classes: supervised learning, unsupervised learning and reinforcement learning. Each class correspond to a particular kind of problem, the *supervised learning* is used in function approximation problems[4].

Usually, when working with supervised learning algorithms, the empirical database is used to train the learning tool and also to test its performance. This means that we divide the mapping $(\vec{v_s}, \overline{q_s})_{s=1,\cdots,S}$ in two parts: one is used to train the tool, and the other one is used to validate it.

If the training set is too large, it is possible that the RNN will be overtrained, i.e. that the performance of RNN will be very good at the training phase while it will give very poor performance for the validating set. The performance is measured as the Mean Square Error (MSE) of the validating set. On the other hand, it is possible that the RNN will be undertrained, that is, that the training set is not large enough to extrapolate well the behavior with respect to the values in the validating set).

Depending on the number of sequences, the number of parameters, and the values that the parameters can take, the percentage sizes of the training and validating sets are chosen. A good start is 80% for the training set and 20% for the validating set.

The size of training and validating sets are one of the decisions to take when a RNN is in use. Another important decision is the RNN topology used (see below).

4.5.3 The RNN topology design

As said before, the customers can not flow in an arbitrary way, a topology is designed that specify which neurons receive customers from each other, and which neurons receive customers from the environment. The topology has an impact in the performance of the RNN approximation, and has to be chosen with some care.

For the RNN there are many possible topologies: feed–forward, recurrent, etc. In our experiments, we use the most common one, the *feed–forward*, where there are no cycles or loops in the network (i.e. a customer cannot visit more than once any given queue). The neurons, in a feed-forward network, are typically grouped in layers; where there is no edge between neurons at the same layer. See Figure 4.7 for a graphical representation of a typical feed-forward network.

[4]Unsupervised learning is a class of learning algorithm that not use human prepared examples to learn [131], it is used mainly in clustering and pattern recognition. Reinforcement learning is a class of learning algorithm that attempt to find optimal actions of an agent in a particular environment. It differs from the supervised learning in that correct input/output pairs are not explicitly presented [318].

Some good properties of the feed–forward topology are: simple formulas and simple training algorithms (see [100–103, 109] for details). In particular, the trained RNN corresponds to a rational function of its input variables; the degrees of the numerator and the denominator depend on the number of hidden layers and the number of neurons in each.

Previous experiments [271] analyze the performance of several neural network topologies, ranged from the extremely simple two–layer design to more complex ones with several hidden layers and feedback between neurons. They didn't show significant variations in performance between topologies, just a faster convergence of the complex ones during the training process. But the training is an off–line process that executes only once; therefore, in our experiments we only used the *two–layer* and *three–layer* designs.

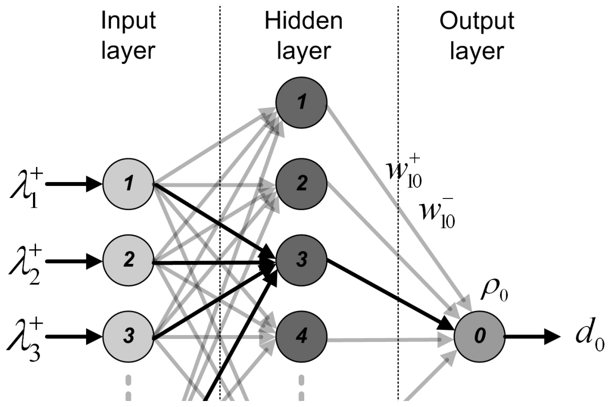


Figure 4.7: Components of a RNN. The schematics show a portion of a three–layer feed–forward network.

4.5.3.1 The simplest design: a Two–Layer RNN Topology

The *two–layer topology* is a feed-forward RNN, where there are P neurons receiving positive customers from the environment. This set of neurons is $\mathcal{P}$, and is called the *input layer*. Each external flow corresponds to one of the quality–affecting parameters, and λ_i^+ is the i^{th} parameter's value. Typically, normalized parameters are used, i.e. $\lambda_i \in [0,1]$. The *output layer* sends customers out of the network, it has only one neuron, denoted here by o, where we approximate the perceptual quality (with its load ϱ_o). See Figure 4.8 for a graphical representation of the two–layer design.

The neuron balance equation (Equation 4.3) can be expressed in function of weights ($w_{ij}^+ = \mu_i p_{ij}^+$ and $w_{ij}^- = \mu_i p_{ij}^-$ a) as:

$$d_i + \frac{1}{\mu_i}\sum_{j=1}^{M}(w_{ij}^+ + w_{ij}^-) = 1 \tag{4.6}$$

For each input neuron $i \in \mathcal{P}$ there are not outputs to the environment ($d_i = 0$) and there are customers flows only to the output neuron o ($w_{ij}^+ = w_{ij}^- = 0$ if $j \neq o$). Using the general

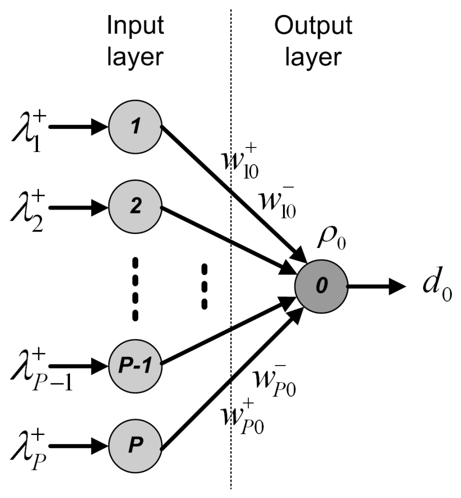


Figure 4.8: Two–layer RNN topology. The simplest RNN topology for PSQA: P input neurons (one for each paramater) and one output neuron o.

neuron load equation (Equation 4.4) we have that:

$$\varrho_i = \frac{\lambda_i^+}{\mu_i} = \frac{\lambda_i^+}{w_{io}^+ + w_{io}^-},\ i = 1, \cdots, P.$$

The output neuron o receives customers (only) from the input layer ($\lambda_o^+ = 0$), and emits all of its customers to the environment ($d_o = 1$). Substituting again the balance equation (Equation 4.6) in the neuron load equation (Equation 4.4), we have:

$$\varrho_o = \frac{\sum_{j=1}^{P} \rho_j w_{jo}^+}{\mu_o + \sum_{j=1}^{P} \rho_j w_{jo}^-}.$$

Diminishing the quantity of free variables, we keep constant the service rate of neuron o to the value $\mu_o = 0.01$.

This allows us to provide a simple closed form expression for quality as a function of quality–affecting parameters ($\vec{\lambda}$):

$$Q(\lambda_1^+, \cdots, \lambda_P^+) = \varrho_o = \frac{\sum_{i=1}^{P} a_i \lambda_i^+}{0.01 + \sum_{i=1}^{P} b_i \lambda_i^+} \tag{4.7}$$

where

$$a_i = \frac{w^+_{io}}{w^+_{io} + w^-_{io}} \quad \text{and} \quad b_i = \frac{w^-_{io}}{w^+_{io} + w^-_{io}}.$$

This also allows to easily analyze the sensitivity of the quality with respect to a specific parameter p. Formally, setting $a_0 = 0$ and $b_0 = 0.01$,

$$\frac{\partial Q}{\partial \lambda^+_p} = \frac{c_0 + \sum_{i=1}^{P} c_i \lambda^+_i}{(b_0 + \sum_{i=1}^{P} b_i \lambda^+_i)^2}$$

where

$$c_i = \begin{vmatrix} a_p & a_i \\ b_p & b_i \end{vmatrix} = a_p b_i - b_p a_i \ \text{(in particular, } c_p = 0\text{)}.$$

See Section 5.2.4 for our application of the two–layer topology.

4.5.3.2 The Three–Layer RNN Topology

Extending the two–layer design, the *three–layer topology* adds a *hidden layer* between the *input layer* and the *output layer*. As in the two–layer topology, the *input layer*, $\mathcal{P}$, has P neurons (one for each quality–affecting parameter) and the *output layer* has one neuron o (where we approximate the perceptual quality by the output neuron load ϱ_o). Between this output neuron and the set of input neurons $\mathcal{P}$, there is a set of H neurons, $\mathcal{H}$, called the *hidden layer*, receiving flows from the set $\mathcal{P}$ and sending customers to neuron o (there is no connection between $\mathcal{P}$ and o). See the graphical representation in Figure 4.9.

Like in the two–layer topology, the load of a neuron $i \in \mathcal{P}$ at the input layer is:

$$\varrho_i = \frac{\lambda^+_i}{\mu_i}, \ i = 1, \cdots, P.$$

The hidden layer is not directly connected to the environment, i.e. that for each neuron $h \in \mathcal{H}$, $\lambda^+_h = \lambda^-_h = d_h = 0$. Therefore, the load at a neuron $h \in \mathcal{H}$ is:

$$\varrho_h = \frac{\sum_{i \in \mathcal{P}} \frac{\lambda^+_i}{\mu_i} w^+_{ih}}{\mu_h + \sum_{i \in \mathcal{P}} \frac{\lambda^+_i}{\mu_i} w^-_{ih}},$$

and finally, the load at the output neuron o is:

$$\varrho_o = \frac{\sum_{h \in \mathcal{H}} \varrho_h w^+_{ho}}{\mu_o + \sum_{h \in \mathcal{H}} \varrho_h w^-_{ho}}.$$

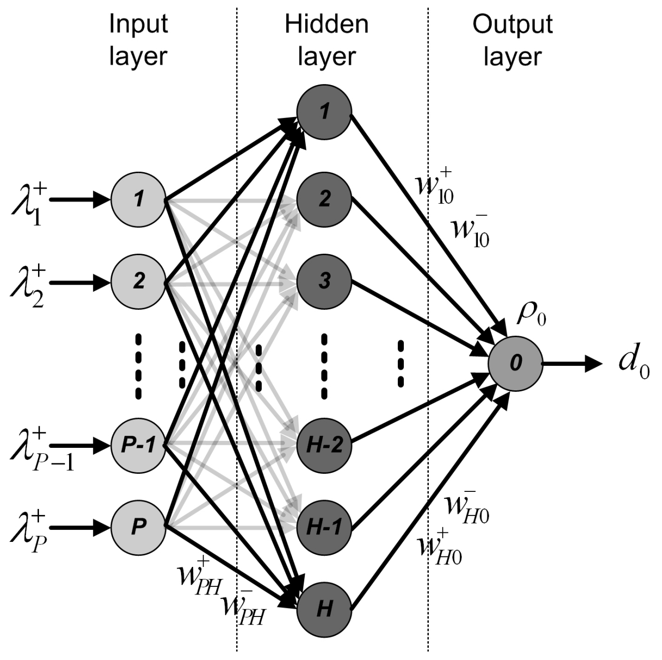


Figure 4.9: Three–layer RNN topology: P input neurons (one for each paramater), H hidden neurons, and one output neuron o.

In the training algorithm there are some restrictions over the weights to mantain the *stability* of the queueing network. To keep the network stable ($\forall i$ we need $\varrho_i < 1$), a sufficient condition is that: for any input neuron $i \in \mathcal{P}$, $\lambda_i^+ < \mu_i$; for any hidden neuron $h \in \mathcal{H}$, $\mu_h \geq P$; and finally $\mu_o \geq P$ in the output neuron[5]. Generally, these stability conditions are considered in the training algorithm of a feed–forward network.

In the three-layer topology is possible to arbitrary adjust the number of neurons in the hidden layer, H. With a small value of H, the RNN will not learn correctly the problem. On the other hand, with a large value of H, the topology will be more complex and there is a risk of overtraining. There are some heuristic methods to find the optimal number of hidden neurons. However, they do not perform well in the general case. Instead, in our experiments, we do the following:

1. Start by one hidden neuron.

2. Train this RNN topology until a minimum error goal is achieved, repeat the training process a few times and take the best one.

3. Until a maximum value is reached, increment the number of hidden neurons by one and go to step 2.

[5]The same stability argument can be applied to the two–layer topology.

4. Select the topology that gives the best performance for both the training and the testing steps.

Usually, the trainig algorithm initializes randomly the weights, so, different executions can achieve different performance. For robustness, it is necessary to train the RNN a few times (and take the best one) for each topology (i.e. each number of hidden neurons). Also, to get the best performance is possible to change the stopping criterion of the training process, for example to diminish the minimum error goal or to increase the amount of iterations.

4.6 Operation Mode

After training the PSQA method (stages I, II and III), the PSQA is very easy to use. It is necessary to measure the quality–affecting parameters at time t, and to evaluate these values with the $Q()$ function (actually the RNN) to obtain the instantaneous perceived quality at t. We represent PSQA in operation in Figure 4.10

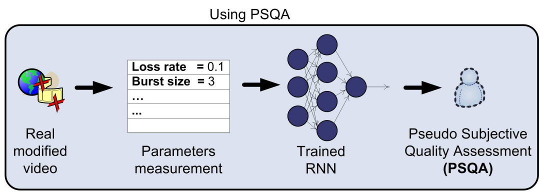


Figure 4.10: Using PSQA method.

4.7 Summary

In this chapter we presented the Pseudo–Subjective Quality Assessment (PSQA) methodology. The configuration of the PSQA method has three stages:

- **Stage I.** The election of the quality–affecting factors that we think will have an impact on quality. The generation of a distorted video database varying these parameters.
- **Stage II.** The evaluation of these parameters by means of a subjective test campaign.
- **Stage III.** The Random Neural Networks(RNN) training process, which allows to learn the mapping from the values of the set of parameters into perceived quality obtained in the test campaign.

After configuring, the PSQA is very easy to use, including to measure and to control video delivery networks in real–time. In next chapter, we explore the sensivity of the perceived video quality with respect to several quality–affecting parameters and obtain some mapping function of quality that will be used in the rest of this work.

Chapter 5

Studying the Effects of Several Parameters on the Perceived Video Quality

In this chapter we apply the PSQA methodology (explained in Chapter 4) to use it in video delivery networks design. In order to increase the generality of the assessment to different contexts, we improve the PSQA methodology in three ways. First, we study the effects of several parameters on the perceived video quality, in particular the frame loss effect, instead of packet loss (studied in all previous works). Second, we look at the influence of source motion in the quality. And third, we analyze the impact on quality of server failures.

We apply the PSQA procedure in two different scenarios: a simple one for MPEG-2 video encoding, and a sophisticated one, that considers video source properties for MPEG-4 video encoding. The results of these tests have been used in the core of many publications and they will help us in the rest of our work; in particular, we use the PSQA quality assessment: to optimize a multi-source streaming technique with respect to server failures behavior (Chapter 6), to implement a robust structured overlay peer-to-peer based on quality guarantees (Chapter 8), and to measure and to monitor the quality in a general Video Delivery Network (VDN) automatically (Chapter 9).

5.1 Parameters Selection: Network level or Application level?

The global quality of the PSQA methodology is very influenced by the quality–affecting factors chosen. There are a lot of factors that influence in the perceptual video quality, even if we consider a specific application (likes our live video delivery over P2P).

The parameters have to be carefully selected: we must include the most important affecting factors, while keeping their number reasonable (an excessive number leads more subjective tests and complex RNN's topologies). With our objectives, they have to be easy to measure in a real–time P2P application. For a general discussion about paramters selection see Section 4.3.

We know from previous work on PSQA that the loss process is the most important global factor for quality [221]. But we can measure the losses at different abstract layers, basically at the network level and at the application level. It is important to observe that in previous work using the PSQA technology, the analysis was done at the packet level. Here, we are looking at a finer scale, the frame one, because, as we will see, quality is more directly influenced by lost frames than by lost packets.

Network-level losses are easier to handle in the testbed and are easy to measure by network managers. But frame-level ones provide a more clear view of the loss process, and in our context they are also easy to measure because we have control over the P2P software application.

Measuring frame losses instead of packet losses covers many application–factors that influence the quality.

- **The buffering technique**. A frame can be delayed in the network or by the software player. In this case, it is possible that the frame will be rejected by the decoder. From the network point of view there are no losses, but we consider the late frame as frame losses.

- **The streaming protocol.** A common stack protocol for video streaming is: RTP/UDP/IP. But today the streaming over HTTP or TCP propietary protocols is more frequent (because of firewall client restrictions) [79]. Due to the retransmissions, losses of UDP and TCP packets have different impact on frame losses. Moreover, the streaming protocols use usually specific mechanisms to mitigate packet losses and network congestion, implying an intricate relationship between packet losses and frame losses.

5.1.1 Packet and Frame Losses Correlation

We measure the empirical relationship between packet and frame losses in the testbed. In our testbed we use a VLC streaming server, a router (which emulates the network, and generates the packet losses), and a VLC client (who receives the stream and measure the frame losses). See the testbed scheme in the Figure 5.1.

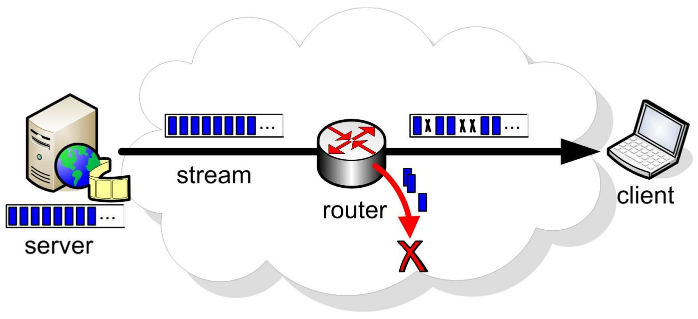


Figure 5.1: Testbed for Packet and Frame Losses Relationship. The router drops some packets and generates a packet loss rate. The client measures the frame loss rate.

We used 16 video sequences. They had different lengths, from two minutes to seven minutes. They used MPEG2 and MPEG4 encoding, with GOP sizes varying from 6 to 350 frames. And also we use two video bitrates: 512 Kbps and 1024 Kbps.

The VLC software admits a lot of streaming protocols: UDP, HTTP, HTTPS, RTP, RTSP and MMS (see VLC web site [334] for details). We streamed our 16 sequences over UDP and HTTP.

The streaming crosses an emulated network with some constraints in delay and bandwidth. The delay is 200 ms all over the tests, and the bandwidth depends on the encoding bitrate: for 512 Kbps of encoding bitrate we have a bandwidth of 717 Kbps, and for 1024 Kbps of encoding bitrate we have a bandwidth of 1280 Kbps. The network also drops packets with different predefined rates.

We can extract important information just looking at the average behavior of several tests.

- *Encoding influence in frame losses.* We streamed (using UDP and HTTP) a set of videos encoded with MPEG-2, with two bitrates (512 Kbps and 1024 Kbps), and in twenty different packet losses rates in the network. On the average we obtained 3.8% of packet losses and 47.9% of frame losses. With the same streaming protocol, bitrates and packet losses situations, we then streamed the same set of videos but this time encoded with MPEG-4. For MPEG-4 we obtain 30.2% of frame losses (there are 3.8% of packet losses again, because we are in the same network situations). Maintaining the packet losses, clearly the encoding standard has an important impact in the frame lossess.

- *Bitrate influence in frame losses.* We can observe the influence of bitrate using the same technique. For instance, if we stream (using UDP and HTTP) two set of videos encoded with MPEG-4, but with two different bitrates (512 Kbps and 1024 Kbps), we obtain, on the average, for the twenty different packet losses rates in the network: 7.4% of packet losses, 30.7% of frame losses in the case of 512 Kbps bitrate, and 21.6% of frame losses in the case of 1024 Kbps.

The streaming protocol influence in the frame loss process is more involved. For instance we can observe the behavior of HTTP and UDP streaming with different packet losses rates. In Table 5.1 we present the average results for our set of MPEG-4 videos, all with a bitrate of 512 Kbps. The network has a delay of 200 ms, and a bandwidth of 717 Kbps.

Table 5.1: Influence of HTTP and UDP streaming protocol over the frame losses.

Protocol	Packet loss rate	Frame loss rate
UDP	3	28.9
	7	27.6
	10	26.4
	15	20.9
	20	23.9
HTTP	1	32.5
	3	31.9
	4	35.3
	5	33.1
	6	46.6

As it can be observed, there is no clear relationship between IP packet losses and frame losses. For example, in UDP streaming frame losses increase when packet losses decrease. We can explain this observing the loss process in depth (for instance with our quality monitor tool presented in chapter 9). It is a consequence of the packet losses distribution, because with higher packet losses we obtain an uniform-like distribution of frame losses, instead of very high concentrated losses (as with smaller packet lost rates).

Our main conclusions is that the relationship between packet losses and frame losses depends on various factors: the video specification, the video bitrate, and the specific player software that processes the stream.

We can see that the correlation decreases with fewer packet losses, and our work will focus on the minor possible losses because we are looking for high quality. Also the correlation decreases with high encoding bitrates, and we expect that it will be bigger in next years. So, the observed correlation and these remarks clearly support our choice of frame losses as quality–impacting factor.

This low correlation between packet and frame losses is shown again, with simple tests, in our Measure and Monitor suite, presented in Chapter 9.

A final remark coming from our experience in PSQA. The accuracy of the PSQA will not change substantially if we apply the methodology correctly, the important improvement will be in the robustness with respect to changes in other factors, for instance: the streaming protocol, the encoding bandwidth, etc. Therefore, working at application level, with frame losses, we develop a general quality mapping function that can be used in applications other than our P2P for live streaming.

5.2 Network and Server Failures Effect

5.2.1 Loss Rate and Mean Loss Burst Size

In this section, we focus on two specific parameters concerning losses. We consider the loss rate of video frames, denoted by LR, and the mean size of loss bursts, $MLBS$, that is, the average length of a sequence of consecutive lost frames not contained in a longer such sequence. The $MLBS$ parameter captures the way losses are distributed in the flow.

5.2.1.1 Emulating the Losses in the Testbed: the simplified Gilbert model

To study the impact of these two parameters (LR and $MLBS$), on quality, and to generate distorted video sequences, we use a testbed. It is not easy to control the LR and $MLBS$ independently and simultaneously, therefore we present here the technique used (see for instance, [221]).

To model the loss process on an end-to-end communication we build a discrete time stochastic process $(X_1, X_2, \ldots)$ where $X_n = 1$ if the nth frame is correctly transmitted, 0 if it is lost. The i.i.d. case (a Bernoulli process) is obviously too simple because in general losses are correlated. To keep the model as small as possible (and especially, to keep the number of parameters as small as possible) we used the so-called simplified Gilbert model, following [221, 224] (we use it at the frame level, while the original model has been proposed for packet losses, but the procedure is the same). It consists in using a 2-state Markov chain for controlling which frames are lost in the flow (so, with 2 parameters; the original Gilbert model has 3 parameters [113]). Let us denote by 1 and 0 the states, with the following semantics: after a correct transmission, we will always be at state 1, and after a loss, at state 0. Figure 5.2 illustrates the dynamics of the chain; at the left, the meaning of transitions, and at the right, the chain itself. The two parameters are then

$$p = \Pr(\text{a loss after a correct transmission})$$
$$\text{and } q = \Pr(\text{a correct transmission after a loss}).$$

In [35, 298, 357] this model is shown to give a good approximation of losses on the Internet (in those papers, packet losses are considered).

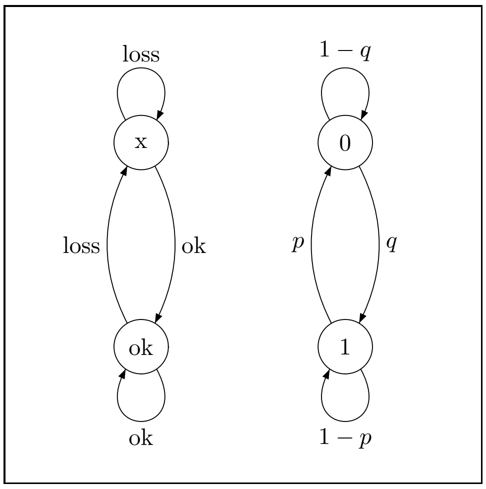


Figure 5.2: The Gilbert-like model to represent the loss process and the associated 2-states Markov chain. When in state “ok”, a transition to state “x” corresponds to a loss, and to the same state “ok” it corresponds to a correct transmission. From state “x”, the loop corresponds to a loss and the transition to “ok” a correct transmission.

The steady-state distribution of this model is given by

$$\pi_1 = q(p+q)^{-1}, \qquad \pi_0 = p(p+q)^{-1}.$$

The distribution of the length S of a generic burst of losses, considering the system in equilibrium, is geometric:

$$\Pr(S = n) = (1-q)^{n-1}q, \qquad n \geq 1,$$

with mean $\mathrm{E}(S) = q^{-1}$.

The Loss Rate LR of the flow, according to this model, and the Mean Loss Burst Size $MLBS$ of the stream, are:

$$LR = \frac{p}{p+q}, \qquad MLBS = \mathrm{E}(S) = \frac{1}{q}.$$

Reciprocally, if we have measured the Loss Rate of a stream LR ($0 < LR < 1$) and its Mean Loss Burst Size $MLBS$ ($MLBS \geq 1$), the (only) values of p and q leading to these LR and $MLBS$ are

$$q = \frac{1}{MLBS}, \qquad p = \frac{LR}{1 - LR} \frac{1}{MLBS}.$$

We implement the Gilbert model as a module in the VLC software player, in order to control the parameters, LR and $MLBS$, over original video sequences. It allow us to see its quality effects in real–time, and to save the distorted video sequences.

5.2.2 Experiment Description

We applied the PSQA technique, as explained in Chapter 4. This involved choosing four MPEG-2 video sequences, of about 10 seconds duration each, with sizes between 1.5 MB and 2.8 MB). For each sequence, we generated twenty five different configurations, where each configuration is defined by a loss rate value chosen at random with an uniform distribution between 0.0 and 0.2, and a mean loss burst size value chosen at random with an uniform distribution between 0.0 and 4.0. For each configuration, we used a simplified Gilbert model (discussed previously) to simulate a frame drop history which was applied to the original video sequences. In this way, we obtained one hundred modified video sequences with variable quality levels. In Figure 5.3 we present the final dispersion of the distorted video sequences with respect to the paramters LR and $MLBS$.

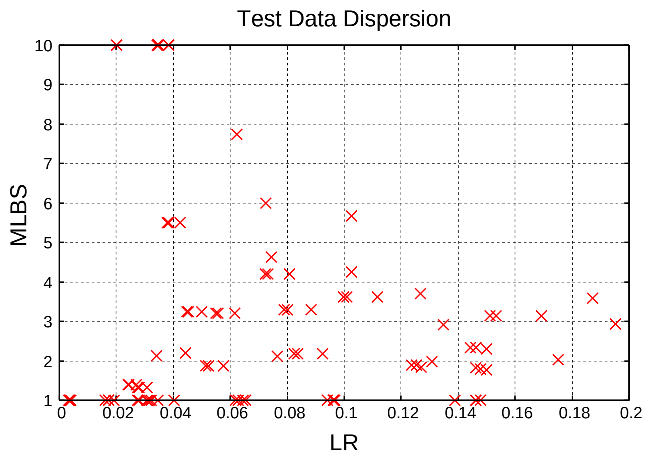


Figure 5.3: Network and Server failures Effect. Video sequences dispersion.

A group of five experts evaluated the distorted sequences and the MOS for each of the copies was computed, following the ITU-R BT.500-11 [171] norm. See Figure 5.4 for the MOS value and the experts' votes for each sequence. The average standard deviation of the subjects (computed with Equation 4.2) is $\overline{\delta} = 0.13$. It means that on the average the experts differed in $\pm 13\%$ of their assessments.

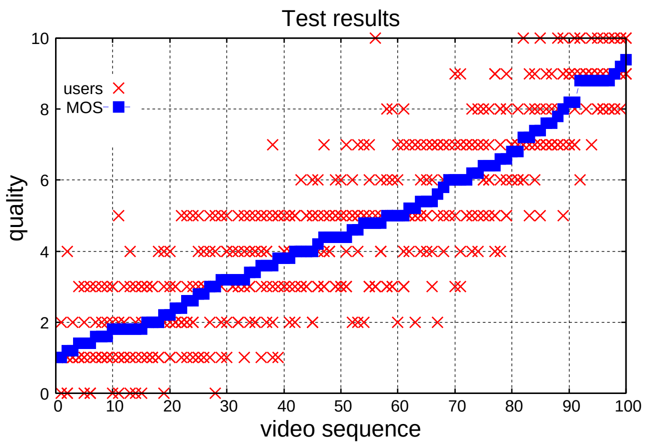


Figure 5.4: Network and Server failures Effect. Subjective test results for building the PSQA metric.

Next, we use the MOS value for each of the sampled points as inputs in order to approximate a quality function of the two variables, LR and $MLBS$.

5.2.3 PSQA *Simple Function*

To obtain the perceptual quality as a function of the parameters that approximate smoothly the empirical test values obtained previously, we train a Random Neural Network (RNN) following the procedure described in Section 4.5.

In this quite simple function (because of the fact that we use only two input variables), we use a three–layer feed–forward neural network. It consists of two neurons in the input layer, seven neurons in the hidden layer, and an output neuron. See the simple RNN topology in Figure 5.5).

To determine the number of neurons in the hidden layer we follow the procedure described in Section 4.5.3.2 which iterate with different configurations and chooses the best one in terms of performance. To measure the performance we use the Mean Square Error (average squared difference between the values of the function and the MOS given by real human observers). We use eighty-five distorted video sequences to train the RNN, and the remaining sequences (fifteen) to validate it. We obtain a global performance of $\text{MSE}_{\text{simple}} = 0.023$ for the validation sequence set. Comparing the estimation of the standard deviation ($\delta_{\text{simple}} = \sqrt{\text{MSE}_{\text{simple}}} = 0.15$) with the average standard deviation of the subjects ($\overline{\delta} = 0.13$) we conclude that our *Simple Function* behaves as to an average human subject.

In Figure 5.6 we can see the PSQA function. For ease of reading, we extrapolated the curve to the borders, but observe that the data are accurate and used on an internal region ($[1\%,\ 20\%]$ for LR, and $[1,\ 10]$ for the $MLBS$)[1]. We can see that quality is monotone in the two variables,

[1]The perceptual quality decreases quickly with high loss rate values ($LR > 20\%$), therefore it is possible to

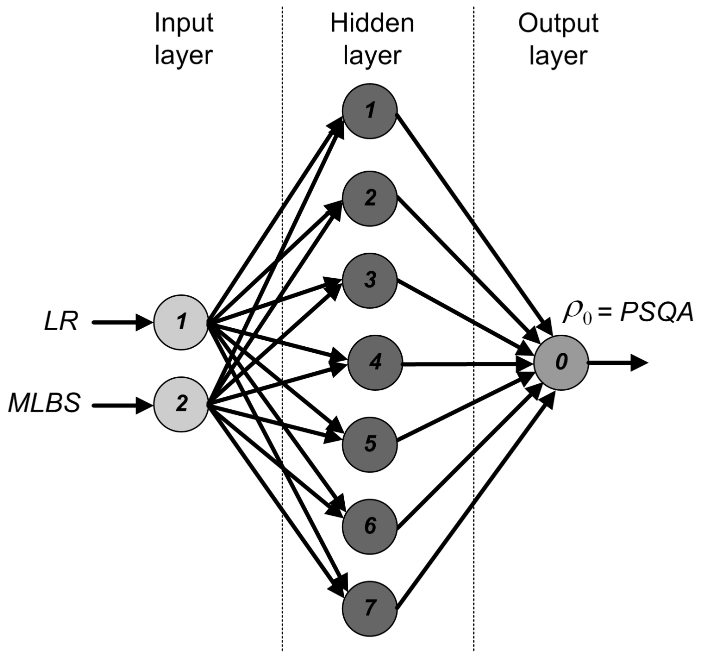


Figure 5.5: Simple RNN Topology.

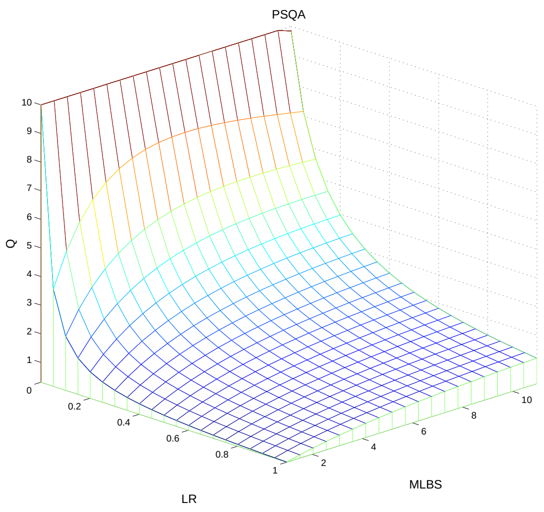


Figure 5.6: The PSQA *Simple Function* (mapping LR and $MLBS$ into perceived quality) after trained and validated.

and in particular increasing with the $MLBS$, meaning that humans prefer sequences where losses are concentrated over those where losses are more isolated.

Training and validating is only made once, when building the mapping function. In operation, the use of PSQA is very simple: we measure the frame loss rate LR and the mean loss burst size $MLBS$ in a short period, and we use the PSQA as an *instantaneous* quality value. This PSQA *Simple Function* is used in the rest of our work as a very simple approximation to the quality. For instance it is used: to verify the usefulness of our streaming mechanism (Chapter 6), to ensure a level of quality in a context with server failures (Chapter 7), and in a prototype monitor tool to measure the perceived quality of a Video Delivery Network in real–time (Chapter 9).

5.2.4 PSQA *Minimal Closed–form Function*

Using the same distorted video database as the PSQA *Simple Function*, we also train a *Minimal Closed–form Function* coming from a two–layer RNN topology. That is, a Neural Network without hidden neurons, just two input neurons and one neuron at the output layer (see the RNN minimal topology in the Figure 5.7).

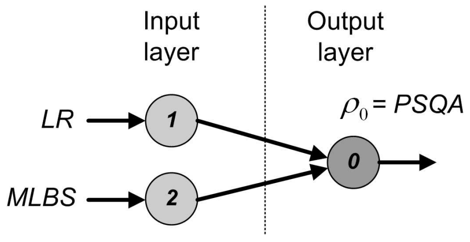


Figure 5.7: Minimal RNN Topology.

This topology allow us to approximate the perceived quality with a very simple formula:

$$Q(LR, MLBS) = \frac{a\,LR + b\,MLBS}{c\,LR + d\,MLBS + \nu}, \tag{5.1}$$

where $a = 0.00142128$, $b = 0.47242076$, $c = 0.00142128$, $d = 0.47242076$ and $\nu = 0.01$.

Using a two–layer architecture, the validation phase led to a Mean Square Error $MSE_{\text{minimal}} = 0.041$. Using the MSEs, we can compare the estimated standard deviation of the *simple function* and the *minimal function*:

$$\delta_{\text{minimal}} = 0.20, \qquad \delta_{\text{simple}} = 0.15,$$

showing that this minimal funtion has 5% less precision than the simple function built before. The minimal form is helpful in some applications, where the precision is not critical and a

use the extrapolation in this region with an acceptable accuracy.

closed–form is useful, like our real–time algorithm to define the buffer size of a video player where the video streaming sources fail (Section 7.3).

In Figure 5.8 we can see the obtained function. This function is valid in the specific considered range of the input variables (that is, in the range corresponding to the values of those variables in the sequences that were used to learn from real human behaviors). The interval where the chosen variables were considered are the ones indicated in the figure axis.

Observe that the function preserve the characteristics of the previously defined simple function (Figure 5.6), where the quality is monotone in the two variables.

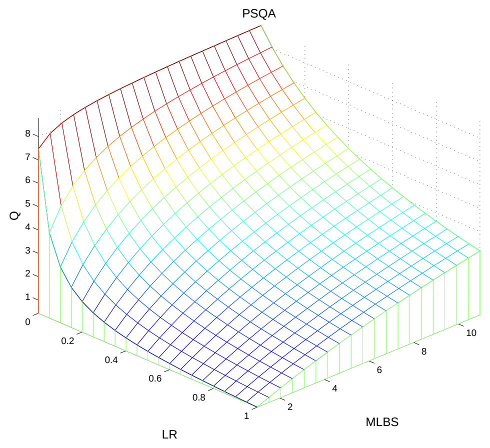


Figure 5.8: The PSQA curve in our setting. Based on a two-layer RNN architecture.

5.3 Adding Video Source Motion Effect

In this section we study the impact of video motion on the perceived quality. This is motivated by the idea that for a given network or server failure, the perceived quality can be different depending on the intensity of video motion. For example, an intuitive prediction would be that for small loss rates, tiny errors are more noticeable if the amount of motion is lower while if the movement activity is higher, errors are probably hardly noticed.

In our previous study, we analyzed how the network and server failures affect the quality. We observe the dependence of the quality with respect to the losses, in particular, the dependence with respect to the frame loss rate LR and the mean lost burst size $MLBS$. As expected, the sensitivity of the quality with respect to the LR is much higher than the sensitivity with respect to the $MLBS$. Therefore, looking for a more sensitive classification of frame loss distribution, in this section we do not consider the $MLBS$ as a quality–affecting parameter and concentrate our efforts in the losses of each type of frame.

As we explained in section 2.1, in MPEG-2 and MPEG-4 specifications, there are three main frame types: the Intra frames (I), the Predicted frames (P), and the Bidirectional or Interpolated frames (B). In both specifications, the idea behind compression is the same: the numerous temporal redundancies between frames are exploited using the P and B frames, by means of using motion estimation and compensation.

Besides the frequency and mean size of each kind of frame, they also differ in how the loss of each of them influences the quality. As there is inter-dependency among them for performing the codification process, if the most important frame is lost the codification is strongly affected, diminishing the perceived quality.

In this section we study the impact on the quality of the combination of the loss rates per frame type, instead of previous global frame loss distribution. Also this loss process classification fits better with our main objective of studying the video source motion effect, because P and B frames are basically an abstraction of the video motion.

5.3.1 Quality-affecting parameters

As said before, the video motion is represented in MPEG standards mainly in P and B frames, using motion vector descriptions. Transmitting more information in P and B frames implies more motion vector descriptions; and therefore, more motion activity. Following this idea, we study and compare the effect of bytes lost of each kind of frame, instead of just its frequency of loss. We analyze the effect of the following distribution parameters on the quality:

I-Frames Loss Rate. In this case, ten seconds videos are considered, so there is basically only one GOP in each sequence, which implies that this parameter can be only 1 or 0. We denoted it by LR_I.

P-Frames Loss Rate. This parameter measures the P-Frames bytes lost in one GOP over the total amount of P-Frames bytes in the GOP. We tested another option consisting in measuring the percentage of P-frames lost per GOP. It is denoted by LR_P.

B-Frames Loss Rate. The same as for P-Frames, but in this case for B-Frames. It is denoted by LR_B.

These first three parameters are loss rates, coming from network or server failures.

With regard to the source parameters, we study the effect of two raw (and simple) representations of the video motion activity:

Average GOP Size. The encoder decides when I-Frames are introduced, depending on the video characteristics, so this parameter reflects if the video sequence has many sudden changes or not. We denoted it by $\overline{\text{GOP}}$. Usually there is a GOP per each video scene, so the average GOP size is close related to the number of scenes in the video, which are the most important motion activity in a video.

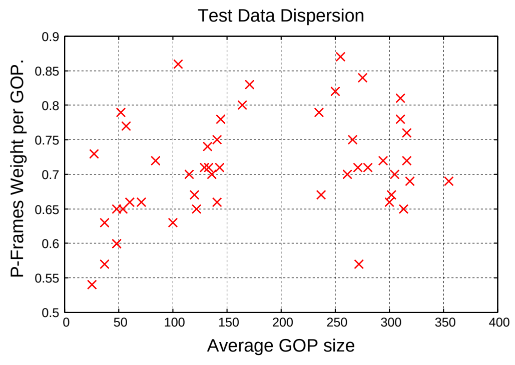
P-Frames Weight per GOP. This parameter indicates the weight in bytes of P-Frames in each GOP, which is also something that depends on the video characteristics used by the encoder to decide how to encode the sequence. The P-Frame weight per GOP represents, in a very simple way, the motion inside a scene.

Next we will show the impact of these new input parameters into the perceived quality. Appendix A compares our simplest video motion parameters indicators with more sophisticated motion activity measures, like the MPEG-7 [51] motion descriptor and other related works. The main conclusion of this appendix is that a very low improves on the quality assessment precision is obtained using complex metrics to compute the motion activity. In our context, a real–time application, they not justify the computational effort.

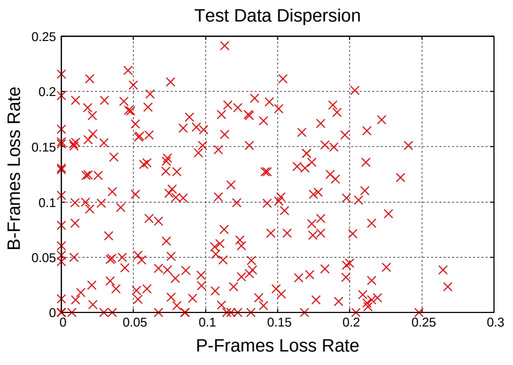
5.3.2 Experiment Description

To apply our PSQA technique, we used fifty MPEG-4 video sequences[2], of about 10 seconds each, with sizes between 238 KB and 2.78 MB. These video sequences were chosen in order to cover the source parameter ranges. The dispersion of the source parameters is shown in Figure 5.9(a).

With this set of fifty videos, a set of 204 distorted sequences was generated, for covering the different possibles values for all the parameters related with the loss process (LR_I, LR_P and LR_B). We used a simplified Gilbert model (discussed previously) to simulate the frame drop history which was applied to the original video sequences. Half of the distorted videos included the I frames ($LR_I = 0$), and the other half didn't ($LR_I = 1$). This is because in a MPEG-4 video sequence of 10 seconds, there are at most only one I frame. With respect to LR_P and LR_B, the final dispersion is shown in Figure 5.9(b).



(a) Average GOP Size and P-Frames Weight per GOP.



(b) LR_P and LR_B.

Figure 5.9: Motion Effect. Video sequences dispersion.

Then, a group of ten experts evaluated the 204 distorted video sequences and the MOS for each of the sequences was computed, following the Section 4.4 (Single Stimulus (SS) subjective test of the ITU-R BT.500-11 recommendation [171]). See Figure 5.4 for the MOS value and the experts' votes for each sequence.

[2]We used the Xvid [354] codec implementation of MPEG-4.

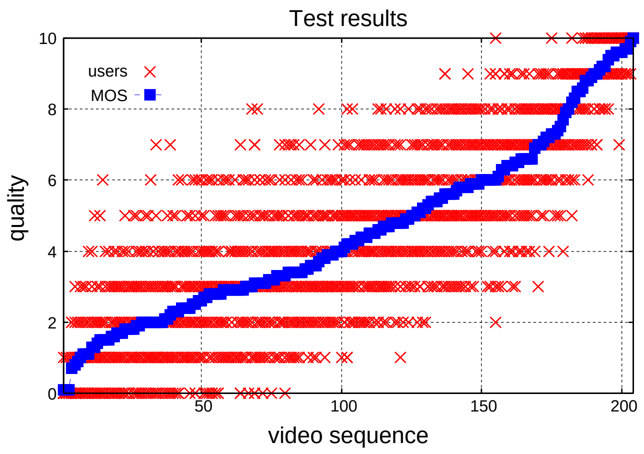


Figure 5.10: Motion Effect. Subjective test results for building the PSQA metric.

The average standard deviation of the subjects (computed with Equation 4.2) is $\overline{\delta} = 0.15$. It means that, on the average, the experts differ in $\pm 15\%$ of their assessments.

In next Subsection we build our PSQA function using this data

5.3.3 PSQA *Complex Function*

Using the subjective test results, we defined four sets of possible input parameters, in order to determine which one approximates better the perceived quality of end users. In Table 5.2 we present the input parameters for each function. NET and NET2 consider only network parameters, while NETSRC and NETSRC2 consider network and source parameters. The other option is to consider the loss rates as bytes loss (NET and NETSRC) or frames loss (NET2 and NETSRC2).

Table 5.2: Input parameters for each function.

Name	LR_I	LR_P	LR_B	$\overline{\text{GOP}}$	P-Frames Weight
NET	yes/no	bytes	bytes	-	-
NET2	yes/no	frames	frames	-	-
NETSRC	yes/no	bytes	bytes	$\surd$	$\surd$
NETSRC2	yes/no	frames	frames	$\surd$	$\surd$

In all cases we use three–layer feed–forward RNNs to approximate the quality function. To determine the number of neurons in the hidden layer we followed the procedure described in section 4.5.3.2 which iterates with different configurations and chooses the best one in terms of performance (i.e. with the minimal value of MSE in the validation stage). For training, approximately 80% of the data was used (160 sequences), while the other 20% (44 sequences) enables the validation stage.

The best results are resumed in Table 5.3. Tests showed no significant differences between the cases which consider bytes loss and the ones with frames loss, which can be seen comparing NET and NET2 and the same between NETSRC and NETSRC2. A small improvement is seen in the case of NETSRC2, considering frames loss, while a more important improvement is seen comparing the systems which only use network parameters with the ones which add source parameters.

Table 5.3: Best RNN for each input parameter set.

Name	Hidden neurons	MSE validation
NET	2	0.0296717
NET2	6	0.0288654
NETSRC	3	0.0281829
NETSRC2	3	0.0254898

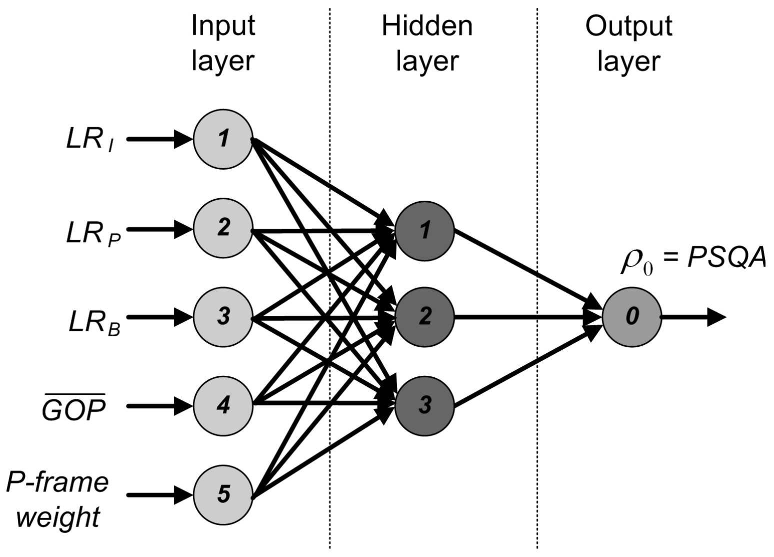


Figure 5.11: Complex NETSRC2 RNN Topology.

We choose the best performance RNN, NETSRC2, as our *Complex Function*. See the complex RNN topology in Figure 5.11. We obtain a global performance of $\text{MSE}_{\text{complex}} = 0.025$ for the validation sequence set. Comparing the estimation of the standard deviation ($\delta_{\text{complex}} = \sqrt{\text{MSE}_{\text{complex}}} = 0.16$) with the average standard deviation of the subjects ($\overline{\delta} = 0.15$) we conclude that our *Complex Function* behaves very similar to an average human subject.

With our *Complex Function*, the loss rate effect of each frame type is illustrated in Figures 5.12, 5.13(b) and 5.13(a). The figures show the perceived quality as a function of two parameters, the other three parameters (hidden in the figures) are fixed with the following values: there are no hidden losses, the average value is used for the source parameters ($\overline{\text{GOP}} = 174$ and P-Frames Weight $= 0.7007$).

In Figure 5.12, quality degrades quickly with an increment in the loss rate of frames I and P. For example, for $LR_P \geq 10\%$ the quality is less than 6 (between good-fair) and the impact of P-frames' losses is a bit higher than for I-frames. Figures 5.13(b) and 5.13(a) show that the quality degrades slowly with an increment in the loss rate of B-frames, as expected.

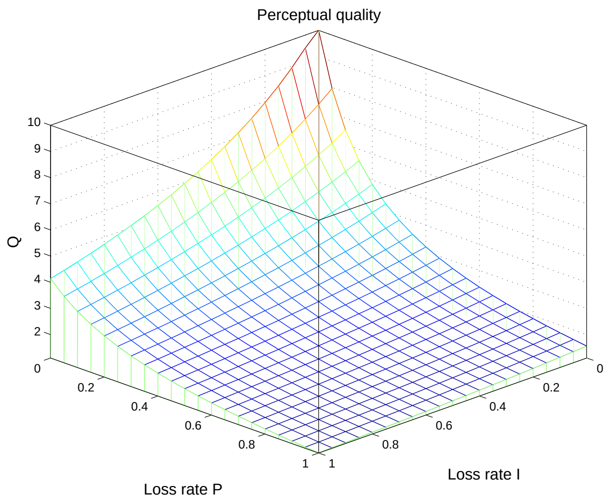
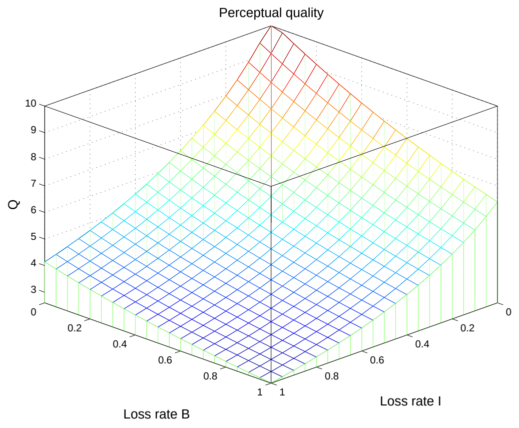
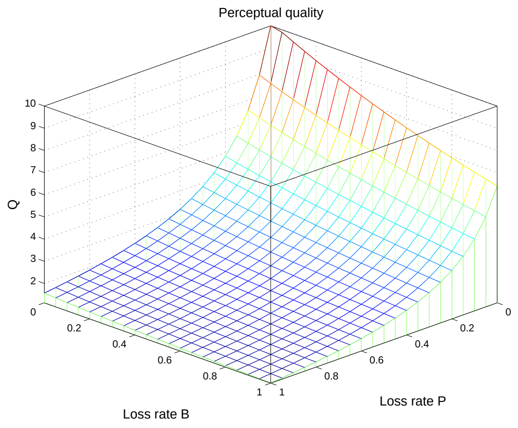


Figure 5.12: The quality degrades quickly with an increment in the loss rates of frames I and P.



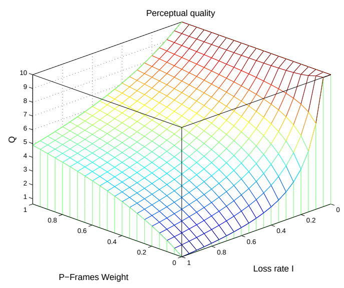
(a) LR_B vs. LR_I.



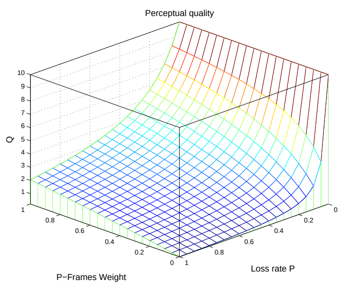
(b) LR_B vs. LR_P.

Figure 5.13: The quality degrades slowly with an increment in the loss rate of frames B.

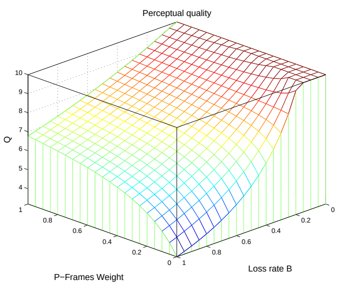
The source parameters do not directly degrade the quality, but they have an impact on how losses affect users' perception. To observe the effect of source parameters into quality, we show their impact with respect to each kind of frame lost independently. In Figures 5.14(a), 5.14(b) and 5.14(c) we show the influence of the P-frames weight per GOP. As we expect, with higher ratios of P-frames weight, there is less impact of I-frames and B-frames losses into the quality.



(a) P-Frames Weight vs LR_I.



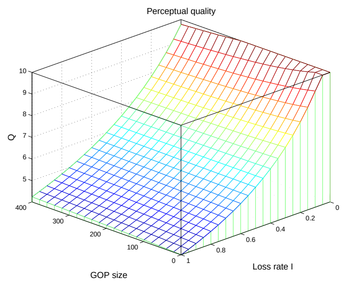
(b) P-Frames Weight vs LR_P.



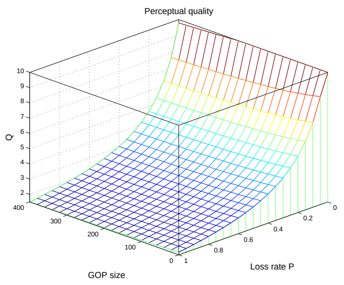
(c) P-Frames Weight vs LR_B.

Figure 5.14: P-Frames Weight influences into the quality to different kind of frame losses.

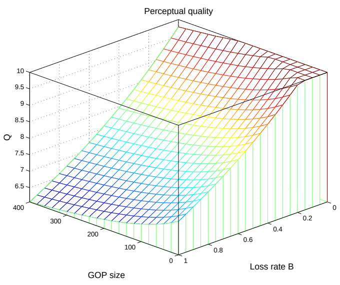
In Figures 5.15(a), 5.15(b) and 5.15(c) we show the influence of the average GOP size. We see that there is no observable influence of GOP size into the quality in a scenario of losses.



(a) GOP size vs LR_I.



(b) GOP size vs LR_P.



(c) GOP size vs LR_B.

Figure 5.15: Average GOP size influences into the quality to different kind of frame losses.

The *Complex Function* is used in the rest of our work. In section 7.2, we find an optimal streaming method, from the user point of view (evaluating the perceived quality with this function), in a context of server failures. We also use it in our P2P prototype, GOL!P2P, for robust delivery of high quality live video (see chapter 10). In this prototype we measure the perceived quality by each user in real–time, and take control actions with this information.

5.4 Summary and Conclusions

In this chapter we studied the effects of network and source parameters on the perceived video quality evaluated using the PSQA methodology (explained in Chapter 4). We applied the PSQA technique in two scenarios. First, we analyzed the network and server failures impact on quality by means of the study of frame lost effect. Second, we improved the measure accuracy considering the influence of video source' motion in quality.

This study extends and generalizes previous works on PSQA, in particular for video quality assessment, evaluating the effect at the frame level (instead of packet level studied previously). The mapping functions (from quality–affecting parameters into perceived quality) obtained in this chapter will be used in the rest of this work for network design purposes and for performance evaluation.

Part III

PEER-TO-PEER

Chapter 6

Multi-Source Streaming Technique

In this chapter we address the problem of designing a content distribution technique for real-time video delivery through a P2P system on the Internet. The main advantage of using the P2P approach is to exploit the available bandwidth unused by the set of machines connected to the network. The main difficulty is that these machines are typically highly dynamic with respect to the network, they continuously enter and leave it. To deal with this problem, we explore a multi-source approach where the stream is decomposed into several redundant flows sent by different peers to each client. Using the PSQA technology (detailed in Chapter 4) for evaluating automatically and accurately the perceived quality at the client side, this chapter focuses on the consequences that the way the stream is decomposed has on the resulting quality.

Our approach allows thus to do the design by addressing the ultimate target, the perceived quality (or Quality of Experience), instead of the standard but indirect metrics such as loss rates, delays, reliability, etc. The main contribution of this chapter is to provide a global methodology that can be used to design such a system, illustrated by looking at three extreme cases. Another important feature of our technique is that it results in a very low signaling cost (overhead), in contrast with Bittorrent-like approaches.

The chapter is organized as follows. Section 6.1 introduces multi-source streaming techniques. In Section 6.2 we develop models to estimate the streaming performance, in a environment with server nodes failures. The conclusions of the chapter are summarized in Section 6.3. This chapter has been partially presented at the IEEE Global Telecommunications Conference (GLOBECOM'07)[282] and Beijing-Hong Kong International Doctoral Forum - Web, Network, and Media Computing (BJ-HK Phd Forum'07)[275] (listed at the Introduction as [globecom07] and [bj-hk07]).

6.1 Introduction to multi-source streaming

In this chapter, we are interested in some aspects related to the use of a P2P architecture to distribute live video. Using a P2P infrastructure for video distribution looks like a good idea due to the high requirements in terms of bandwidth of these applications. The main problem is how to provide good quality levels in a context where this quality depends on other clients that are delivering the stream, and given the fact that users connect and disconnect very frequently. The main idea that has been considered to deal with these problems is to build the systems using some redundancy in the signals. In this thesis we explore one of ways to reach this objective: multi-source streaming. This means that the live video stream is received by the client from flows sent by many sources simultaneously. This needs some degree of intelligence on the senders that build a set of sub-streams from an original one, allowing to re-compose the stream to be played with the different received components, and some degree of intelligence on the client side, for performing this last task, and perhaps for building a satisfactory stream even when some of those components are missing. This approach allows for a large flexibility of the system, modulated by the dynamics of the network. In particular, it is in principle possible to increase the number of sources and/or the amount of redundant information sent through the network; this flexibility can be used as a tool to deal with the problem of nodes leaving the network and causing partial signal losses to some clients. We will say that a node *fails* to refer to the fact that it leaves the network.

The system's flexibility must be carefully tuned in order to get a satisfactory quality level at a minimal cost. The usual approach is to focus on a well chosen metric, that we know plays an important role in quality, such as the loss rate of packets, or of frames. In this work we instead address the problem of measuring *perceived* quality by means of the PSQA technology (see Part II). PSQA is a general procedure that allows the automatic measure of the perceived quality, accurately and in real-time.

Summarizing, in order to face the high dynamics of a P2P live video distribution network, we explore a multi-path approach where (i) the stream is decomposed in some way into several flows, (ii) each client receives several flows following different paths and sent from different other clients, (iii) the client is able to reconstruct the stream from the whole set of received flows and possibly from part of them; moreover, (iv) the system measures automatically the perceived quality at the client continuously, and takes its decisions (basically, periodically rebuilding the architecture of the network) using these values.

6.1.1 Simple extreme multi-source methods: single, copied and splitted flows

The main architecture we are considering is the following one. Some server producing a live video stream splits this stream into K flows, with some amount of redundancy in them (that is, together they can transport "more information" than contained in the original video signal), and it sends each of these flows to a specific set of clients. The clients in turn send the received flows to other nodes. The architecture must ensure that each client receives the different flows from different nodes. So, from the client's point of view, we have a multi-source delivering system.

Please note that, for a simplified notation, and with the idea of the client point of view, in our explanation we have one client and a set of K servers, where actually the servers are other peers of the P2P network, and also the client can serve to other peers.

The simplest situation is when there is a *single* server node which sends all the streaming information to the clients (see Figure 6.1). Let us consider instead the case where a set of

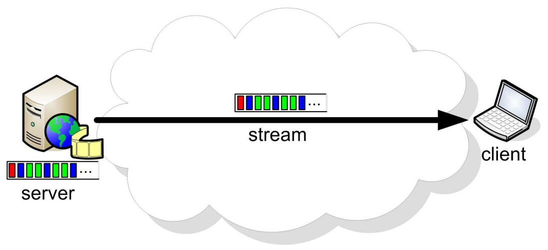


Figure 6.1: *Single* source streaming method ($K = 1$).

servers will send more than one flow composing the original signal. The quality perceived at the client node will be a function of the policy being used to distribute the streaming among the different flows, of the number of flows, of the degree of redundancy, and losses due to transport network conditions or to instabilities at the P2P server nodes. An important aspect is the degree of redundancy being employed; in this case of multiple servers, the extreme cases are to completely replicate all the information, or to apply no redundancy at all. In the first case, the policy being applied is *copy*: each of the server nodes sends the full streaming to the client, which will then be less prone to quality problems caused by frames lost by communication problems. This is a full redundant scheme where the client receives many copies of the complete flow (see Figure 6.2).

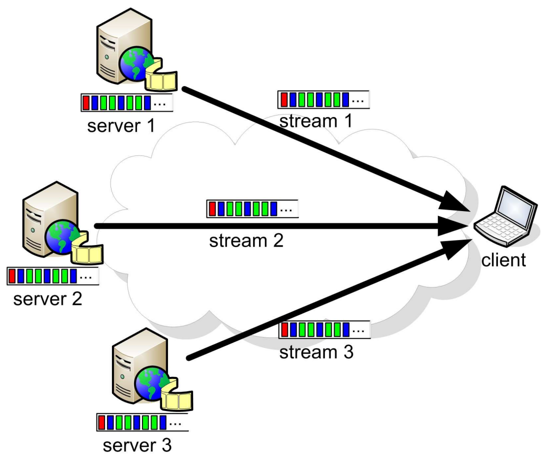


Figure 6.2: Multi-source streaming *copy* method. Three servers send the original stream to the client ($K = 3$).

In the second case, we have a *split* policy: each server sends a fraction of the streaming information, without any redundancy, and the loss of information at any of these flows will imply also losses at the client. Figure 6.3 represents this scheme. More precisely, we will consider the case of sending frame 1 in flow or sub-stream 1, frame 2 in flow 2, up to frame K in flow K, then frame $K + 1$ in flow 1, etc..

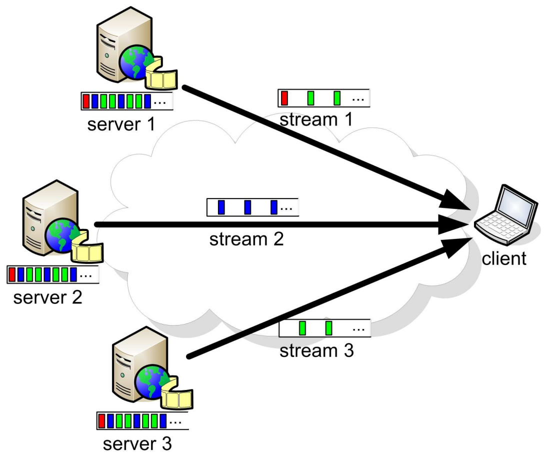


Figure 6.3: Multi-source streaming *split* method. Three servers send three different and no redundant sub-streams to the client ($K = 3$).

A situation between these two extreme policies is to split the stream into K redundant sub-streams, with the necessary redundancy to avoid losses during a single server disconnection. This means that each frame is sent exactly twice (by two different servers to a given client). Figure 6.4 represents this scheme.

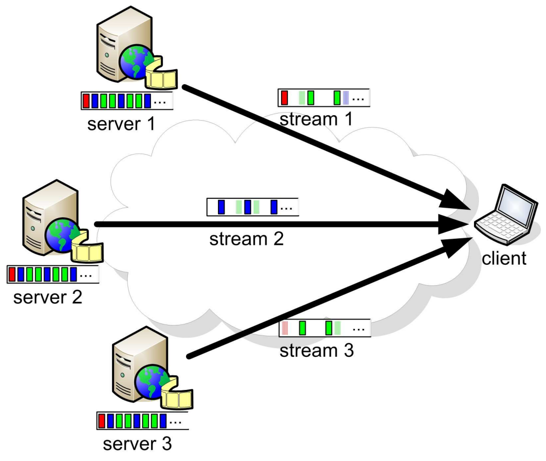


Figure 6.4: Multi-source streaming *redundant split* method. Three servers send three different redundant sub-streams to the client ($K = 3$).

Obviously, we can expect that in a scenario subject to failures, with more redundancy better quality will be achieved. The cost of this improvement is a larger bandwidth consumption. If the bandwidth of the original stream is B, then the bandwidth of the *copy* method is KB, B for the simple *split* method, and $2B$ for the *redundant split* method.

Although in this chapter we concentrate on these extreme policies (either zero redundancy, full replication of the information sent by each server, or redundancy to avoid losses during one server fault), it is clear that the degree of redundancy admits many other possibilities in-between. In next chapter we present an optimal streaming technique (between these simple cases), which also considers the heterogeneity of the peers dynamics.

6.1.2 Related multi-source strategies

Previous studies (see [18]) show that in 30-80% of the cases for a given transmission there is at least an end-to-end path better (from the quality perspective) than the one chosen by usual protocols. This is the main idea behind the multi-path and path diversity studies. In the particular case of video flows, it is easier to recover from isolated losses than from consecutive ones [17]. The basic difference between the multi-path and multi-source concepts is that in the former case we consider a single source sending data to the destination following different paths. The main difficulty here is the routing aspects: we must specify and control which path must be followed by which packet in the flow. The multi-source approach implies that there are multiple and independent sources for the signal, and that some general scheme allows the receiver to get the stream from a set of servers. Since they are different, the path that the packets will follow would be a priori different, without any need of acting on the routing processes. Of course, in practice, the different paths could share some nodes, which are good candidates to become bottlenecks of the communication system. Detecting such cases is an active research area [291].

Focusing on P2P applications, observe first that many P2P data sharing and distribution systems implicitly assume that a sending peer is capable of supporting one or more receiving peers. It can be complicated to implement this in practice, especially for video streaming.

The same proposals suggest to implement the multi-source streaming at the video encoding level, using the Multiple Description Coding (MDC) [16, 118, 191, 204, 304]. MDC is a video coding technique, which generates an arbitrary number of sub-streams (called descriptions, in this context) from one single stream. Each sub-stream can be decoded independently from the others; however, the quality improves when more descriptions are decoded in parallel. In MPEG-2 and MPEG-4, it is possible to encode a video in layers. Similar to MDC, layered coding techniques generate an arbitrary number of layers, and the quality improves when more layers are decoded. But here, there are a base layer and a series of hierarchical enhancement layers. In order to decode a layer is necessary to decode the layer immediate before (they are not independent of each other). MDC and layered coding are developed to allow a finer control of the streaming and to provide error resilience to media streams. Both have high complexity, with less efficient compression. Therefore, they are not widely deployed, and there are no open-source implementations.

Other works propose to implement the multi-source streaming at the transport level, for instance using Network Coding [6, 55, 56]. Network Coding is a method based on information and coding theories, that allows to mix the data at intermediate network nodes, while the receiver extracts form these data the original message. It attempts to maximize the information flow in a network. Network Coding is not necessarily suitable for P2P networks; the dynamics of the overlay topology, and the time needed by peers to decode data encoded with this procedure, suggest that it is difficult to use it in this context, and especially for live streaming [54].

But, as we will see in next chapter, the election of the servers that are part of the multi-source streaming is important, especially in a P2P context. It has been shown that peers are heterogeneous in their capability and/or willingness to contribute resources to other peers [299]. Few systems considered the problem of selecting multiple supplying peers for a receiver based on peer heterogeneity as well as network tomography information (for an example, CollectCast). Nguyen and Zakhor [234] propose streaming video from multiple sources concurrently, thereby exploiting path diversity and increasing tolerance to packet loss. They subsequently extend their work to use Forward Error Correction [233] encodings. However, the framework is not designed for P2P environments. Therefore, it does not address the selection and dynamic switching of senders. Rodrigues and Biersack [273] show that parallel download of a large file from multiple replicated servers achieves significantly shorter download time. The subsets of a file supplied by each server are dynamically adjusted based on network conditions and server load. However, their work targets bulk file transfer, not real-time media streaming. Moreover, it does not consider the sender selection problem and it does not leverage network tomography techniques. In [16, 18] it is proposed to use striped video and MDC to exploit path diversity for increased robustness with respect to packet loss. The idea is to build an overlay composed of relays, and having each stripe delivered to the client using a different source. The papers examines the performance of the MDC, but they don't describe an infrastructure to actually forward the stripes to the clients.

6.2 Utility of multi–source streaming techniques: Its impacts on Quality

This section focuses on the analysis of the impact on the perceived quality, as captured by the PSQA metric, of multi-source streaming techniques. Our main goal is the description of a global methodology that can be used to design such a P2P distribution algorithm. This is illustrated by considering the three extreme cases: *single*, *copy*, and *split*. The quantitative evaluation of these models not only can give some insights into QoE characteristics of multi-source streaming in a P2P network, but can also serve as bounds for the expected behavior of other policies with an intermediate replication level.

Moreover, we want to verify the utility of the multi–source streaming approach, in the sense of improving the perceived quality. We focus on the most important global quality–affecting factors, the two specific parameters concerning losses [221]. We consider the loss rates of video frames, denoted by LR, and the mean size of loss bursts, $MLBS$, that is, the average length of a sequence of consecutive lost frames not contained in a longer such sequence. The $MLBS$ parameters capture the way losses are distributed in the flow. Therefore, we use the PSQA *simple function* to approximate the perceived quality from these two parameters (see Section 5.2 to know how this function was obtained).

6.2.1 Multi-source Streaming Models

We develop here stochastic (Markovian) models for the frame loss process in multi-source streaming, in the three considered cases.

We start from the *single* source case, using one of the simplest models considered in the literature, which nevertheless can take into account the two chosen parameters of the loss process, LR and $MLBS$. We do not differentiate among losses due to the server node itself and losses due to the underlying Internet connection between server and client; we just apply a descriptive model, whose parameters can be completely characterized by the observed values of the above mentioned LR and $MLBS$ parameters[1].

6.2.1.1 Streaming from a single source

To model the loss process on an end-to-end communication we used the so-called simplified Gilbert model, presented in Section 5.2.1.1. It consists of using a 2-state Markov chain for controlling which frames are lost in the flow (see Figure 5.2 for a illustration of the chain dynamics). It has two parameters:

$$p = \Pr(\text{a loss after a correct transmission})$$

and

$$q = \Pr(\text{a correct transmission after a loss}).$$

Summarizing, the steady-state distribution of this model is given by $\pi_1 = q(p+q)^{-1}$, $\pi_0 = p(p+q)^{-1}$. The distribution of the length S of a generic burst of losses, considering the system in equilibrium, is geometric: $\Pr(S = n) = (1-q)^{n-1}q$, $n \geq 1$, with mean $\mathrm{E}(S) = q^{-1}$. Finally, the Loss Rate LR of the flow, according to this model, and the Mean Loss Burst Size $MLBS$ of the stream, are

$$LR = \frac{p}{p+q}, \quad MLBS = \mathrm{E}(S) = \frac{1}{q}.$$

6.2.1.2 Sending K copies of the stream

Assume there are K copies of the same stream travel following independent and stochastically equivalent paths to the same terminal. The loss process at any of the K streams is represented by the model previously described, with parameters p and q. It is clear that the receiver will observe the loss of a frame if all the copies of the frames are lost. If LR_K^{copy} denotes this global Loss Rate, we then have

$$LR_K^{copy} = \left(\frac{p}{p+q}\right)^K = LR^K.$$

If S_K denotes the size of a generic burst of losses, we have

$$\Pr(S_K = n) = \left[(1-q)^K\right]^{n-1}\left[1-(1-q)^K\right],$$

[1]In a P2P network, we expect that the losses due to server disconnections dominate those due to the best-effort Internet. We keep the two possibilities for generality reasons.

giving a global Mean Loss Burst Size $MLBS_K^{copy} = \mathrm{E}(S_K)$ as follows:

$$MLBS_K^{copy} = \frac{1}{1-(1-q)^K} = \frac{1}{1-(1-MLBS^{-1})^K}.$$

6.2.1.3 Complete split of the stream into $K \geq 2$ sub-streams

In the other extreme case considered in this section, we have K sub-streams transporting each a $1/K$ fraction of the frames in the following way: frame 1 goes through sub-stream 1, frame 2 through sub-stream 2, until frame K going through sub-stream K, then frame $K+1$ through sub-stream 1, etc. In general, frame n is sent by sub-stream $((n-1) \mod K)+1$.

Again, the loss processes at any of the K streams are stochastically equivalent, and are represented by the previous model, with parameters p and q. We obviously have here, for the Loss Rate of this scheme, the same value as for the single source case:

$$LR_K^{split} = LR = p(p+q)^{-1}.$$

The evaluation of the Mean Loss Burst Size is more involved. The probability $\Pr(S_K = 1)$ is the probability that after a loss (which follows a correct transmission), next packet is not lost. For $\Pr(S_K = 2)$, observe that a burst has size 2 if after the first loss (following a correct transmission), we have a loss and then a correct transmission. And so on.

Consider first the case of $K = 2$. The probability $\Pr(S_2 = 1)$ is $1-p$, because since $K = 2$ and the case of interest ($S_2 = 1$) corresponds to the sequence '1 0 1', the state of the simplified Gilbert model in the sub-stream corresponding to the last '1' is '1' (the state after a correct transmission); the associated conditional probability is $1-p$. So, $\Pr(S_2 = 1) = 1-p$. For $S_2 = 2$, we have the pattern '1 0 0 1' and we want to evaluate the probability of observing the suffix '0 1' knowing that the previous pattern (the prefix) is '1 0'. We have $\Pr(S_2 = 2) = pq$. For size 3, we obtain $\Pr(S_2 = 3) = p(1-q)q$. Following the same reasoning, we have

$$\Pr(S_2 = n) = p(1-q)^{n-2}q, \qquad n \geq 2.$$

From this expression and after some algebra, we get

$$\mathrm{E}(S_2) = 1-p+\sum_{n\geq 2} np(1-q)^{n-2}q = \cdots = 1+\frac{p}{q}.$$

Let us illustrate the evaluation in the case of $K = 4$. The general case is done the same way. The probability $\Pr(S_4 = 1)$ is the probability that after a loss (which follows a correct transmission), next packet is not lost. The probability of this correct transmission depends on the state of the corresponding sub-stream: if it is '1', the value is $1-p$; if it is '0', the value is q. Using the steady-state distribution of the simplified Gilbert model, we have

$$\Pr(S_4 = 1) = \pi_1(1-p)+\pi_0 q = \pi_1 \left(= \frac{q}{p+q}\right).$$

For $\Pr(S_4 = 2)$, we need the probability of a loss and then a correct transmission after a first loss that follows a correct transmission. The last event (the correct transmission closing the burst) has probability π_1, as we have just seen. For the first one, conditioning again on the state of the corresponding sub-stream, the probability is $\pi_1 p + \pi_0(1-q) = \pi_0 = p(p+q)^{-1}$. Concluding,

$$\Pr(S_4 = 2) = \pi_0\pi_1.$$

Now, for $S_4 = 3$, there is a change. Denoting again by '1' a correct transmission and by '0' a loss, a burst of losses with size 3 corresponds to a partial sequence '... 1 0 0 0 1 ...'. The probability $\Pr(S_4 = 3)$ is the probability of the sequence '0 0 1', knowing that it is preceded by the prefix '1 0'. The probability of the first two 0s in '0 0 1' is π_0^2, as seen before. For the last element, 1, its probability is now $1-p$, since we know that on the same sub-stream the state was necessary 1 after a correct transmission. So,

$$\Pr(S_4 = 3) = \pi_0^2(1-p).$$

The same reasoning now leads to $\Pr(S_4 = 4) = \pi_0^2 pq$, $\Pr(S_4 = 5) = \pi_0^2 p(1-q)q$, and, in general,

$$\Pr(S_4 = n) = \pi_0^2 p(1-q)^{n-4}q, \quad n \geq 4.$$

The computation of the expectation of S_4 is straightforward and after some algebra, we obtain

$$\begin{aligned}\mathrm{E}(S_4) &= \pi_1 + 2\pi_1\pi_0 + 3\pi_0^2(1-p) + \sum_{n\geq 4} k\pi_0^2 p(1-q)^{n-4}q \\ &= \cdots = 1 + \frac{p}{q}.\end{aligned}$$

As said before, this is independent on K as far as $K \geq 2$.

With a similar process, we get

$$\mathit{MLBS}_K^{\mathit{split}} = 1 + \frac{p}{q} = \frac{1}{1 - \mathit{LR}},$$

for all $K \geq 2$. If we compare it to MLBS, which corresponds to a single stream transporting the whole sequence, we have

$$1 + \frac{p}{q} \leq \frac{1}{q} \Longleftrightarrow p + q \leq 1.$$

This means that the dispersion of losses with this scheme, assuming independent and equivalent paths, can be higher or lower than for a single one. This is of course specific to our simplistic assumptions. A consequence of the preceding remark is that if $\mathit{LR}_K^{\mathit{split}} \leq p_0$, we then have

$$\mathit{MLBS}_K^{\mathit{split}} \leq m_0 = \frac{1}{1 - p_0}.$$

For instance, if $\mathit{LR}_K^{\mathit{split}} \leq 0.2$ then $\mathit{MLBS}_K^{\mathit{split}} \leq 1.25$.

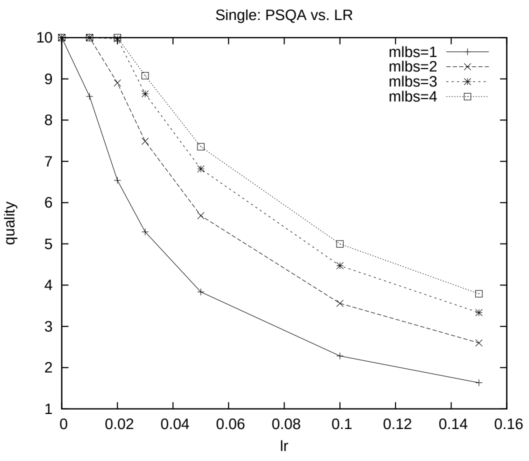
6.2.2 Evaluating and first results

Now we can study how the frame loss rate and frame mean loss burst size parameters affect the quality (as measured by the PSQA technique) for the *single* server and multiple server (*copy* and *split*) streaming policies.

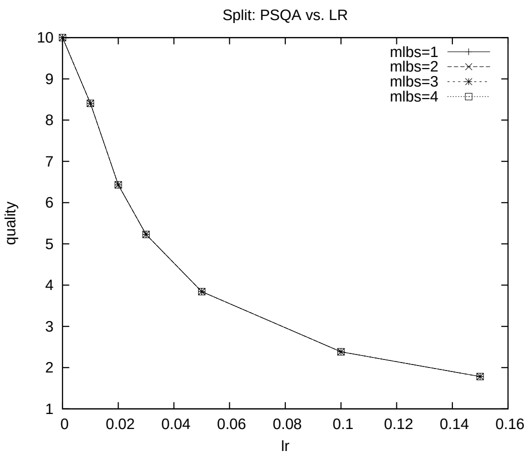
We use the PSQA *simple function* explained in Subsection 5.2.3. With this function, we can now compare the effect of the different streaming policies on the quality perceived at the client node. In Figures 6.5(a), 6.5(b), and 6.5(c) we see respectively the situation for the *single* server policy, the *split* policy with $K = 2$, and the *copy* policy with $K = 2$. In the three cases we have the same frame loss process going on at the server side. The figures show the perceived quality as a function of parameter LR, for different values of $MLBS$. In the figures, the LR and $MLBS$ values represent the frame loss process at the servers side. In the three cases, the perceived quality deteriorates quickly with increasing values of LR; but the quality values and the shape of the curves are very different for every policy. In the *single* server case, the behavior is almost insensitive to the $MLBS$ parameter; the quality depends very heavily on LR, deteriorating very quickly as this parameter grows a little bit from 0. In the *split* policy with two servers, the effect of LR is similar; we can observe that this policy completely cancels out the effect of the $MLBS$ parameter at the sources. In the *copy* policy, the observed quality levels for large LR are much higher than the corresponding ones in the other two models. Here the effect of $MLBS$ is more pronounced than in the previous two cases, larger $MLBS$ values having rather better quality.

In order to make it easier to compare the different policies, we show in Figures 6.6(a) and 6.6(b) the quality values for the three policies as a function of LR, with $MLBS = 1.0$ and $MLBS = 4.0$ respectively. These figures show how the *copy* policy with two servers, which transmits every frame twice, benefits from this redundancy and copes much better with high values of LR, maintaining better quality results, for both low and high $MLBS$ scenarios. In the case where $MLBS = 1$, the s*ingle* and *split* policy give the same results (this follows directly from our analytical models). When $MLBS = 4$, the *split* policy fares worse than the *single* policy. This comes from the fact that, in this case, the *split* policy results in a lower value for the $MLBS$ perceived by the user with respect to the $MLBS$ at the server side, and that the MOS computed from the experts opinions show that lower $MLBS$ result in lower quality values (i.e. the users prefer concentrated losses, and the *split* policy "breaks up" some of the frame loss correlation in the transmission process).

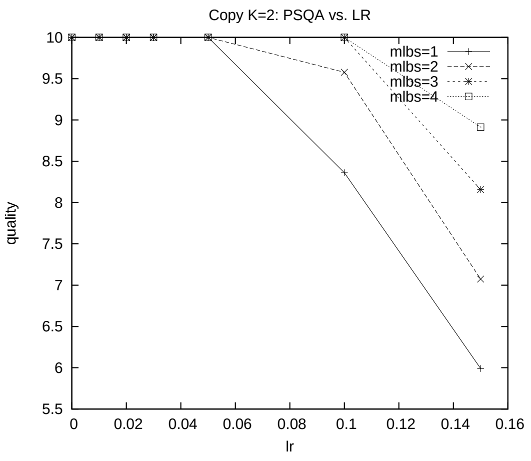
We can conclude that (obviously) sending the signal twice leads to a much better quality than only once, and that sending disjoint pieces of the stream is not always useful. A non-trivial conclusion about these results is that, from quality point of view, people prefer to concentrate losses instead of spreading them (in this range of losses). These results suggest that some intermediate multi-source streaming policy, between the extreme *copy* and *split* cases, can be useful to improve the perceived quality. Observe that the use of PSQA allows to say not only that the *copy* method is better than the *split* one, but for *how much*, and with respect to the user's perception point of view.



(a) Single method.

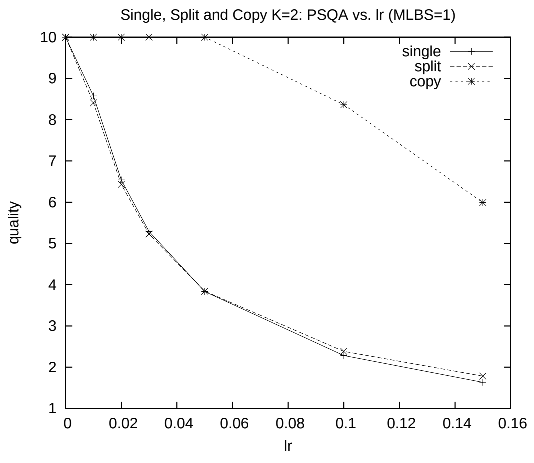


(b) Split method.

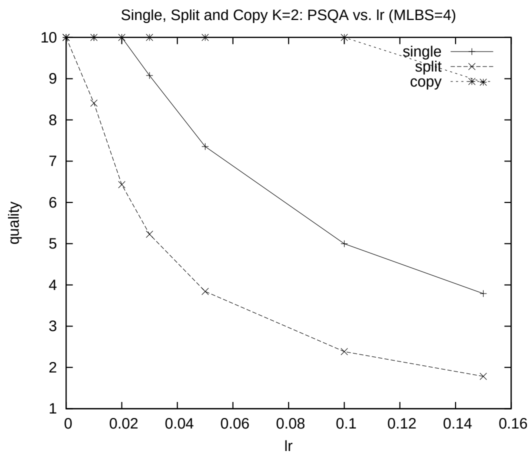


(c) Copy method.

Figure 6.5: Multi-source streaming methods, in the loss domain. The LR and $MLBS$ values represent the frame loss process at the servers side.



(a) Without bursts, $MLBS = 1$.



(b) With bursts, $MLBS = 4$.

Figure 6.6: Compare multi-source streaming methods. The LR and $MLBS$ values represent the frame loss process at the servers side.

6.3 Summary and Conclusions

This chapter proposes some general principles for the design of a live-video P2P distribution system following a multi-source procedure where the video stream is decomposed into different flows that travel independently through the network. We use the PSQA technique that allows to measure automatically the perceived quality as seen by the final users. The chapter focuses on the impact on quality (as measured by PSQA) on three extreme cases: sending the stream from one node to the other (for reference purposes), sending two complete copies of the stream through different paths, and sending two disjoint sub-streams whose union recomposes the original one.

We are able to evaluate the resulting perceived quality associated with these extreme architectures. The main conclusions are: (i) thanks to an improved version of PSQA we see that quality increases as losses concentrate in the stream (for a fixed loss rate); (ii) sending the signal twice obviously leads to a much better quality, even if, as expected, losses are less concentrated in this case; (iii) sending disjoint pieces of the stream is not useful, of course under our simplifying assumptions and scenarios.

This study suggests that some intermediate point between the extreme cases we considered, with a higher redundancy or with a higher number of flows per stream, can be an interesting solution. The chapter shows the utility of the delivery technique, It also strongly suggest to look for more realistic models. In next chapter we present a finer analysis of different splitting cases.

Chapter 7

Optimal Quality in Multi-Source Streaming

This chapter presents a deeper study of our proposed multi-source streaming technique. We show, using real data, how our technique allows to compensate efficiently the possible losses of frames due to peers leaving a P2P system. Moreover, we can statistically ensure some level of quality, when the dynamics in the network is specified (Section 7.1). This section has been partially presented at the IFIP/ACM Latin America Networking Conference (LANC'07)[283] and 4th Euro-FGI workshop on "New trends in modelling, quantitative methods and measurements" (Euro-FGI WP IA.7.1)[281] (listed at the Introduction as [lanc07] and [euro-fgi07]).

In Section 7.2, we improve the technique in different ways, specially considering the peers' heterogeneity, we obtain an optimal multi-source streaming technique that will be used in our GOL!P2P prototype. This result has been developed in coolaboration with A.P. Couto da Silva and presented at the IEEE International Conference on Communications (ICC 2008)[75] (listed at the Introduction as [icc08]).

Section 7.3 studies the multi-source approach from the receiver point of view, and defines a frame buffer policy in order to ensure a given quality level. This work has been partially presented at the 8th International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS'07)[74] (listed at the Introduction as [pmccs07]), again with the collaboration of A.P. Couto da Silva.

7.1 Video Quality Assurance: introducing the peers behavior

In Chapter 6 we presented multi-source streaming, and we studied its impacts in the quality for simple extreme streaming policies (Section 6.2.1). This analysis was stationary, in the sense that it showed the average quality behavior of the policies in a long term period.

The main difficulty in the design of a P2P system for distribution real-time video, is how to provide good quality levels in a context where this quality depends on other clients that are delivering the stream, and given the fact that users connect and disconnect very frequently. This section focuses then on the analysis of the impact that the peers' dynamics (represented by a transient model) has on the perceived quality. This is presented by considering the extreme multi-source streaming cases: *copy* (where the flows are just copies of the original sequence, with a very high redundancy level), *split* (where the sequence is split into simple disjoint sub-streams, without redundancy at all), and *redundant split* (where the sequence is split into redundant sub-streams, when is possible to reconstruct completely the stream with only one server failure). See Section 6.1 for details on these policies. We do some experiments in order to explore the consequences of these architecture choices on the quality level, extending the preliminary results presented in Section 6.2.1.

To evaluate the impact of this approach on quality, we use the PSQA *simple function* of Section 5.2, which maps the loss rates of video frames, LR, and the mean lost burst size, $MLBS$, into the perceived quality. This simple function is accurate enough to model the impact of the peers' behavior, illustrating with real data how the methods allow to compensate efficiently the possible losses of frames due to peers leaving the system.

In Subsection 7.1.1 we present a P2P network for live multi-source streaming, and a transient model of the nodes disconnection from the client's perspective. Subsection 7.1.2 develops models to estimate the streaming performance, in a environment with server nodes fails, needed to the construction of the PSQA measuring module. These models calibrate the multi-source streaming on the basis of the perceptual quality. In Subsection 7.1.3 some experimental results are introduced.

7.1.1 P2P Network and Dynamics Model

Multi-source streaming techniques allow for a large flexibility of the system, modulated by the dynamics of the clients. In particular, it is possible to increase the number of sources and/or the amount of redundant information sent through the network (the three cases discussed have different redundant information degree and can be potentially used with any amount of servers).

To decide which peer will serve which other nodes, some degree of intelligence and knowledge about the peers and the network state is needed. An important research effort is being done in the community on this hot topic (which will be presented in Chapter 8). The different proposals are based on decentralized or centralized algorithms, with structured or unstructured delivery, etc., depending on the specific application considered. Again, from the client point of view, these possible assignments can be modeled after some delay (or time of convergence) T. That is, considering a client receiving the stream from K independent servers, when one of these servers leaves the network, whatever the assignment algorithm used, it will need some time to operate, time denoted in the sequel by T.

In this section we describe a simple Markovian model used to represent the server connection/disconnection process in a multi-source streaming context. We adopt the following simplifying assumptions. The connection-time of any node acting as a server (that is, most of the nodes in the network) is exponentially distributed with some parameter λ. That is, $1/\lambda$ is the expected time a customer remains connected. It can be estimated from network statistics (strictly speaking, we refer here to the servers' connection time, which means that, to estimate λ, we must sample on the population of clients acting as servers; this usually happens after a minimal amount of connection time). Since we further assume that the servers leave the network independently of each other, the number of connected servers sending the stream to a fixed but arbitrary customer,, at time t, considering that the network was re-built at time 0 and that no other re-building process is done in $[0, t]$, is a Markov process with states $K, K-1, \ldots, 1, 0$. The corresponding transition graph is shown in Figure 7.1.

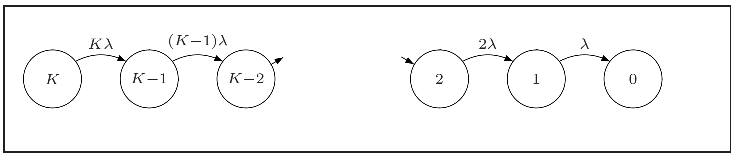


Figure 7.1: The Markovian model used to represent the evolution of the number of connected servers sending the stream to the same (arbitrary) client.

Since the failures of the components are assumed to occur independently, the probability that any of them is operating at time t is $e^{-\lambda t}$, and thus, the number of active servers at time t is Binomial with parameters K and $e^{-\lambda t}$. In other words, if $p_{K,i}(t)$ is the probability that i servers among the initial K are still operating at time t, then we have

$$p_{K,i}(t) = \binom{K}{i} e^{-i\lambda t}(1 - e^{-\lambda t})^{K-i}, \quad K \geq i \geq 0,\ \lambda, t \in \Re.$$

In this work we use $1/\lambda = 900$ sec. and $T = 10$ sec. To compute the value of λ, we employed logs of user behavior (specifically connection times) from our live-video reference service (see Section 2.5 for details). We filtered the information of very short lived nodes: since in the proposed architecture we suppose that the servers are P2P nodes, it is reasonable to assume that the mean-life of the users will correspond to the expected stay of the servers in our model. Some values of $p_{K,i}(t)$ are given in Table 7.1 based on measured data. Observe that when K increases, the probability of observing a server failure increases as well. So, *the interest in using several servers must be balanced against the probability of having failures before the next re-configuration point.*

7.1.2 Multi-source Streaming Models

In this subsection we develop models for our three multi-source streaming policies (with K servers each one of them). The goal is to evaluate the values of the two parameters we need to build the PSQA quality assessment values, LR and $MLBS$.

7.1.2.1 Sending K copies of the stream

Assume K copies of the same stream travel following independent and stochastically equivalent paths to the same terminal. The loss process at any of the K streams is represented by the server failure model described in the previous subsection. It is clear that the receiver will observe the loss of a frame only if all the K copies of the frames are lost. If $LR^{copy}_{K,i}$ denotes this global Loss Rate with K initial servers, and i still connected, we have:

$$LR^{copy}_{K,i} = \begin{cases} 1 & \text{if } i = 0 \\ 0 & \text{otherwise} \end{cases}, \quad K \geq i \geq 0,\ K \geq 1.$$

Table 7.1: The probability of having i servers still connected at time T, for K initially connected servers at time 0, representing the last re-configuration of the network, that is, the number $p_{K,i}(T)$, for some values of K and all $i \leq K$; other data: $1/\lambda = 900$ sec, $T = 10$ sec.

i/K	1	2	3	4	5
0	0.0110	0.0001	0.0000	0.0000	0.0000
1	0.9890	0.0219	0.0004	0.0000	0.0000
2		0.9780	0.0324	0.0007	0.0000
3			0.9672	0.0427	0.0012
4				0.9565	0.0528
5					0.9460

i/K	6	7	8	9	10
0	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.0018	0.0000	0.0000	0.0000	0.0000
5	0.0627	0.0024	0.0001	0.0000	0.0000
6	0.9355	0.0724	0.0032	0.0001	0.0000
7		0.9252	0.0818	0.0041	0.0001
8			0.9149	0.0910	0.0050
9				0.9048	0.1000
10					0.8948

The global Mean Loss Burst Size is, in this simple case,

$$MLBS^{copy}_{K,i} = \begin{cases} \infty & \text{if } i = 0 \\ \nexists & \text{otherwise} \end{cases}, \quad K \geq i \geq 0,\ K \geq 1.$$

7.1.2.2 Simple split of the stream into $K \geq 2$ substreams

In the other extreme case considered in this section, we have K sub-streams transporting each a frame over K in the following way: frame 1 goes through sub-stream 1, frame 2 through sub-stream 2, until frame K going through sub-stream K; then frame $K+1$ is sent through sub-stream 1, frame $K+2$ through sub-stream 2, etc. In general, frame n is sent by sub-stream $((n-1) \mod K) + 1$.

Assuming independence in server failures again, the global Loss Rate of this scheme is obviously proportional to the number of faulty servers; when i of them are still connected, we have

$$LR^{split}_{K,i} = \frac{K-i}{K}, \quad K \geq i \geq 0,\ K \geq 1.$$

To get a feeling of the numerical values, we plot in Table 7.2 this global Loss Rate for some values of K and all $i \leq K$.

The evaluation of the Mean Loss Burst Size is much more involved than the previous one, but since our goal is to guarantee some quality level, we only use a trivial lower bound and an upper bound, by observing that, by definition,

$$1 \leq MLBS^{split}_{K,i} \leq K - i, \quad K \geq i \geq 0,\ K \geq 1.$$

Table 7.2: Simple split method. Global loss rate, $LR^{split}_{K,i}$ (that is, the number $1 - i/K$), for some values of the initial numbers K of servers and all possible values of the number i of surviving servers.

i/K	1	2	3	4	5
0	1.0000	1.0000	1.0000	1.0000	1.0000
1	0.0000	0.5000	0.6667	0.7500	0.8000
2		0.0000	0.3333	0.5000	0.6000
3			0.0000	0.2500	0.4000
4				0.0000	0.2000
5					0.0000

i/K	6	7	8	9	10
0	1.0000	1.0000	1.0000	1.0000	1.0000
1	0.8333	0.8571	0.8750	0.8889	0.9000
2	0.6667	0.7143	0.7500	0.7778	0.8000
3	0.5000	0.5714	0.6250	0.6667	0.7000
4	0.3333	0.4286	0.5000	0.5556	0.6000
5	0.1667	0.2857	0.3750	0.4444	0.5000
6	0.0000	0.1429	0.2500	0.3333	0.4000
7		0.0000	0.1250	0.2222	0.3000
8			0.0000	0.1111	0.2000
9				0.0000	0.1000
10					0.0000

7.1.2.3 Split of the stream into $K \geq 2$ substreams, adding complete redundancy

Between these two extreme policies (copy and split cases), we can for example split the stream in K sub-streams adding some redundancy to each one in order to diminish the effect of losses at least when only one server disconnects (fails). If the original stream needs bandwidth B Kbps, then we assume that each sub-stream will use B/K Kbps plus some bandwidth needed to transport redundant data. Substream j is completely sent by server j, and its content is also sent by the remaining $K-1$ servers, each of them sending exactly a $1/(K-1)$th part of it.

Let us look now at the losses when there are only i active servers, among the K initially connected. In this case, without any redundancy we will loose a fraction $(K-i)/K$ of the stream. But with the adopted redundancy scheme, this is diminished by the fraction of this information that is transported, as redundant data, by the remaining connected servers. We have:

$$LR^{red}_{K,i} = \frac{K-i}{K} - \frac{(K-i)}{K}\frac{i}{K-1} = \frac{(K-i)(K-1-i)}{K(K-1)}.$$

Some values of $LR^{red}_{K,i}$ are given in Table 7.3.

For the evaluation of the Mean Loss Burst Size we can use the same trivial lower and upper bounds than in the *split* case:

$$1 \leq MLBS^{red}_{K,i} \leq K-1, \quad K \geq i \geq 0,\ K \geq 1.$$

7.1.3 Perceived Quality Evaluations and Results

We have obtained the equivalent LR and $MLBS$ parameters from the client point of view, for the three multiple server streaming policies (*copy*, *split* and *redundant split*). There only

Table 7.3: Split method with redundancy. Global loss rate, $LR^{red}_{K,i}$, for some values of the initial numbers K of servers and all possible values of the number i of surviving servers.

i/K	1	2	3	4	5
0	1.0000	1.0000	1.0000	1.0000	1.0000
1	0.0000	0.0000	0.3333	0.5000	0.6000
2		0.0000	0.0000	0.1667	0.3000
3			0.0000	0.0000	0.1000
4				0.0000	0.0000
5					0.0000

i/K	6	7	8	9	10
0	1.0000	1.0000	1.0000	1.0000	1.0000
1	0.6667	0.7143	0.7500	0.7778	0.8000
2	0.4000	0.4762	0.5357	0.5833	0.6222
3	0.2000	0.2857	0.3571	0.4167	0.4667
4	0.0667	0.1429	0.2143	0.2778	0.3333
5	0.0000	0.0476	0.1071	0.1667	0.2222
6	0.0000	0.0000	0.0357	0.0833	0.1333
7		0.0000	0.0000	0.0278	0.0667
8			0.0000	0.0000	0.0222
9				0.0000	0.0000
10					0.0000

remains to evaluate the PSQA *simple function* (of Section 5.2) mapping these two parameters into perceived quality.

Sending K copies of the stream. Using the loss model for the *copy* method of Subsection 7.1.2.1, we evaluate the PSQA measure in the different $LR^{copy}_{K,i}$ possibilities, the result is summarized in Table 7.4.

Simple split of the stream into $K \geq 2$ substreams. Using the loss model for the *split* method of Subsection 7.1.2.2, we evaluate the PSQA measure in the different $LR^{split}_{K,i}$ possibilities, the result is summarized in Table 7.5 for the lower quality bound (based in lower bound $MLBS^{split}_{K,i} = 1$) and in Table 7.6 for the upper quality bound (based in upper bound $MLBS^{split}_{K,i} = K - i$).

Split of the stream into $K \geq 2$ substreams, adding complete redundancy, $r = 1$. Using the loss model for the *redundant split* method of Subsection 7.1.2.3, we evaluate the PSQA measure in the different $LR^{red}_{K,i}$ possibilities, the result is summarized in Table 7.7 for the lower quality bound (based in lower bound $MLBS^{red}_{K,i} = 1$) and in Table 7.8 for the upper quality bound (based in upper bound $MLBS^{red}_{K,i} = K - i$).

Table 7.4: Copy method. Perceived Quality as a function of K (initial number of servers) and i (number of surviving servers).

i/K	1	2	3	4	5
0	0.000	0.000	0.000	0.000	0.000
1	10.000	10.000	10.000	10.000	10.000
2		10.000	10.000	10.000	10.000
3			10.000	10.000	10.000
4				10.000	10.000
5					10.000

i/K	6	7	8	9	10
0	0.000	0.000	0.000	0.000	0.000
1	10.000	10.000	10.000	10.000	10.000
2	10.000	10.000	10.000	10.000	10.000
3	10.000	10.000	10.000	10.000	10.000
4	10.000	10.000	10.000	10.000	10.000
5	10.000	10.000	10.000	10.000	10.000
6	10.000	10.000	10.000	10.000	10.000
7		10.000	10.000	10.000	10.000
8			10.000	10.000	10.000
9				10.000	10.000
10					10.000

7.1.3.1 Assuring Video Quality

The PSQA technique makes it possible to know the subjective quality associated with every state of the network (i.e, with any combination of working and failed servers). This information allows us to answer different interesting and relevant questions.

Under the given assumptions, the worst situation is just before a network rebuilds the broken connections, that is, at time T^- (if we consider that the last re-configuration happened at time 0). The expected perceived quality (considering the whole client population) at that time is then

$$\mathrm{E}(Q_K) = \sum_{i=1}^{K} Q(LR_{K,i}, MLBS_{K,i}) p_{K,i}(T),$$

where $Q()$ is the PSQA *simple function* of Section 5.2.

Table 7.9 and Figure 7.2 show the values of $\mathrm{E}(Q_K)$ for the three policies, for K varying from 1 to 10 (the data was computed using the lower bound for the perceived quality, the difference with the upper bound is completely negligible). Observe that, as it should be expected, the expected value is close to the maximal one (10), for any number of servers. We can observe also that there is a slight trend towards smaller values as K increases. This is due to the fact that the probability of having at least a failure (a missing flow) before T increases as K increases (this probability is $1 - e^{-K\lambda T}$). This compensates the effect of increasing the number of flows, and with the (realistic) numerical values used here, the combined effect is this decreasing trend. In the case where there is no redundancy (simple *split*), the subjective quality degenerates rapidly with the growth of servers K. In the *redundant split* policy, passing from one server to two servers improves greatly the quality levels; adding additional servers can lead to slight decreases in perceived quality, but the behavior is very robust in this aspect.

Table 7.5: Simple split method. Minimal Perceived Quality ($MLBS_{K,i}^{split} = 1$) as a function of K (initial number of servers) and i (number of surviving servers).

i/K	1	2	3	4	5	6	7	8	9	10
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	10.000	0.051	0.045	0.045	0.045	0.045	0.045	0.045	0.045	0.045
2		10.000	0.075	0.051	0.045	0.045	0.045	0.045	0.045	0.045
3			10.000	0.099	0.063	0.051	0.045	0.045	0.045	0.045
4				10.000	0.123	0.075	0.059	0.051	0.046	0.045
5					10.000	0.146	0.087	0.067	0.057	0.051
6						10.000	0.168	0.099	0.075	0.063
7							10.000	0.190	0.111	0.083
8								10.000	0.212	0.123
9									10.000	0.233
10										10.000

Table 7.6: Simple split method. Maximal Perceived Quality ($MLBS_{K,i}^{split} = K-i$) as a function of K (initial number of servers) and i (surviving servers).

i/K	1	2	3	4	5	6	7	8	9	10
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	10.000	0.051	0.075	0.097	0.113	0.126	0.136	0.145	0.152	0.158
2		10.000	0.075	0.085	0.097	0.113	0.126	0.136	0.145	0.152
3			10.000	0.099	0.105	0.109	0.113	0.126	0.136	0.145
4				10.000	0.123	0.126	0.127	0.127	0.128	0.136
5					10.000	0.146	0.146	0.144	0.143	0.142
6						10.000	0.168	0.165	0.162	0.158
7							10.000	0.190	0.185	0.179
8								10.000	0.212	0.205
9									10.000	0.233
10										10.000

Table 7.7: Redundant split. Minimal Perceived Quality ($MLBS_{K,i}^{red} = 1$) as a function of K (initial number of servers) and i (surviving servers).

i/K	1	2	3	4	5	6	7	8	9	10
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	10.000	10.000	0.075	0.051	0.045	0.045	0.045	0.045	0.045	0.045
2		10.000	10.000	0.146	0.083	0.063	0.053	0.047	0.045	0.045
3			10.000	10.000	0.233	0.123	0.087	0.070	0.060	0.054
4				10.000	10.000	0.333	0.168	0.115	0.090	0.075
5					10.000	10.000	0.442	0.219	0.146	0.111
6						10.000	10.000	0.555	0.274	0.179
7							10.000	10.000	0.669	0.333
8								10.000	10.000	0.781
9									10.000	10.000
10										10.000

Table 7.8: Redundant split method. Maximal Perceived Quality ($MLBS_{K,i}^{red} = K - i$) as a function of K (initial number of servers) and i (surviving servers).

i/K	1	2	3	4	5	6	7	8	9	10
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	10.000	10.000	0.126	0.109	0.113	0.126	0.136	0.145	0.152	0.158
2		10.000	10.000	0.243	0.179	0.158	0.149	0.143	0.145	0.152
3			10.000	10.000	0.388	0.263	0.219	0.197	0.183	0.174
4				10.000	10.000	0.554	0.360	0.288	0.250	0.227
5					10.000	10.000	0.732	0.468	0.365	0.309
6						10.000	10.000	0.916	0.585	0.448
7							10.000	10.000	1.101	0.709
8								10.000	10.000	1.283
9									10.000	10.000
10										10.000

Table 7.9: Average Quality (worst case) of multi-source streaming methods, as a function of the number of servers K.

K/Method (bandwidth)	*copy* (KB)	*split* (B)	*redundant split* $(2B)$
1	9.890	9.890	9.890
2	9.999	9.781	9.999
3	10.000	9.675	9.996
4	10.000	9.570	9.993
5	10.000	9.466	9.989
6	10.000	9.364	9.983
7	10.000	9.264	9.977
8	10.000	9.166	9.970
9	10.000	9.068	9.963
10	10.000	8.973	9.955

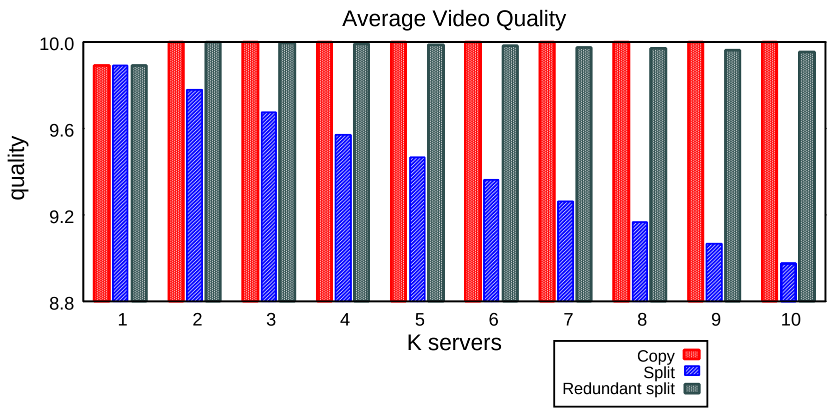


Figure 7.2: Graphical representation of the average quality (worst case) of multi-source streaming methods, as a function of the number of servers K.

The *copy* policy has always the best perceived quality levels, but at the expense of an important transmission overhead. It is possible to compare the *copy* and *redundant split* cases, where (for the frequency of disconnection of our real scenario) there does not seem to be much gain in sending K copies of the streaming (*copy*), as sending a single copy (*redundant split*) only loses a little percentage of the quality.

Another interesting question that it is possible to answer with our approach is: how many servers are needed in order to ensure that with a given probability, the quality of the transmission will be greater or equal than a pre-defined quality level? We have

$$\Pr(Q_K > Q_{\text{min}}) = \sum_{i=1/Q_{K,i}>Q_{\text{min}}}^{K} p_{K,i}(T),$$

where $Q_{K,i} = Q(LR_{K,i}, MLBS_{K,i})$.

Table 7.10: $\Pr(Q_K > 9.99)$ in our multi-source streaming methods, as a function of the numbers K of servers.

K/Method (bandwidth)	*copy* (KB)	*split* (B)	*redundant split* $(2B)$
1	0.988950	0.988950	0.988950
2	0.999878	0.978023	0.999878
3	0.999999	0.967216	0.999636
4	∼1.000000	0.956529	0.999278
5	∼1.000000	0.945959	0.998806
6	∼1.000000	0.935507	0.998222
7	∼1.000000	0.925170	0.997529
8	∼1.000000	0.914947	0.996729
9	∼1.000000	0.904837	0.995826
10	∼1.000000	0.894839	0.994820

Assume we want to ensure a perfect quality ($Q_{\text{min}} = 9.99$). Table 7.10 gives $\Pr(Q_K > 9.99)$ for different values of K, in the three streaming policies. For instance, if we want $\Pr(Q_K > 9.99) \geq 0.999$ (an alternative equivalent interpretation is that we want to ensure that at least 99.9% of the users will perceive perfect quality), this table says that, this can be achieved if at least 2 servers are used in the *copy* method; or if between 2 and 4 servers are used in the *redundant split* method. The condition is impossible to satisfy if there is no redundancy at all (that is, with the *simple split* method).

This study suggests that among the different policies for multi-source streaming techniques, the ones employing a limited amount of redundancy may well be the methods of choice, as they allow to improve the QoE at the expense of a limited transmission overhead. Simple analytical models can be useful to understand the qualitative and quantitative behavior of the different policies. In order to support the designers' decision making, it can be useful to develop more realistic models; in particular, next section introduces the peers dynamics heterogeneity, among other improvements.

7.2 Improving the Multi-Source Streaming and Looking for Optimality: Considering the peers' heterogeneity

It has been shown that peers are heterogeneous in their capability [299] and a network design has to consider this behavior to be scalable. The election of the peer servers that are part of the multi-source streaming is important, with respect to their bandwidth capacity, and especially to their lifetime.

In our approach, the P2P system has the necessary knowledge about the peers and the network state, and can thus decide which peer will serve which other node (see Chapter 8 for the structural P2P design problem), and how flows must be split and merged. The network topology is then re-built periodically, every T seconds, with $T = 10$ seconds as well as the previous section. A generic peer starts the cycle by receiving the video stream from K heterogeneous and independent server peers, but before T some server peers can become disconnected.

In this section we will describe some improvements on the multi-source streaming scheme, on the models, and how we use them for deriving optimality results. These improvements go in the direction of sending the most important data from peers highly "engaged" with the network. All the improvements described here were implemented (Section 10.1) and tested (Section 10.3) on our GOL!P2P prototype.

7.2.1 First improvement of the scheme: unequal splitting

Instead of splitting into sub-streams with equal bandwidth, we send a fraction y_k of the stream by flow (server) k. We call y_k the *weight* of server k. As we want that the best servers send most of the data, we set $y_{k+1} = \gamma y_k$, with $\gamma > 1$. This means that $y_k = \gamma^{k-1} y_1$, with $y_1 = (\gamma - 1)/(\gamma^K - 1)$.

We also add some redundancy $r \in [0, 1]$ to the global flow; $r = 0$ means no redundancy, $r = 1$ means that any frame is sent twice. If the original stream has a bitrate of B Kbps, the total bandwidth employed is thus $BW^r = (1 + r) B$ Kbps. The redundancy is distributed in the following way. A fraction r of the original data sent by server k is also sent by the other servers, proportionally to their weights. The total bandwidth BW_k^r used by server k is then

$$BW_k^r = y_k B + \sum_{j=1, j\neq k}^{K} r y_k B \frac{y_j}{1 - y_k} = (1 + r)\, y_k B,$$

where $r y_k B y_j / (1 - y_k)$ is the bandwidth used by server k to stream the redundancy of sub-stream j. The technique implies that each frame is sent either once or twice, but no frame is sent more than twice.

This improvement generalizes the split methods presented before, where *split* uses $\gamma = 1^+$ and $r = 0$, and *redundant split* uses $\gamma = 1^+$ and $r = 1$.

7.2.2 Improving the model of the peers' dynamics

Concerning the model, we consider now that peers belong to different classes, according to their mean connection time in the network (peer's lifetime). The average behavior of the peers in each class is obtained from our video delivery reference service presented in Section 2.5. We proceed as follows. After choosing K, we sort the whole set of peers in the available logs according to their mean connection times. If M is the size of the population, the first M/K peers are assigned to class 1, the second M/K to class 2, etc. With class k we associate the failure rate $\lambda_k = 1/m_k$ where m_k is the mean connection time of the peers in class k, and we then assume that in that class, peers remain connected for an exponentially distributed period of time. Observe that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_K$ (best peers are at the end). The reason we use as many classes as sub-streams is that we consider here that peers are all equivalent with respect to the service, so, that they must all receive the same service. This leads to the fact that each client is served (at the beginning of a cycle) by one server of each class.

Let us denote by $N_k(t)$ the binary r.v. equal to 1 iff server k is connected at t, and by $\vec{N}(t)$ the vector $\vec{N}(t) = (N_1(t), \cdots, N_K(t))$. Then, for any *configuration* $\vec{n} \in \{0, 1\}^K$, we have

$$\Pr(\vec{N}(t) = \vec{n}) = \prod_{j:n_j=1} e^{-\lambda_j t} \prod_{j:n_j=0} (1 - e^{-\lambda_j t}). \tag{7.1}$$

To explore in more detail the degraded case where the client is not receiving all the sub-streams and the next reconstruction point didn't arrived yet, consider a configuration $\vec{n} \neq \vec{1}$, where $\vec{1} = (1, \cdots, 1)$. This means that at least one server left the network. Let j be such that $n_j = 0$, the fraction of lost data due to the lack of the jth sub-stream is:

$$LR_j^r = y_j - \frac{r y_j}{1 - y_j} \sum_{i:n_i=1} y_i.$$

The total loss rate at configuration $\vec{n}$ is then

$$LR_{\vec{n}}^r = \sum_{j:n_j=0} LR_j^{red} = 1 - \Big(\sum_{i:n_i=1} y_i \Big) \Big(1 + r \sum_{j:n_j=0} \frac{y_j}{1 - y_j} \Big). \tag{7.2}$$

7.2.3 Second improvement to the transmission method: discrimination per frame type

In the most adopted standard specifications for video (in this work we consider a MPEG-4 codec), the transmission units are the *frames*, which belong to three main types: I, P and B. The frames differ in how the loss of each of them influences the QoE (see Section2.1).

It is immediate to see that our transmission scheme can be extended in order to control the transmission of frames I, P and B separately. That is, we can build the same scheme with parameters γ_I, γ_P, γ_B, and for different redundancy factors per frame type, r_I, r_P, r_B. See graphically the general method in Figure 7.3. The loss rate of each frame type is given by

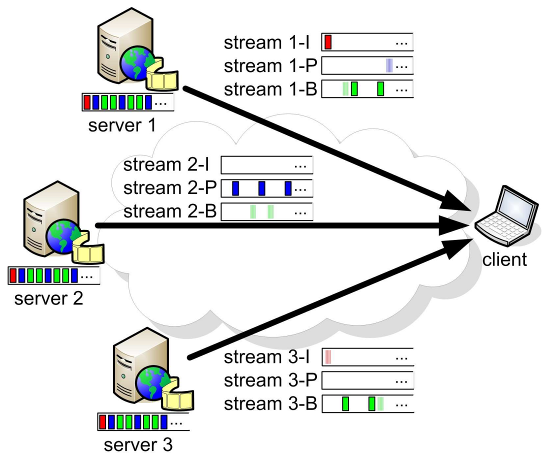


Figure 7.3: General split streaming method. Where three servers ($K = 3$) send one flow per frame type, with different weight and redundancy.

Equation (7.2), where we use the weights and redundancy for that frame type. We denote by $LR_{x,\vec{n}}$ the loss rate of x frames, where x is I, P or B, when the configuration is $\vec{n}$.

Using now the PSQA *complex function*, of Section 5.3.3, giving the perceived quality as a function of the per-frame type loss rates[1] , the QoE in that configuration is $Q(LR_{I,\vec{n}}, LR_{P,\vec{n}}, LR_{B,\vec{n}})$. We fix the motion source parameters of the original function to its average values (the average GOP size is 174 frames, and the P-Frames Weight per GOP is 0.7007), because we do not use them in this analysis.

Our main objective here is to maximize the mean QoE value, considered at the end of the cycle (worst case analysis), and *when at least one server is down*, because this is the main critical situation. This expectation is

$$\mathrm{E}(Q_K)(T) = \sum_{\vec{n} \neq \vec{1}} Pr(\vec{N}(T) = \vec{n}) Q(LR_{I,\vec{n}}, LR_{P,\vec{n}}, LR_{B,\vec{n}}),$$

where the peer's probability connection for each configuration is defined by Equation (7.1).

7.2.4 Optimization Results

In what follows, we analyze the behavior of the system when at least one peer leaves the network, in order to address the robustness of our proposal. The presented results focus on the evaluation of all concerned parameters for maximizing $\mathrm{E}(Q_K)(T)$. The needed bandwidth for delivering the original video is $B = 512$ Kbps. We will consider three redundancy levels: 0%, 25% and 50%. In the two last cases, the total bandwidth is equal to 640 and 768 Kbps respectively.

The optimization results were obtained using the *fminsearch* function of Matlab[2], which finds the minimum of a scalar function of several variables, starting at an initial estimation. For all the results, we randomly chose 10 starting points, with $2 \leq K \leq 10$, and the maximum mean quality among all provided results was selected.

For the scenario we are considering, first observe that if we have only one *server peer* and it leaves the network, the QoE is obviously zero. Now, in real systems, peers have sometimes bandwidth constraints. We will consider two cases: either the peers upload information without bandwidth constraints (i.e. only the download bandwidth received by the client is constrained by $BW^r = (1 + r)\, B$ Kbps), or the upload bandwidth is restricted to less than BW^r/K Kbps. The two scenarios result in different restrictions on the γ and r parameter values. The *fminsearch* function does not consider constraints, we augmented the objective function with bandwidth constraints in each case in order to solve it in Matlab.

No upload bandwidth limitations. We start by the simple case, where the bandwidth is limited only by its total amount, without caring about the bandwidth of each *server peer*. We report our results on Tables 7.11, 7.12 and 7.13 ($r = 0\%$, $r = 25\%$ and $r = 50\%$).

Equal upload bandwidth limitation. Now, consider that each *server peer* has a bandwidth limitation. Tables 7.14 and 7.15 show the results ($r = 25\%$ and $r = 50\%$).

The following conclusions can be listed. (1) As expected, more bandwidth results in a better quality for the delivered video. (2) For P-frames, the largest quality is achieved when γ_P is

[1]This function shows that the quality degrades quickly with an increment in the loss rate of frames I and P, and that the impact of P-frames' losses is a bit higher than for I-frames. Also, it shows that the quality degrades slowly with an increment in the loss rate of B-frames.

[2]© 1984-2007 The MathWorks, Inc.

around 2. This means that *server peers* that stay longer in the system will be responsible for delivering the most important information. These results match with practical experiences, where P-frames have a crucial role on the quality perceived by the end-user. (3) After the P-frame in importance order with respect to the quality, we have I-frames and then B-frames, confirming the results of the *subjective testing* sessions. (4) The redundancy value is counterbalanced by the γ value: for greater γ we can use a smaller value of the redundancy factor, with the following interpretation: if the largest part of the information is delivered by the most stable peers, it is not necessary to use a high redundancy factor.

To explore the robustness of the optimization procedure, we selected some cases with more starting points in order to obtain the best $\mathrm{E}(Q_K)(T)$. Just for illustration, let us consider the case with $r = 25\%$ and $K = 2$ (Table 7.12). We randomly chose 70 starting points leading to 70 values of $\mathrm{E}(Q_K)(T)$ ranging in $[2.571, 3.716]$ with an arithmetic average value of 3.250. Coming back to Table 7.12, we can see that this does not basically change our pseudo-optimal points.

The analysis shows that the highest the number of server peers, the better the quality. On the other hand, the signaling overhead also increases with that number. The results show, however, that a few number of servers is enough for an excellent quality level (e.g., for $\mathrm{E}(Q_K)(T) \geq 9.0$ and 25% of redundancy, 7 servers are enough). In Section 10.1, we discuss about the main problems of implementing the *multi-source streaming technique* and the practical effects of increasing the number of server peers.

Table 7.11: Multi-source configurations without upload bandwidth limitations, γ and redundancy factors optimization (0% redundancy)

K	$\mathrm{E}(Q_K)(T)$	γ_I	γ_P	γ_B
1	0	-	-	-
2	2.471	1.999e+00	2.000e+00	1.999e+00
3	4.368	1.999e+00	2.000e+00	1.999e+00
4	6.041	1.997e+00	1.999e+00	1.999e+00
5	7.297	1.999e+00	1.999e+00	1.993e+00
6	8.159	2.000e+00	1.999e+00	1.999e+00
7	8.727	1.999e+00	2.000e+00	1.999e+00
8	9.099	1.999e+00	1.999e+00	1.999e+00
9	9.346	1.999e+00	1.999e+00	1.999e+00
10	9.477	1.999e+00	2.000e+00	1.999e+00

K	$\mathrm{E}(Q_K)(T)$	r_I	r_P	r_B
1	0	-	-	-
2	2.471	8.554e-02	1.942e-02	8.535e-02
3	4.368	3.852e-02	4.085e-02	3.455e-02
4	6.041	9.749e-02	1.904e-02	7.325e-02
5	7.297	1.607e-02	4.961e-02	8.200e-03
6	8.159	8.506e-02	2.005e-02	8.160e-03
7	8.727	8.642e-02	2.009e-02	7.965e-02
8	9.099	8.749e-02	2.000e-02	7.898e-02
9	9.346	9.477e-02	4.007e-02	1.867e-02
10	9.477	8.046e-02	3.937e-02	9.900e-04

Table 7.12: Multi-source configurations without upload bandwidth limitations, γ and redundancy factors optimization (25% redundancy)

K	E$(Q_K)(T)$	γ_I	γ_P	γ_B
1	0	-	-	-
2	3.493	1.189e+00	1.999e+00	1.146e+00
3	5.593	1.000e+00	1.999e+00	1.712e+00
4	7.016	1.535e+00	1.992e+00	1.999e+00
5	7.857	1.000e+00	1.999e+00	1.589e+00
6	8.606	1.524e+00	1.995e+00	1.744e+00
7	9.076	1.109e+00	1.999e+00	1.921e+00
8	9.332	1.000e+00	1.999e+00	1.993e+00
9	9.479	1.890e+00	1.999e+00	1.999e+00
10	9.591	1.999e+00	1.999e+00	1.898e+00
K	E$(Q_K)(T)$	r_I	r_P	r_B
1	0	-	-	-
2	3.493	9.797e-01	2.547e-01	2.171e-01
3	5.593	9.999e-01	2.942e-01	8.694e-02
4	7.016	9.999e-01	2.252e-01	2.941e-01
5	7.857	9.223e-01	3.228e-01	1.491e-06
6	8.606	9.378e-01	2.789e-01	6.931e-06
7	9.076	9.949e-01	2.583e-01	4.065e-04
8	9.332	9.991e-01	2.154e-01	8.857e-02
9	9.479	9.999e-01	1.366e-01	9.171e-02
10	9.591	9.999e-01	1.829e-01	6.647e-02

Table 7.13: Multi-source configurations without upload bandwidth limitations, γ and redundancy factors optimization (50% redundancy)

K	E$(Q_K)(T)$	γ_I	γ_P	γ_B
1	0	-	-	-
2	5.011	1.258e+00	1.874e+00	1.615e+00
3	6.636	1.147e+00	1.863e+00	1.999e+00
4	7.865	1.999e+00	1.999e+00	1.999e+00
5	8.711	1.420e+00	1.993e+00	1.993e+00
6	9.024	1.644e+00	1.999e+00	1.601e+00
7	9.324	1.809e+00	1.999e+00	1.999e+00
8	9.610	1.999e+00	1.999e+00	1.998e+00
9	9.669	1.822e+00	1.995e+00	1.999e+00
10	9.766	1.910e+00	1.999e+00	1.914e+00
K	E$(Q_K)(T)$	r_I	r_P	r_B
1	0	-	-	-
2	5.011	9.999e-01	6.629e-01	4.926e-06
3	6.636	9.999e-01	5.739e-01	3.093e-01
4	7.865	7.043e-01	6.323e-01	1.456e-01
5	8.711	9.999e-01	5.662e-01	3.887e-01
6	9.024	9.999e-01	4.340e-01	8.044e-01
7	9.324	5.996e-01	5.650e-01	5.259e-01
8	9.610	7.697e-01	6.520e-01	1.957e-01
9	9.669	4.824e-01	7.285e-01	4.523e-04
10	9.766	8.409e-01	7.082e-01	6.855e-06

Table 7.14: Multi-source configurations with upload bandwidth limitations, γ and redundancy factors Optimization (25% Redundancy)

K	E$(Q_K)(T)$	γ_I	γ_P	γ_B
1	0	-	-	-
2	3.557	1.000e+00	1.960e+00	1.607e+00
3	5.442	1.296e+00	1.999e+00	1.771e+00
4	7.020	1.108e+00	1.999e+00	1.999e+00
5	7.972	1.220e+00	1.999e+00	1.797e+00
6	8.719	1.999e+00	1.997e+00	1.999e+00
7	9.064	1.409e+00	1.999e+00	1.661e+00
8	9.319	1.999e+00	1.999e+00	1.692e+00
9	9.510	1.995e+00	1.932e+00	1.974e+00
10	9.586	1.999e+00	1.996e+00	1.949e+00
K	E$(Q_K)(T)$	r_I	r_P	r_B
1	0	-	-	-
2	3.557	9.913e-01	2.953e-01	5.410e-04
3	5.442	9.999e-01	1.705e-01	4.744e-01
4	7.020	9.999e-01	2.279e-01	2.610e-01
5	7.972	9.999e-01	1.988e-01	3.856e-01
6	8.719	8.286e-01	3.305e-01	9.977e-06
7	9.064	9.999e-01	2.462e-01	2.373e-01
8	9.319	8.124e-01	2.946e-01	6.276e-06
9	9.510	7.986e-01	3.428e-01	5.984e-05
10	9.586	1.550e-01	3.959e-01	4.255e-02

Table 7.15: Multi-source configurations with upload bandwidth limitations, γ and redundancy factors Optimization (50% Redundancy)

K	E$(Q_K)(T)$	γ_I	γ_P	γ_B
1	0	-	-	-
2	4.787	1.000e+00	1.909e+00	1.000e+00
3	6.999	1.088e+00	1.991e+00	1.850e+00
4	7.865	1.999e+00	1.999e+00	1.999e+00
5	8.711	1.420e+00	1.993e+00	1.993e+00
6	9.188	1.273e+00	1.999e+00	1.761e+00
7	9.470	1.495e+00	1.998e+00	1.896e+00
8	9.610	1.999e+00	1.999e+00	1.998e+00
9	9.669	1.822e+00	1.995e+00	1.999e+00
10	9.752	1.999e+00	1.967e+00	1.655e+00
K	E$(Q_K)(T)$	r_I	r_P	r_B
1	0	-	-	-
2	4.787	9.929e-01	6.576e-01	4.480e-03
3	6.999	9.999e-01	6.552e-01	1.138e-01
4	7.865	7.043e-01	6.323e-01	1.456e-01
5	8.711	9.999e-01	5.662e-01	3.887e-01
6	9.188	9.810e-01	6.599e-01	1.057e-01
7	9.470	9.999e-01	6.222e-01	1.825e-01
8	9.610	7.697e-01	6.520e-01	1.957e-01
9	9.669	4.824e-01	7.285e-01	4.523e-04
10	9.752	9.574e-01	6.826e-01	1.589e-05

7.3 Assuring Quality at the Receiver: buffering protection

In this section we study the protection strategies that a client is enable to use in a network with server failures in order to improve (and also ensure) a quality level. This analysis is from the client point of view, where it is not possible to change the network behavior. In particular, we discuss about the client buffer size to protect it of a server disconnection. It hardly depends on the network topology rebuild strategy, so, our analysis is done for different rebuild scenarios. Because of mathematical issues, we limit the concluded analytical expression of the buffer size to the case with only one server; we are working in a extension for the multi-source approach.

We are interested here in the impact of the disconnection failures on the quality of the stream as seen by the clients. The traditional way of analyzing this is to study performance measures of the systems, or dependability ones. Here, we will follow a performability-like approach, considering at the same time the failures (as defined before) and the resulting performance. However, instead of looking at standard metrics which we qualify here of *indirect* (loss rates, delays, or reliability, availability...), we will address the *ultimate target*, the quality of the stream, as perceived by the end user, using PSQA.

We show a possible use of the PSQA *minimal closed-form function* of Subsection 5.2.4. It is rational function $Q()$ of the chosen variables (here, LR and $MLBS$), mapping them into a MOS-like quality value. This means that we are approximating the perceived quality at t by $Q(LR(t), MLBS(t))$ where $LR(t)$ and $MLBS(t)$ are instantaneous values obtained by measuring.

Recall that $Q(LR, MLBS)$ has here the following form (seen in Subsection 5.2.4):

$$Q(LR, MLBS) = \frac{a\,LR + b\,MLBS}{c\,LR + d\,MLBS + \nu},$$

where $a = 0.00142128$, $b = 0.47242076$, $c = 0.00142128$, $d = 0.47242076$ and $\nu = 0.01$. In Figure 5.8 we can see this function, for the valid specific range of the input variables.

7.3.1 Buffering Model

We will use a simple model in order to study the way quality would be perceived in such a system. For now, we will consider the streaming from a single source. Consider the following simplifying assumptions. The time to process a data unit (a frame) is exponentially distributed with parameter μ. The arrival rate of frames to the client, when the node sending them to it is working, is obviously also μ. The client starts with a buffer containing N frames. When everything goes correctly, he receives frames at the same rate he plays them, so (we assume) the buffer level will remain constant. The node sending the frames can fail (typically, it will leave the network), which happens with some failure rate λ (that is, we assume an exponentially distributed sojourn time of a peer in the network). The network reconfigurates itself at points in time distributed as a Poisson process with rate $f = 1/T$. After a reconfiguration (an instantaneous action, in our model), the clients that had lost their stream because of a failure are connected again (to some peer/source). Let us call *cycle* the period between two consecutive network reconfigurations.

With the usual independence assumptions, we can build the following Markov process X describing the evolution of such a client, during a cycle. The state space is the set of non-negative integers. State 0 means that the node sending the frames to the observed client (from now, the server peer) is active, and everything goes normally. The client's buffer level is N. If the server fails, X moves to state 1 (with rate λ). State n, for $1 \leq n \leq N+1$ means that the server is down (left the system) and that the client has already played $n-1$ frames in that situation; the number of frames remaining in the buffer is $N-n+1$. State $N+m$, $m \geq 1$ means that the server is down and that the client has already lost $m-1$ frames (because their playtimes arrived and there was nothing to play). Figure 7.4 shows the transition diagram of process X. The loop at state 0 represents the fact that there is a regeneration of the whole network with rate f. Of course, if the client started with N frames in its buffer, after the next reconfiguration it will have a random number Y of frames, $Y \leq N$. We will look at that later. For the moment, we will use X to compute the loss rate at the client side. We call this a *performability* model because it takes into account failures, repairs, and services. Let π_n denote the probability that

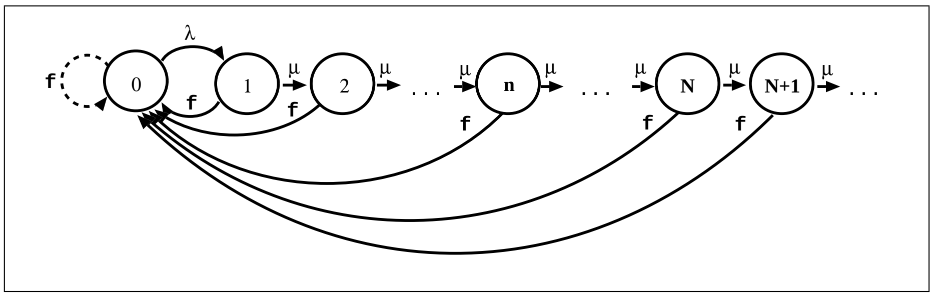


Figure 7.4: Evolution of a client during a cycle.

X is at state n in steady state. Then, the loss rate in the cycle will be defined as the probability of being at states $N+1$, $N+2$, etc. (which is, due to the PASTA property, the probability that at the moment we must play a frame, there is no frame in the buffer). In formal terms,

$$LR = \sum_{n \geq N+1} \pi_n.$$

Process X is always stable, and we have

$$\pi_0 = \frac{1}{1+\varepsilon}, \quad \varepsilon = \frac{\lambda}{f},$$

$$\pi_1 = \frac{\pi_0}{1+\delta}, \quad \delta = \frac{f}{\mu}$$

and for any $n \geq 2$,

$$\pi_n = \frac{\varepsilon}{1+\varepsilon} \frac{\delta}{(1+\delta)^n}.$$

After some algebra, this leads to

$$LR = \frac{\varepsilon}{1+\varepsilon} \frac{1}{(1+\delta)^N}.$$

The computation of the $MLBS$ is easier. First, we are now counting, so, we work on the canonical discrete time Markov chain embedded into X. In Figure 7.5 we see its transition probabilities. The parameters of this model are

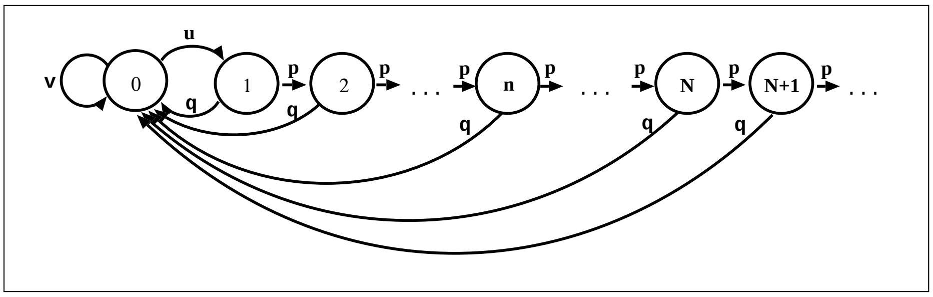


Figure 7.5: The canonical discrete time Markov chain embedded in X.

$$u = \frac{\varepsilon}{1+\varepsilon}, \quad v = 1-u, \quad p = \frac{1}{1+\delta}, \quad q = 1-p.$$

The probability that a burst of losses has size j being $p^{j-1}q$, we have:

$$MLBS = \frac{1}{q} = 1 + \frac{\mu}{f}.$$

Observe that if the cycle starts with N frames in the buffer, at the beginning of next cycle the buffer can contain N, $N-1$, ..., frames. We denoted by Y the number of frames present in the buffer at the beginning of next cycle. We have

$$\Pr(Y = N) = v + uq,$$

then, for $j = N-1, N-2, \cdots, 1$,

$$\Pr(Y = j) = up^{N-j}q,$$

and, finally,

$$\Pr(Y = 0) = up^N.$$

7.3.2 Buffering Protection Strategy

The main objective is to analyze how different sojourn times of a peer in the network, as well as different reconfiguration strategies, affect the quality of the video played by a client; and to react to this situation choosing an adequate buffer size in order to ensure a quality level.

We consider here two strategies of network rebuilding. In the first case, every $T = 1/f$ secs, on the average, the network is reconstructed. By means on message exchanges, the system discovers which nodes left and, if they were serving other peers, they replace them by other nodes to allow the former to continue receiving the signal. In the second case, we consider using a supplementary server peer that delivers the information to the client having lost its source until the network can be reconfigured.

Let us analyze the first scenario. We will consider two cases here: when a cycle begins the client has either 0 or 100 frames in the buffer. For the failure rate, we consider a broad range: $\lambda = 0.1, 0.01, 0.001$. Figure 7.6 shows the values for the loss rate metric in these scenarios as a function of the reconfiguration rate f, in a range from 1 sec to 20 sec (larger reconfiguration intervals decrease significantly the network performance). As expected, if the server spends more time in the network, the client observes less losses. A similar correlation can be observed with respect to the initial number of frames in the buffer: the larger this number, the smaller the loss rate.

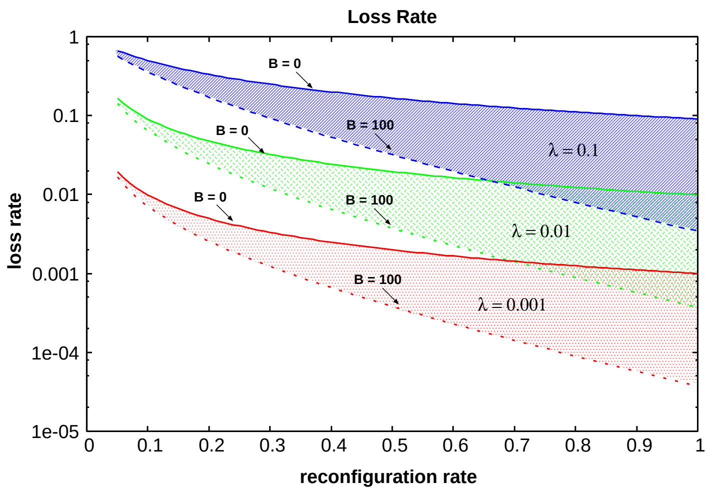


Figure 7.6: Loss rate, LR, measure for the first strategy of reconfiguration.

Now, let us consider the $MLBS$ metric. For the set of values of λ and f, the $MLBS$ reaches high values (600 to 30 frames). In these cases, we are out of the input domain of $Q()$. What simply happens is that such high values of $MLBS$ mean that quality will be very low (just 30 frames correspond to a second of signal). Moreover, observe that those very long bursts are rare. The probability of observing at least m consecutive losses is up^{N+m} in the case of a buffer having at the beginning of the cycle N frames, that is, $\varepsilon(1+\varepsilon)^{-1}(1+\delta)^{-N-m}$. This means that a very few number of peers will observe those bursts, leading to a poor quality. In other words, the situation where $MLBS = 600$ corresponds to a case of very rare losses, but when happening, arriving in long bursts.

In any case, a reconfiguration rate $f = 0.1$ means a mean reconfiguration period of $T = 10$ sec., and $f = 1$ means a reconfiguration every sec, on the average. This can be too low. In order to improve the performance of the system, let us consider the use of a special server peer that sends the stream to nodes having lost their servers, before the next reconfiguration point arrives. Let us consider that the mean time to such a special server starts transmitting information is 250 msec. In this case, as expected, the loss rate and the $MLBS$ are lower than in the previous one. Again we consider two different initial buffer sizes in the cycle: either 0 or

100 frames, and the same range for the failure rate λ. The largest loss probability is equal to 0.025 and the $MLBS$ varies between 7 and 8.4, as shown in Figure 7.7. The LR and $MLBS$

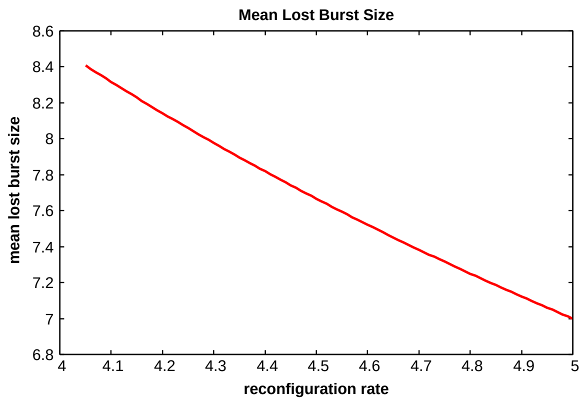


Figure 7.7: $MLBS$ measure for the second strategy of reconfiguration.

values were mapped into a perceived quality level using the $Q()$ function. The resulting quality Q had a small variation in this context: the maximum quality is equal to 0.87 and the minimum quality is equal to 0.82. This means that in the considered situation, quality is good enough.

Using our model and the minimal $Q()$ function introduced before, we can provide an analytical expression of the perceived quality as a function of the system's parameters λ, f, N and μ. Just replace LR and $MLBS$ in (5.1) by the corresponding expressions provided by the model.

This can allow to easily obtain rough answers to interesting questions. For instance, which is the minimal buffer size necessary to assure a given quality level? That is, what is the minimal value of N such that $Q(\lambda, f, N, \mu) \geq Q_0$? After some algebra, this leads to:

$$N \geq \frac{\log \dfrac{cQ_0 - a}{\left(1 + \dfrac{f}{\lambda}\right)\left[(bQ_0 - d)\left(1 + \dfrac{\mu}{f}\right) - Q_0\nu\right]}}{\log\left(1 + \dfrac{f}{\mu}\right)}.$$

For $Q_0 = 0.87$, it is possible to see that the necessary buffer size is very small, with a maximum of 28 frames ($\sim$ 1 second) when we have $\lambda = 0.1$ and the highest reconfiguration rate $f = 5$.

In this section, we considered two network design parameters in our analysis: the reconfiguration rate f of the network and the number N of frames in the buffer just after a reconfiguration. The coupling between PSQA and the performability model allows to evaluate the impact of f and N on the QoE. Current work is being done now to extend these results to multiple sources and to use transient views of the system.

7.4 Summary and Conclusions

This chapter proposes some transient models that may be useful for the design of a live-video P2P distribution system following a multi-source procedure.

A first focus is on how to ensure a high QoE for the users, when simple multi-source policies are used. Three policies are considered. In the first one, the original stream is entirely replicated and independently sent from K different servers. In the second one, the original stream is divided into K independent substreams, one per server, so that their union reconstructs the original signal. In the third one, the original stream is also divided into K independent substreams, but at each stream some redundancy is added, so that the loss of any substream can be recovered from the $K - 1$ remaining ones. To evaluate the different options, we develop analytical models for computing the loss process as functions of the total number of servers K and the number of servers in working order, for each of the cases. We also give a simple transient Markovian model with an explicit analytical solution for computing the distribution of the number of working and failed servers at a given time T.

We then focused our analysis on how to maximize the delivered QoE of an improved *multi-source technique* based on the heterogeneous peers' lifetimes. This improved technique provides good quality to the end-users, mitigating the impact of losses coming from peers disconnections. The main results concern the joint impact of different frame type losses on the QoE, using the PSQA methodology, and how to identify an optimal parameter setting, in order to obtain the best possible quality level for a given peers' dynamics. This led to the method that we use in our prototype; the main objective was to provide a methodology that can be used in other P2P systems with similar constraints and users' behavior, in order to set the best possible QoE.

The QoE perceived by the end users is captured using different PSQA functions, showing the applicability of the PSQA methodology to network design. The developed models are experimentally evaluated using configurations computed from realistic parameters (computing using information collected at our video delivery reference service).

The improved multi-source technique is applied at the server side of the network in order to ensure an optimal delivery. In this chapter, we also study the protection strategy at the client side. We present a simple Markov performability model combined with PSQA. The proposed model allows us to describe the evolution of a peer client during a *cycle*, i.e, the period between two consecutive network reconfigurations. Using PSQA we are able to work with the perceived quality as a function of the design parameters and thus to perform a quantitative analysis of our system.

Chapter 8

Structured Overlay Peer-to-Peer based on Quality Guarantees

This chapter explores a preliminary design of a tree-based overlay topology, with a centralized control, for a P2PTV network. The aim is to provide good quality levels in a highly dynamic P2P topology, where the frequent connections/disconnections of the nodes make it difficult to offer the QoE needed by the clients.

We consider the problem of selecting which peer will serve which other, i.e. the topology construction. The election of the peer edges is important for the robustness of the tree network, especially when the peers are heterogeneous in their capabilities.

Section 8.2 introduces a mathematical programming model to maximize the global expected QoE of the network (evaluated using PSQA), selecting a P2P connection overlay topology which enhances the robustness. In addition, we provide approximated algorithms to solve the proposed model (Section 8.3), and we apply them to solve a case study based on real life data (Section 8.4). The main contributions of this part of our work are then summarized in Section 8.5.

The model and preliminary results have been presented at the 4th Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS'07) [41] (listed at the Introduction as [lagos07]). An approximated algorithm that uses a Random Neural Network (RNN), nowadays submitted [213] (listed at the Introduction as [submitted08b]), has been designed and implemented with the participation of A. Morón and M. Martinez, who completed their engineering project at the Universidad de la República, Uruguay, in this subject.

8.1 Background Definitions

We study here the characteristics of a tree-based overlay solution for live video broadcasting. In particular, in this section we develop a mathematical programming model for the stream assignment problem in order to maximize the expected perceived quality taking into account the network dynamics.

Time. The system is reconfigurated at discrete points in time, every Δt. Let us use Δt as unit of time, and denote by I_n the nth interval $[t_n, t_{n+1})$ for each n. This is a simplifying assumption, that allows to evaluate the main difficulties; in reality, a more continuous solution can be implemented.

Distribution scheme. The stream is originally sent to the clients (or terminals) by a brodcaster node s. Most clients act also as servers, relaying streams to other clients. For this purpose, each node v has an available output bandwidth BW_v^{out}. The system distributes a single stream of live video by means of K sub-streams denoted by $\sigma_1, \sigma_2, ..., \sigma_K$. Each substream σ_k is sent with a constant bandwidth bw_k. The total bandwidth used to send the stream is $\sum_{k=1}^{K} bw_k$. When a client receives the K substreams, it reproduces the stream with perfect quality. If it does not receive all the K substreams, it will reproduce a stream that can have lower quality, depending on which substreams are received and which redundant scheme is used (the particular multi-source streaming technique used is presented in Chapter 7). To evaluate the quality at the client side, we use the PSQA technology.

Dynamics. The evolution of the system from t_n to $(t_{n+1})^-$ is as follows: some nodes leave in I_n, possibly disconnecting other clients in some substreams; at the same time, some nodes enter the network requesting for connection; they remain isolated from the rest of the nodes until t_{n+1} when the new reconfiguration action is performed. The goal of the reconfiguration is to reconnect the disconnected nodes and to connect the new arrivals to the network. The connexion scheme always builds trees of peers. At time $(t_{n+1})^-$, just before the reconfiguration, the general situation is the following. For each substream σ_k there is a *main tree*, $\mathcal{P}_k$, containing (at least) the source s; all its nodes receive substream σ_k. There are also $M_k \geq 0$ other trees, disjoint between them and with $\mathcal{P}_k$, denoted by $\tau_{k,1}, \tau_{k,2}, \cdots, \tau_{k,M_k}$; their nodes do not receive σ_k. The set of trees associated with substream σ_k is called *plane k*. A *perfect network* is a system where for each substream σ_k there is only one directed tree ($\mathcal{P}_k$, the main one), meaning that $M_k = 0$. Figure 8.1 shows a graphical description of the model.

Optimization. As said before, the reconfiguration action will try to build a perfect network, and, among the huge number of possible perfect networks, it will try to build a robust one. For this purpose, we keep statistics about the nodes' lifetime behavior allowing us to evaluate their expected departure times from the network. Specifically, we maintain an estimate p_i of the probability that node i remains connected in the current period, when it is connected at the beginning (see below).

8.2 P2P Robust Assignment Model

We propose now an Integer Mathematical Programming Model which contemplates the P2P dynamics (described above) in a time interval $[t_n, t_{n+1})$. Consider the system at time t_n and let $\mathcal{N}(t_n)$ be the set of connected nodes at t_n ($N = \|\mathcal{N}(t_n)\|$). Define for each $k \in \{1, 2, \cdots, K\}$ and $i, j \in \mathcal{N}(t_n)$,

$$x_{i,j}^k = \begin{cases} 1 & \text{if node } i \text{ sends } \sigma_k \text{ to node } j, \\ 0 & \text{otherwise,} \end{cases}$$

$$y_{i,j}^k = \begin{cases} 1 & \text{if node } i \text{ precedes node } j \text{ in the tree containing } j \text{ in plane } k, \\ 0 & \text{otherwise.} \end{cases}$$

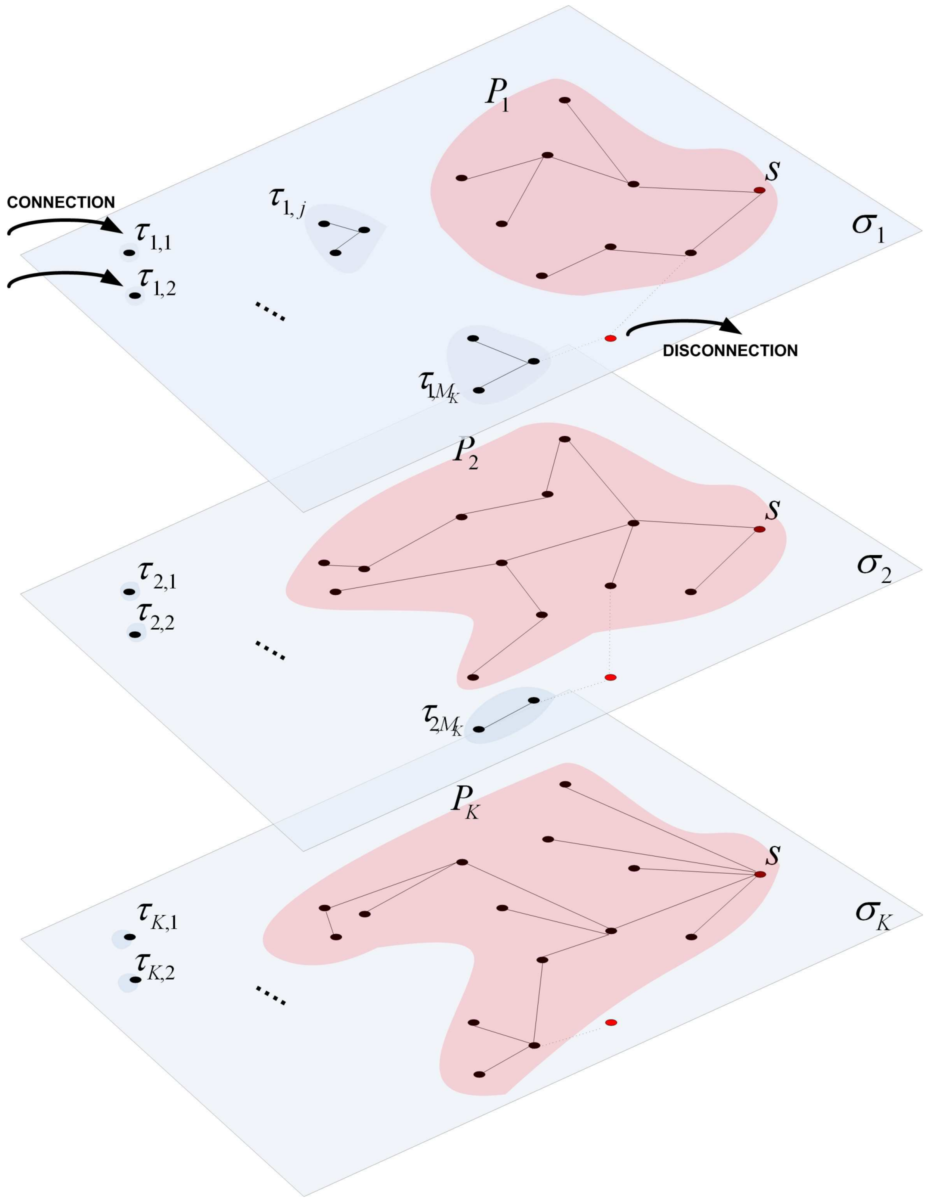


Figure 8.1: Graphical model description.

Since the perceived quality at node i depends on which substream is received by i, we assume available a function $f()$ of K variables such that the quality at time t_n experienced by node $i \in \mathcal{N}(t_n)$ (and measured using PSQA) is

$$Q_i = f(y^1_{s,i}, y^2_{s,i}, .., y^K_{s,i}).$$

For all $i \in \mathcal{N}(t_n)$, let z_i be the binary random variable defined at time t_n by

$$z_i = \begin{cases} 1 & \text{if node } i \text{ will remain connected until } (t_{n+1})^-, \\ 0 & \text{otherwise,} \end{cases}$$

where $\Pr(z_i = 1) = p_i$.

The sets $\left\{x^k_{ij}\right\}$ and $\left\{y^k_{ij}\right\}$ specify the network configuration at t_n whereas the network configuration at $(t_{n+1})^-$ is determined by the variables $\left\{\tilde{x}^k_{ij}\right\}$ and $\left\{\tilde{y}^k_{ij}\right\}$ satisfying the following relations:

$$\tilde{x}^k_{i,j} = z_i x^k_{i,j} z_j,$$

$$\tilde{y}^k_{i,j} = \tilde{x}^k_{i,j} + \sum_{l=1}^{N} z_i \tilde{y}^k_{i,l} \tilde{x}^k_{l,j}, \forall i \in N, \forall j \in N - \{s\}, \forall k \in K,$$

$$\tilde{y}^k_{i,j} \leq y^k_{i,j}.$$

The PSQA evaluation of the quality as perceived by node i at time $(t_{n+1})^-$ is a random variable which is a function of $\left\{\tilde{y}^k_{s,i}\right\}$ and r.v. $\{z_i\}$. We will maximize the expected value of the perceived quality of all the nodes in the network. Actually, we use a scaled expression,

$$\mathrm{E}\left\{\frac{\sum_{i=1}^{N} \tilde{Q}_i}{\sum_{i=1}^{N} z_i}\right\} = \sum_{\overrightarrow{z} \in [0,1]^N} \frac{\sum_{i=1}^{N} \tilde{Q}_i(\overrightarrow{z})}{\sum_{i=1}^{N} \overrightarrow{z}_i} p(z = \overrightarrow{z}),$$

where $\tilde{Q}_i = f(\tilde{y}^1_{s,i}, \tilde{y}^2_{s,i}, .., \tilde{y}^K_{s,i})$.

See Figure 8.2 for a summary of the complete model.

8.3 Looking for a Solution

The model developed in previous section is a stochastic, mixed integer non-linear programming model, where the objective function is not given explicitly, but algorithmically (using the PSQA built function for the QoE). It does not seem possible to explicitly solve it, even for small size problems, while real-life applications require to take into account a large number of nodes. Of course, in larger networks, it is possible to divide the nodes into small groups (for example, based on some proximity parameter), but this implies a separation from the global optimal network. In some way, with the mathematical formalization we want to find the theoretical efficiency bounds of a network with these characteristics, instead of analyzing its effective deployment.

$$\max \mathsf{E}\left\{\frac{\sum_{i=1}^{N}\tilde{Q}_i}{\sum_{i=1}^{N} z_i}\right\}$$ // global expected perceived quality

st:

$y^k_{i,j} + y^k_{j,i} \le 1, \forall i,j \in N, \forall k \in K,$ // loops are not allowed

$y^k_{i,j} = x^k_{i,j} + \sum_{l=1}^{N} y^k_{i,l} x^k_{l,j}, \forall i \in N, \forall j \in N - \{s\}, \forall k \in K,$ // precedence constraints

$\sum_{i=1}^{N} x^k_{i,j} \le 1, \forall j \in N, \forall k \in K,$ // each substream arrives from only one node

$\sum_{k=1}^{K} \{bw^k \sum_{j=1}^{N} x^k_{i,j}\} \le BW^{out}(i), \forall i \in N,$ // bandwidth capacity constraints

$x^k_{i,j} = 1, \forall i \in N, \forall j \in N, \forall k \in K | x^k_{i,j} \in E^k_t,$ // network configuration at t

$\tilde{x}^k_{i,j} = z_i x^k_{i,j} z_j, \forall i \in N, \forall j \in N, \forall k \in K,$ // a link is preserved if source and terminal do not leave

$\tilde{y}^k_{i,j} = \tilde{x}^k_{i,j} + \sum_{l=1}^{N} z_i \tilde{y}^k_{i,l} \tilde{x}^k_{l,j}, \forall i \in N, \forall j \in N - \{s\}, \forall k \in K,$ // $\tilde{y}$ represents the precedence at $(t_{n+1})^-$

$\tilde{y}^k_{i,j} \le y^k_{i,j}, \forall i \in N, \forall j \in N, \forall k \in K,$

$z_i \sim Bern(p_i), \forall i \in N,$ // random variables with Bernoulli distribution (parameters p_i)

$x^k_{i,j}, y^k_{i,j} \in \{0,1\}, \forall i \in N, \forall j \in N, \forall k \in K$

Figure 8.2: Mathematical Programming Model.

Instead of using exact solvers, we will look at some heuristics that can find feasible solutions, hopefully of good quality. In particular, we develop a greedy constructive heuristic, and two version of a GRASP-based metaheuristic.

As we have seen in the formal model description, the time line is divided into intervals of the form $I_n = [t_n, t_{n+1})$. In this way, we simplify the problem by considering only a discrete number of points $(t_0, t_1, t_2, ...)$ on the time line. Just before each interval I_n (we will call this instant t_n^-), the state of the network is the result of all the changes which have occurred in the previous interval, I_{n-1}. During this period, some nodes may have left the network (willingly or not), while new clients may have requested connection in order to receive the video stream. Therefore, new disconnected trees may have appeared. It is at this instant t_n that our heuristic algorithm is executed in order to connect the new nodes and disconnected trees to the main tree, and this has to be done in such a way that the expected quality is maximized, taking into account the upstream bandwidth restrictions of each node. Theoretically, we assume an instantaneous execution of the algorithm, so that at instant t_n we will have our reconfigured system[1]. Then, new connections and disconnections will occur during the interval I_n, and the algorithm will be executed again at the beginning of I_{n+1}. The output of the algorithm will then be a set of *assignments*, where each assignment indicates to which node of the main tree we will attach a disconnected tree (or a new node[2]) in order to enable the transmission of a certain substream to that tree. An assignment can then be seen as a triplet (parent, child, substream), where the *child* is the root of the disconnected tree which will be attached to a node of the main tree (the *parent*), in order to receive the corresponding *substream*. As it was stated before, the algorithm should produce a solution which maximizes the global perceived quality, that is, a solution which satisfies the restriction of the formal model described before. However, we have seen that this is an extremely difficult problem, so the algorithm will try to construct a solution which is as close to the optimal one as possible.

8.3.1 Greedy heuristic solution

The *Greedy* algorithm constructs a solution based on a simple iterative decision process. The procedure used is outlined in Figure 8.3. The procedure $\mathsf{GREEDY}_{\mathsf{p2p}}$ is executed with the initial state of the network g_0. First, a list of all possible assignments is determined given the current state of the network. That is, a list of all possible (parent, child, substream) triplets is constructed to determine which trees need to be reconnected to the main one and which nodes have enough upstream bandwidth available in order to transmit a certain substream to a disconnected tree (line 2). While an assignment will be possible, the procedure will choose iteratively the "best" of them, until one of the two following conditions will be satisfied:

- all the disconnected trees have been reassigned to the main tree for each substream;
- no nodes in the main tree have enough bandwidth to transmit a substream to a disconnected tree.

[1]In the actual implementation, this "instantaneous execution" would mean that the execution time of the algorithm has to be negligible compared to the duration of each interval I_n.

[2]New nodes can be seen as disconnected trees with only one element, so there is no need to make a distinction.

```
Procedure GREEDY_p2p
Input: $g_0$

1:  $g \leftarrow g_0$
2:  $C \leftarrow$ FindAllAssignmentsCandidates$(g)$
3:  while $C \neq \emptyset$ do
4:      $C_{PARENT} \leftarrow$ SelectBestConnectedParent$(g, C)$
5:      $C_K \leftarrow$ SelectKLargestChilds$(g, C_{PARENT})$
6:      $a \leftarrow$ SelectBestCriteria$(C_K)$
8:      $C \leftarrow$ UpdateCandidates$(g, a, C)$
9:      $g \leftarrow$ ApplyAssignments2Graph$(g, \{a\})$
10: end while

11: return $g$
```

Figure 8.3: GREEDY algorithm

To choose the "best" assignment, we first select a subset of the set of all possible assignments (line 4). The assignments of this subset have the parents that provide the highest probability to be connected at the final of the current period. This is computed using the disconnection probability of each node in the path from the parent to the root. Then (line 5), from this subset, we select one assignment per sub-stream (K in total). We choose the assignments that connect the largest number of nodes. If, at the same sub-stream, two assignments have the same number of nodes, we select the one that has the larger available bandwidth. Then, the candidate assignment that has the highest enhancement is chosen (line 6)[3]. The list of candidates is then updated (line 8). The assignment is applied to the graph (line 9), since the tree corresponding to the selected assignment will no longer be disconnected, and the upstream bandwidth of the node to which that tree will be connected will have decreased. This process is repeated until there is no more possible assignments to do.

8.3.1.1 Enhancement criteria of a given assignment

Two different criteria are considered for estimating the enhancement provided by a certain assignment.

Current Perceived Quality. The "current" criterion selects the assignments which provide a better perceived quality for the network at the current time.

Future Perceived Quality. The "future" criterion gives higher priority to the assignments that will result in better perceived quality for the next iteration. This is computed by a Monte Carlo estimation (by randomly simulating sequences of nodes entries and exits in the period). The precision and the time consumed by the estimation depend on the number of simulations (*trials*).

[3]Three different criteria are considered for estimating the enhancement provided by a certain assignment (see below).

```
Procedure GRASP_p2p
Input: $g_0$

1:  $q^* \leftarrow -1$
2:  for $i = 1$ to $i_{max}$ do
3:      $g_i \leftarrow$ Construction_p2p$(g_0)$
4:      $q_i \leftarrow$ PSQA_expected$(g_i)$
5:      if $(q_i > q^*)$ then
6:          $q^* \leftarrow q_i$
7:          $g^* \leftarrow g_i$
8:      end if
9:  end for

10: return $g^*$
```

```
Procedure Construction_p2p
Input: $g$

1:  $C \leftarrow$ FindAllAssignmentsCandidates$(g)$
2:  $A \leftarrow \emptyset$
3:  while $C \neq \emptyset$ do
4:      $C_{RCL} \leftarrow$ GenerateRCL_p2p$(g, C)$
5:      $a \leftarrow$ SelectRandom$(C_{RCL})$
6:      $A \leftarrow A \cup \{a\}$
7:      $C \leftarrow$ UpdateCandidates$(g, A, C)$
8:  end while
9:  $g_c \leftarrow$ ApplyAssignments2Graph$(g, A)$

10: return $g_c$
```

Figure 8.4: GRASP-based algorithm

8.3.2 Algorithmic solution based on GRASP

GRASP [270] is a well known metaheuristic that has been successfully used to solve many hard combinatorial optimization problems. It is an iterative process which operates in two phases. In the *Construction Phase* an initial feasible solution is built by means of a greedy algorithm. This algorithm must also have a random component, so that different executions lead to different results. The neighborhood of this initial solution is then explored in the *Local Search Phase*, in order to improve locally the solution.

In the following sections, we describe the GRASP-based algorithm which is used to reassign connections in the network after a certain interval of time.

8.3.2.1 Construction phase

The objective of the *Construction phase* of the algorithm is to use a greedy and randomized procedure to construct an *initial solution* for the problem. This solution can then be refined in the *Local Search phase*. The procedure used for the Construction phase is outlined in Figure 8.4.

When the procedure $\mathsf{GRASP}_{\mathsf{p2p}}$ is executed with the initial state of the network g_0, i_{max} initial solutions are constructed (lines 1 - 3) using a procedure $\mathsf{Construction}_{\mathsf{p2p}}$. Since this procedure has a random component, n executions of the procedure will lead to n different results. Then, from the i_{max} solutions obtained, we pick the one which gives the network the better global perceived quality in the future interval (lines 4 - 7). To evaluate the global perceived quality in the future interval for each solution, we use the procedure $\mathsf{PSQA}_{\mathsf{expected}}$, which relies on statistical data about the nodes in order to "predict" the connections and disconnections in the future interval and see how a given solution is affected by these events. We will now examine the procedure used to construct each one of these initial solutions, $\mathsf{Construction}_{\mathsf{p2p}}$. First of all, a list of all possible assignments is determined given the current state of the network (line 1). Then, using a specific criterion, we select a subset of this set of all possible assignments (line 4) which is called *Restricted Candidate List* (RCL). The criterion used to select the "best" assignments is configurable[4]. Here we use the same enhancement criteria presented

[4]A specific interface in the algorithm must be implemented for each criterion.

for the greedy algorithm, i.e. the current and future perceived quality. This subset will contain the assignments which provide the "better" quality improvement to our solution for the current state of the network according to the improvement criteria specified. This is the *greedy* part of the GRASP algorithm. Then, one of these candidate assignments is randomly selected from the RCL (line 5) and added to a set A (line 6), this constitutes the *random* component of the algorithm. The list of candidates is then updated (line 7), since the tree corresponding to the selected assingment will no longer be disconnected, and the upstream bandwidth of the node to which that tree will be connected will have decreased. This process is repeated until the set A satisfies one of the two following conditions:

- All the disconnected trees have been reassigned to the main tree for each substream;
- No nodes in the main tree have enough bandwidth to transmit a substream to a disconnected tree.

The set A of assignments is then our *initial solution*.

8.3.2.2 Local search phase

Our customized GRASP algorithm does not contain the traditional local search phase. In the local search phase, the solution built in the construction phase is improved by exploring its neighborhood in order to find a local optimum. In the construction phase of our algorithm, a feasible solution is built applying successive individual *assignments* to the initial network configuration. Let $\mathcal{T} = \{a_1, a_2, ..., a_k, ..., a_n\}$ be the set of assignments applied to build a feasible solution. In this case, for the local search phase, the choice of the neighborhood structure should be one where small variations on the assignments of $\mathcal{T}$ produce feasible solutions. The main problem here is that a particular assignment is strongly dependent on the previous one. In this sense, if we want to change assignment a_k, there is a chance that next assignments $a_{k+1}, ..., a_n$ could not be applied. This is easily seen when, for example, assignment a_{k+1} involves, as a parent, one of the new available parent nodes added in a_k. That means that a_k must not be changed in that particular case.

A local search phase is not useful in this specific problem, because it introduces a new complex combinatorial problem. Therefore, we will not implement this phase. Instead, we propose an alternative way to improve the algorithm, based on the RNN model, of the solutions built during the GRASP construction phase. It consists on replacing the assignment selection method in a random fashion, in order to explore new zones of the solution space, which may allow us to find better solutions.

8.3.3 GRASP improvement using RNN

The Random Neural Network (RNN) model [101] has been applied with success to many different optimization problems such as the *minimun vertex covering problem* [107] and the *traveling salesman problem* [110]. In the RNN model, signals in the form of spikes of unit amplitude circulate among the neurons. Positive signals represent excitation, and negative signals represent inhibition. Signals coming from the outside form Poisson processes. These models have been presented in Chapter 4 and used for learning purposes in deriving our PSQA modules (Chapter 5). Here, the main difference is in the neuron network topology, which, as we will see, is arbitrary. The model is parameterized by the following elements: the number n of neurons, the firing rate r_i of neuron i, the probability p^+_{ij} (resp. p^-_{ij}), for a signal sent by i, to go to j as a positive (resp. negative) one, the probability $d_i = 1 - \sum_j (p^+_{ij} + p^-_{ij})$ of the signal to go outside, and the exogenous rates α_i and β_i of the signal flows arriving at i. In the stable case, probability q_i that neuron i is excited, in steady state, is given by [101]

$$q_i = \frac{\lambda^+_i}{r_i + \lambda^-_i}$$

where λ^+_i and λ^-_i are defined as $\lambda^+_i = \sum_{j=1}^{n} q_j w^+_{ji} + \alpha_i$, $\lambda^-_i = \sum_{j=1}^{n} q_j w^-_{ji} + \beta_i$, and the *weights* are $w^+_{ji} = r_i p^+_{ji}$, $w^-_{ji} = r_i p^-_{ji}$. Observe that the information contained in the RNN is represented by the frequency at which the signals travel (the set of weights). The computation of the q_i's is thus a fixed point problem. See [101] for details about existence and unicity. During this computation, if we get a value $q_i > 1$ then we force $q_i = 1$ until convergence (we say neuron i is *saturated*).

Following directly the ideas in [40], we use a RNN in the GRASP construction phase. We replace the RCL generation and the random assignment selection (lines 4-5 of Construction$_{\mathsf{p2p}}$, Figure 8.4) with an RNN improved algorithm (explained below). In addition, on each construction, a Bernoulli parameter, rnn, is used to determine which assignment selection is used: this parameter indicates the probability of using the RNN based method for constructing solutions in that iteration, instead of the original GRASP based method[5].

Next, we present the algorithm based on the RNN model to solve the problem of selecting an specific assignment during the iterative process of the GRASP construction phase.

We define a *neuron* as the pair (node, substream), denoted by v^k, which belongs to one of the following classes:

$$\mathcal{A} = \{v^k \mid v \in \mathcal{P}_k \ , \ BW^{out}_v \geq bw_k \ ; \ k = 1, \dots, K\}$$
$$\mathcal{O} = \{v^k \mid v \in \{\tau_{k,1}, \dots, \tau_{k,M_k}\} \ , \ M_k \geq 0 \ ; \ k = 1, \dots, K\}$$

We use $\mathcal{A}^k$ and $\mathcal{O}^k$ to denote the subsets of neurons of $\mathcal{A}$ and $\mathcal{O}$ respectively, where the nodes of the neurons are contained in the substream σ_k. In addition, we define a *possible assignment* as the pair (i, j) where $i \in \mathcal{A}^k$ and $j \in \mathcal{O}^k$, $\forall k \in \{1, \dots, K\}$. The excitatory and inhibitory weights of the neural network are set as follows:

$$w^+_{ij} = \frac{AIC_{ij}}{\overline{AIC_i}}, \quad \text{if } (i,j) \text{ is a } \textit{possible assignment,} \tag{8.1}$$

$$w^-_{ij} = 1, \quad \text{otherwise,} \tag{8.2}$$

where AIC is the *"assignment improvement criteria"* function corresponding to the greedy criteria implemented in the same way it is done in GRASP. AIC_{ij} is the evaluation of the AIC function after the assignment (i, j) is applied and $\overline{AIC_i}$ is the average of all AIC function evaluations corresponding to all possible assignments containing neuron i as a parent, which

[5]The algorithm GRASP+RNN with the parameter $rnn = 0$ is equivalent to the simple GRASP described before.

```
Procedure Select_RNN_Assignment_p2p
Input: $C$, $\alpha$, $MaxIter$

1:  for each $(i, j) \in C$ do
2:      compute $w^+_{ij}$ {using the equation 8.1}
3:  end for
4:  $q = (q_1, \dots, q_n) \leftarrow$ Initialize($n$) {$n$-orphans neurons vector}
5:  compute $F(q)$ {using the equation 8.3}
6:  while $||F(q) - q||_2 > \alpha$ and $iter < MaxIter$ do
7:      $q \leftarrow F(q)$
8:      compute $F(q)$ {using the equation 8.3}
9:      $iter \leftarrow iter + 1$
10: end while
11: $j^* \leftarrow \underset{j}{\text{argmax}}\{q_j \mid j = 1, \dots, n\}$
12: $i^* \leftarrow \underset{i}{\text{argmax}}\{\overline{\mathsf{AIC}}_i \mid (i, j^*) \in C\}$ {AIC is the greedy function}
13: return $(i^*, j^*)$
```

Figure 8.5: RNN Algorithm

is formally defined as $\overline{AIC}_i = \sum_{j \in \mathcal{O}^k} AIC_{ij} / \left|\mathcal{O}^k\right|$ where $i \in \mathcal{A}^k, \forall k \in \{1, \dots, K\}$. All other network parameters are set to zero. Using this set up, note that the firing rate is computed as $r_i = \sum_j (w^+_{ij} + w^-_{ij})$. The connection weights have been chosen in a manner such that the neural network captures connectivity information about the possible assignments which can be done. When the neural network is set up in this manner, an excited neuron will have a greater excitatory effect on neurons which have high excitatory connections (i.e., higher assignment improvement) with it. Basically, the procedure in our RNN consists in artificially exciting the *parent neurons* in $\mathcal{A}$ and then computing the values q_j for all the *orphan neurons* in $\mathcal{O}$ iteratively solving the fixed point problem using 8.3.3. According to the RNN set up presented previously, we simplify Gelenbe's equation as follows:

$$\begin{aligned}
q_j &= \frac{\sum_i q_i w^+_{ij} + \alpha_j}{r_j + \sum_i q_i w^-_{ij} + \beta_j} = \frac{\sum_i q_i w^+_{ij}}{\sum_i (w^+_{ji} + w^-_{ji}) + \sum_i q_i w^-_{ij}} \\
&= \frac{\sum_{i \in \mathcal{A}^k} w^+_{ij}}{\sum_i w^-_{ji} + \sum_{i \in \mathcal{A}^l \cup \mathcal{O}, l \neq k} q_i} = \frac{\sum_{i \in \mathcal{A}^k} w^+_{ij}}{\sum_i w^-_{ji} + (\sum_{p \in \mathcal{A}^l, l \neq k} q_p + \sum_{c \in \mathcal{O}, c \neq j} q_c)} \\
&= \frac{\sum_{i \in \mathcal{A}^k} w^+_{ij}}{|\mathcal{A}| + |\mathcal{O}| - 1 + (|\mathcal{A}| - |\mathcal{A}^k| + \sum_{c \in \mathcal{O}} q_c)} \\
&= \frac{\sum_{i \in \mathcal{A}^k} w^+_{ij}}{2|\mathcal{A}| - |\mathcal{A}^k| + |\mathcal{O}| - 1 + \sum_{c \in \mathcal{O}, c \neq j} q_c}, \qquad \forall j \in \mathcal{O}^k \text{ with } k = 1, \dots, K. \qquad (8.3)
\end{aligned}$$

Figure 8.5 presents the pseudocode of the RNN algorithm. First, we compute the excitatory weights for all the possible assignments (set C, candidates) (lines 1-3). This is necessary for the computation of $F(q)$ as can be seen in Equation 8.3 (line 5 and 8). Before the iteration, we need to initialize vector q (line 4); each neuron excitation probability is usually initialized with a near zero value because this way, the network iterates until convergence in a smaller number of steps. The stop criteria depends on the threshold α, and in case the iteration does not converge, we limit the number of iterations to $MaxIter$. When convergence is reached, we choose the neuron j with the highest value q_j (line 11) and then we compute the values $\overline{AIC_i}$ for all the neurons $i \in \mathcal{A}$ where (i, j) is a possible assignment. Finally, we choose the neuron i with the highest value $\overline{AIC_i}$ (line 12) and the assignment (i, j) is selected as the solution.

8.4 Preliminary performance estimation in a hybrid CDN-P2P structured network

In the introduction to the background technologies (Section 2.3.4), we discussed the advantages of combining a Content Delivery Network (CDN) architecture with a Peer-to-Peer (P2P) network. The main idea was to extend the resources available in the CDN with the efficiency of P2P delivery, keeping the high quality, security, and centralized control of the CDN.

In this section, we show how our tree-based overlay algorithms (presented in previous section) performs in a hybrid architecture. When the data is available, we use real scenarios coming from our video delivery reference service.

8.4.1 Generation of realistic test cases

We generate some instances from the collected real data of our reference service. A long trace of users connections/disconnections is used in order to specify the instances. Here, we will present five of these instances that come from a particular football match period. During this match, a server fails in the CDN, forcing the reconnection of many users over a short period. Studying this particular case allow us to analyze the behavior of our algorithm also during a flash crowd.

Each instance has a total duration of one hour. We consider a rebuilding period of ten seconds; therefore, there are a total of 360 iterations[6] in each instance.

One instance has 30 simultaneous users on the average, while the other four instances have 60 simultaneous users on the average. Figure 8.6 shows the connection/disconnection dynamics in the system (the four sixty users instances are tagged in the figure with different K values, which be used later in the experiments). Usually, per iteration, between 0 and 1 users change their state (connect/disconnet to/from the network), except at the 22th minute (130th iteration), when a server of the CDN fails, and disconnects a lot of peers (which will be immediately reconnected to other server).

We choose the upload bandwidth capacity of the nodes sampling from an uniform distribution between 256 Kbps and 1024 Kbps, because this information is not available in our reference CDN service[7]. Figure 8.7(a) shows the obtained bandwidth distribution of each instance. The upload bandwidth available at the broadcaster (the root server) is 5 Mbps[8]. The total available bandwidth in the peer-to-peer network changes with the peers connections/disconnections,

[6]The algorithms presented before have to be executed 360 times per instance.

[7]Upload bandwidth bounds estimated from the popular `http://broadbandreports.com` speed-test, on May, 2007.

[8]Considering a streaming with bitrate 640 Kbps, only seven clients can be served simultaneously in a traditional CDN architecture.

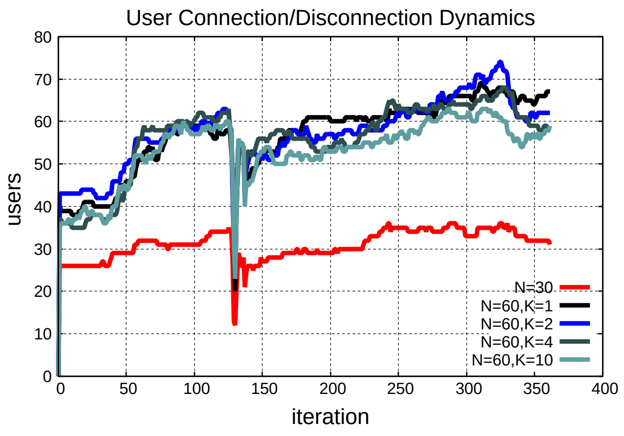


Figure 8.6: Evolution of the number of users connected in the system.

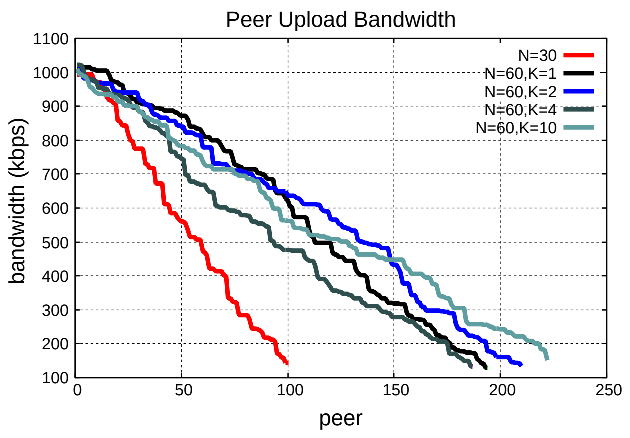
Figure 8.7(b) shows the evolution of the global available bandwidth (including the upload capacity of each connected peer).

The probability to remain connected ($\Pr(z_i = 1) = p_i$ in our model) is obtained from statistical information of each particular user included in the instances. For each user, we estimate, from his average lifetime behavior in several sessions, the probability to remain connected in the next ten seconds. For this purpose, we use an exponential distribution[9]. Figure 8.8 shows the distribution of the probability to remain connected in the next 10 seconds of each peer.

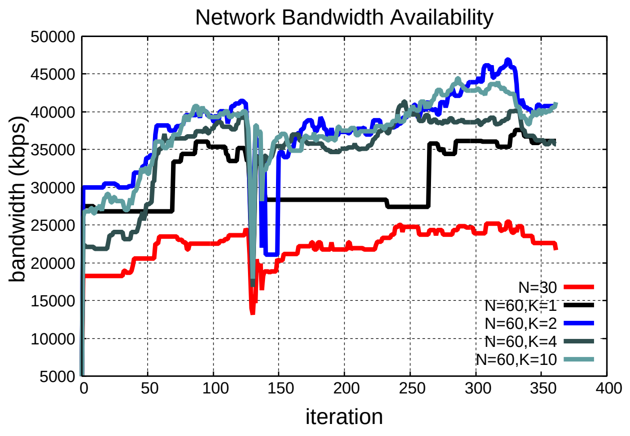
To deliver the video, we use four multi-source streaming configurations. We use the optimal configurations specified in Table 7.12, for $K \in \{1, 2, 4, 10\}$ (see Section 7.2 for details about the procedure used to obtain these configurations). The bitrate of the original stream is 512 Kbps, and a global redundancy of $r = 25\%$ is applied to achieve a overall bandwidth of 640 Kbps (considering all the sub-streams). There is no constraint on the upload bandwidth of the multi-source streaming configurations chosen. Table 8.1 shows the bandwidth consumed per sub-stream for each configuration.

The perceived quality depends on which sub-stream is received by the user. The model uses a function $f()$ of K variables such that the quality at time t_n experienced by node i is $Q_i = f(y^1_{s,i}, y^2_{s,i}, .., y^K_{s,i})$. For instance, if $K = 1$, user i has perfect quality when he receives the stream ($Q_i = f(1) = 10$), and zero quality otherwise ($Q_i = f(0) = 0$). We compute the quality function $f()$ from the PSQA *complex function* of Section 5.3.3, using the loss process model described in Section 7.2. Tables 8.2 and 8.3 show the quality function $f()$ for $K = 2$ and $K = 4$ respectively. The quality function $f()$ for the multi-source streaming with ten sub-streams ($K = 10$) is not shown here to save space. Observe that higher sub-streams indexes have larger impact in the perceived quality (as we expect from the methodology presented in Section 7.2).

[9]This is for lack of enough data (observe that we consider here a specific client) and the choice means a priori a pessimistic point of view.



(a) Upload bandwidth capabilities of the peers in the system.



(b) Evolution of the global available bandwidth.

Figure 8.7: User bandwidth distributions.

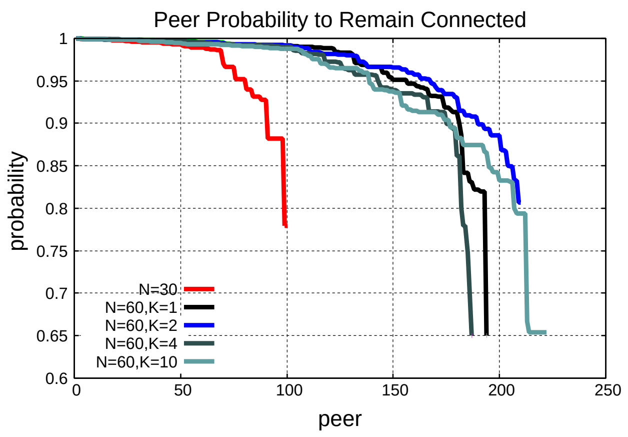


Figure 8.8: Statistical probability distribution to peers' remain connected in a given period ($\Pr(z_i = 1) = p_i$).

Table 8.1: Upload bandwidth consumed on server i, for the multi-source streaming technique defined in Table 7.12, with K-substreams.

i/K	1	2	3	4	5
1	492.7	239.9	109.2	46.5	35.7
2		400.1	188.4	88.9	54.7
3			342.4	171.3	91.0
4				332.4	160.6
5					295.7
total	492.7	640.0	640.0	639.0	637.5

i/K	6	7	8	9	10
1	13.3	11.9	11.1	1.2	0.7
2	24.3	17.0	13.2	2.4	1.3
3	44.8	26.7	17.5	4.8	2.5
4	83.6	45.0	26.0	9.5	5.0
5	157.6	80.5	43.0	18.8	9.8
6	299.4	150.0	76.9	37.2	19.4
7		286.8	144.8	73.9	38.3
8			280.4	146.7	75.7
9				291.5	149.9
10					296.7
total	622.9	617.8	612.8	585.9	599.2

Table 8.2: Perceived quality function, $Q_i = f(y^1_{s,i}, y^2_{s,i})$, when $K = 2$. Values come from the multi-source streaming technique defined in Table 7.12.

$y^1_{s,i}$	$y^2_{s,i}$	$f()$
0	0	0.00
0	1	3.64
1	0	2.28
1	1	10.00

Table 8.3: Perceived quality function, $Q_i = f(y^1_{s,i}, y^2_{s,i}, y^3_{s,i}, y^4_{s,i})$, when $K = 4$. Values come from the multi-source streaming technique defined in Table 7.12.

$y^1_{s,i}$	$y^2_{s,i}$	$y^3_{s,i}$	$y^4_{s,i}$	$f()$
0	0	0	0	0.00
0	0	0	1	2.31
0	0	1	0	1.40
0	0	1	1	4.82
0	1	0	0	1.04
0	1	0	1	3.45
0	1	1	0	2.09
0	1	1	1	7.70
1	0	0	0	0.90
1	0	0	1	2.88
1	0	1	0	1.75
1	0	1	1	6.17
1	1	0	0	1.28
1	1	0	1	4.36
1	1	1	0	2.66
1	1	1	1	10.00

Considering the four multi-source configurations, we generate four test cases from the small instance ($N = 30$). For each large instance ($N = 60$), we choose only one multi-source configuration. In what follows, we consider eight test cases, which will be denoted: $N30K1$, $N30K2$, $N30K4$, $N30K10$, $N60K1$, $N60K2$, $N60K4$, and $N60K10$.

8.4.2 Algorithms calibration

In order to test the implemented algorithms, we develop a program to simulate the behavior of the network during a certain period of time, divided into a discrete number of intervals. The algorithm to be executed at the end of each interval can be plugged into the simulator by implementing a specific function header. In our case, we implemented the Greedy heuristic, and the GRASP / GRASP+RNN based algorithms described in the previous sections[10]. This simulator was implemented in the C language, using the GNU C Compiler (`gcc`).

GRASP and GRASP+RNN are parametric in the size of the Restricted Candidate List (RCL), and GRASP+RNN also is parametric in the RNN use percentage (rnn). First, we need to define suitable values for these parameters. We do this by evaluating the performance of the algorithms with different parameter configurations, in the eight test cases.

The tests in this chapter were run on Intel Pentium 4 / AMD Athlon 64 machines with 1 GB of RAM, running under Ubuntu Linux 6.10.

8.4.2.1 Restricted Candidate List (RCL)

For each test case, we execute the GRASP algorithm with three different RCL sizes. In particular, we explore with the values $RCL \in \{5, 10, 25\}$.

[10]It is important to note that *any* assignment algorithm can be used provided its interface can be adapted to the one required by the algorithm.

Also, we present the results for the two enhancement criteria of a given assignment presented before, where "current" criteria selects the assignments which provide a better perceived quality for the network at the current time; and the "future" criteria gives higher priority to the assignments that will result in better perceived quality for the next iteration. The "future" criteria is also parametric in the number of trials (we use 3, 5 and 10 trails).

We measure the global perceived quality (PSQA) of the network after running the assignment algorithm on each interval, as well as the execution time of the algorithm. Table 8.4 shows the average perceived quality (our objective performance function) in each case. The average is computed over all the iterations. The results do not show significant differences between different RCL sizes. For this reason, we select $RCL = 5$, which leads to a moderate computational time.

Also, notice that the number of trails in the "future" criteria does not improve the solution for the considered values.

Table 8.4: GRASP performance for three different RCL sizes.

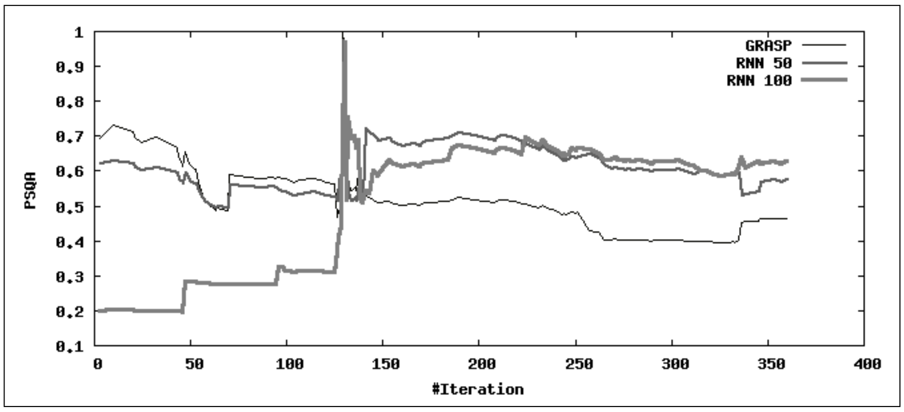
	GRASP											
	$RCL = 5$				$RCL = 10$				$RCL = 25$			
test	current	future			current	future			current	future		
cases		3	5	10		3	5	10		3	5	10
$N30K1$	4.44	4.02	4.18	4.15	4.42	4.37	4.19	4.39	3.61	4.30	4.33	4.05
$N30K2$	7.29	7.31	7.48	7.16	7.19	7.53	7.69	7.54	7.02	7.01	6.35	7.38
$N30K4$	9.51	8.97	8.86	9.57	9.72	9.17	9.29	9.68	9.79	9.35	9.51	9.44
$N30K10$	10.00	10.00	9.99	9.71	10.00	9.99	10.00	9.99	10.00	10.00	9.95	10.00
$N60K1$	3.64	3.31	3.04	3.13	3.82	3.45	3.62	3.64	3.30	3.55	3.51	3.75
$N60K2$	6.48	5.18	5.32	5.77	5.82	5.47	5.64	5.81	6.20	6.10	6.04	4.76
$N60K4$	5.24	5.09	6.61	5.42	6.03	5.25	6.14	6.06	5.88	5.55	6.03	5.71
$N60K10$	10.00	9.51	9.64	9.68	10.00	9.83	9.82	9.77	9.92	9.76	9.44	9.75
Average	7.08	6.67	6.89	6.82	7.12	6.88	7.05	7.11	6.96	6.95	6.90	6.85
Average	**6.8659**				**7.0415**				**6.9172**			

8.4.2.2 RNN use percentage (rnn) in the RNN-improved GRASP

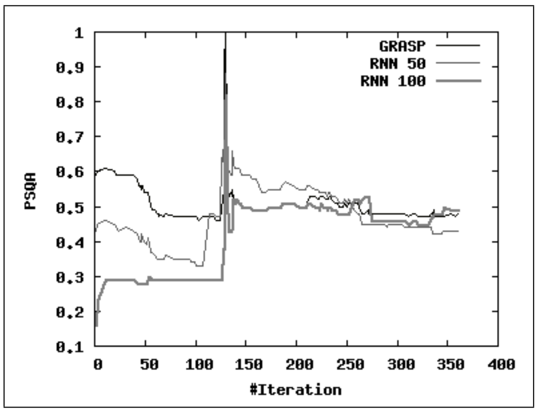
In GRASP+RNN we need to define the percentual use of the RNN improvement (where $rnn = 1$ means that all the assignments came from the improved selection, and $rnn = 0$ reduces the GRASP+RNN to the GRASP algorithm). We ran the simulation using several values for the rnn parameter, $rnn \in \{0\%, 25\%, 50\%, 75\%, 100\%\}$.

For instance, Figures 8.9(a) and 8.9(b) show the evolution of the PSQA during the simulation for each criteria in the $N60K4$ test case. As we can see, while the use of RNN initially leads to weaker results, this soon improves and then the perceived quality stabilizes at a higher level than the one obtained using GRASP-only assignments. We also observed, for the "future" improvement criteria, a noticeable decrease in the execution time of the algorithm when using RNN (see Figure 8.9(c)).

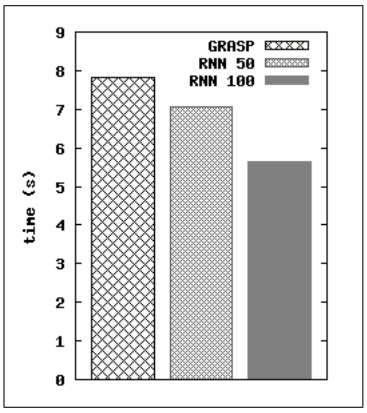
Table 8.5 summarizes the GRASP+RNN global performance for the different rnn probabilities values. The use of high rnn values ($rnn \geq 0.75$) degrades the algorithm performance. We select $rnn = 0.25$ for our GRASP+RNN algorithm.



(a) PSQA evolution ("current" criteria)



(b) PSQA evolution ("future" criteria)



(c) Execution time

Figure 8.9: Detail results for the $N60K4$ test case and the GRASP+RNN algorithm.

8.4.2.3 Selection of the enhancement criteria of a given assignment: "current" and "future" perceived quality

The three presented algorithms use a criteria to select the "best" assignment to be applied in the construction of the solution (see Section 8.3 for an introduction to the algorithms). The term "best" refers to how this assignment improves the current solution. Two criteria are evaluated in our work: "current" and "future" perceived quality. Because of computation time costs, we made our test of the "future" criteria with only three future possible scenarios (trails). Table 8.6 shows the performance of the enhancement criteria for the studied algorithms. There are no important differences between the two criteria; we then choose the "future" criteria, expecting better performance (but higher execution times also) when we increase the number of trials.

Other possible criteria can be considered in future work. For instance, how to increase the available bandwidth with the assignment. It can be calculated as the addition of the available bandwidth in the nodes of the disconnected tree. Intuitively, this favors solutions where more trees can be added.

Table 8.5: GRASP+RNN performance for four different rnn probabilities values.

	GRASP+RNN ($RCL = 5$)									
	$rnn = 0.00$		$rnn = 0.25$		$rnn = 0.50$		$rnn = 0.75$		$rnn = 1.00$	
test cases	current	future 3	current	future 3	current	future 3	current	future 3	current	future 3
$N30K1$	4.44	4.02	3.99	4.23	4.24	4.75	4.37	3.80	4.38	4.46
$N30K2$	7.29	7.31	6.07	7.46	7.81	7.29	7.58	6.01	5.32	5.02
$N30K4$	9.51	8.97	9.15	9.12	8.32	8.41	8.62	7.67	6.42	6.27
$N30K10$	10.00	10.00	9.75	9.80	9.58	9.31	9.49	9.63	7.22	7.61
$N60K1$	3.64	3.31	3.62	3.59	3.76	3.61	4.16	3.83	4.40	3.69
$N60K2$	6.48	5.18	6.55	4.98	5.29	6.12	5.75	4.61	5.89	5.02
$N60K4$	5.24	5.09	5.68	5.15	6.14	4.71	4.49	4.51	5.03	4.16
$N60K10$	10.00	9.51	9.82	9.84	9.73	9.76	8.62	9.19	4.86	6.45
Average	7.08	6.67	6.83	6.77	6.86	6.75	6.63	6.16	5.44	5.34
Average	**6.8747**		**6.7999**		**6.8016**		**6.3956**		**5.3875**	

Table 8.6: Performance of the enhancement criteria for the studied algorithms.

	Algorithm			
Criteria	Greedy	GRASP ($RCL = 5$)	GRASP+RNN ($RCL = 5, rnn = 0.25$)	**Average**
current	8.18	7.08	6.83	7.36
future (3 trails)	8.00	6.67	6.77	7.15

8.4.3 Comparing the performance of our algorithms: Greedy, GRASP and GRASP+RNN

For each test case, Table 8.7 shows the global execution time, and the average perceived quality considering all the iterations. The Greedy algorithm performs better than the GRASP-based algorithms, in terms of quality and execution time. The low performance of GRASP-based algorithms is related to the fact that they are tested with relative low parameter values (in particular the number of GRASP iterations). Increasing the number of iterations in GRASP increments linearly the computation time. With an average of 12 seconds (GRASP+RNN) in the considered (small size) test cases, it can be considered only to estimate, in laboratory, the performance of a constructive heuristic (like our Greedy proposal). In our GOL!P2P prototype (Chapter 10) we use the Greedy algorithm presented in this chapter.

Another observations can be extracted from the test results. For the two network sizes tested ($N = 30$ and $N = 60$), with an increment of the number of sub-streams (K), the global performance of the method increases too. This is because of the relaxation of the problem constraints with the K increment. It is possible to measure the number of nodes that receive the stream with a low quality in each iteration. Table 8.8 shows the average percentage of nodes that perceive a quality below 3 (poor). This is another measure of the improvement on the solution with an increment in the number of sub-streams.

Moreover, the available bandwidth in the network is better used when K increases. Table 8.9 shows this behavior. This can also be used in order to increase the quality of the stream or its redundancy level, etc.

Table 8.7: Average performance and Execution time of our three algorithms over all test cases.

test case	Algorithm Greedy PSQA	Greedy Execution time (sec)	GRASP ($RCL = 5$) PSQA	GRASP ($RCL = 5$) Execution time (sec)	GRASP+RNN ($RCL = 5, rnn = 0.25$) PSQA	GRASP+RNN ($RCL = 5, rnn = 0.25$) Execution time (sec)
$N30K1$	5.89	0.0000	4.02	0.1602	4.23	0.0303
$N30K2$	8.32	0.0028	7.31	0.5608	7.46	0.1984
$N30K4$	9.44	0.0138	8.97	3.0304	9.12	1.1901
$N30K10$	9.33	0.2099	10.00	22.6145	9.80	13.2700
$N60K1$	6.28	0.0028	3.31	0.1354	3.59	0.1019
$N60K2$	7.85	0.0221	5.18	4.2072	4.98	1.4518
$N60K4$	7.57	0.1450	5.09	20.1412	5.15	7.0579
$N60K10$	9.34	0.5635	9.51	101.9945	9.84	74.0193
Average	8.00	0.1200	6.67	19.1055	6.77	12.1650

Table 8.8: Percentage of nodes that perceive with low quality.

test case	Algorithm Greedy %nodes	GRASP ($RCL = 5$) %nodes	GRASP+RNN ($RCL = 5, rnn = 0.25$) %nodes
$N30K1$	0.44	0.65	0.62
$N30K2$	0.22	0.31	0.32
$N30K4$	0.08	0.14	0.12
$N30K10$	0.00	0.00	0.00
$N60K1$	0.40	0.72	0.69
$N60K2$	0.22	0.62	0.65
$N60K4$	0.33	0.67	0.66
$N60K10$	0.00	0.00	0.00

8.4.4 Extending our Video Delivery Reference Service with a P2PTV system

We compare the performance of our P2P assignment metaheuristic with a traditional CDN-based implementation.

Two important variables have to be considered in the analysis: the global perceived quality of the network, and the total bandwidth consumed (at the broadcaster and at the peers). We use the $N60K10$ case, coming from our live video delivery service (see Section 2.5 for details). This service has 20.000 access of different users per month, and an average of 50 concurrent users per live-TV channel.

In the CDN architecture case, the broadcaster is a set of servers in the ISP datacenter, where the bandwidth consumption is the most expensive component cost of the service. The broadcaster absorbs all the load of the clients, with a stream of 512 Kbps; this means at least 50 Mbps of bandwidth in peak use. The broadcaster of the CDN does not fail[11]. If we consider the packet losses in the network negligible, we can assume that the CDN network has a perfect global quality on the average (i.e $Q = 10.00$). We simulate a P2P architecture with the same

[11]Actually the servers in the CDN fails, but the load is absorbed immediately by another active server, and the users do not perceive the failure.

Table 8.9: Percental global consumed bandwidth.

	Algorithm		
	Greedy	GRASP	GRASP+RNN
test		$(RCL = 5)$	$(RCL = 5, rnn = 0.25)$
case	$\%BW$	$\%BW$	$\%BW$
$N30K1$	0.622	0.612	0.629
$N30K2$	0.837	0.801	0.723
$N30K4$	0.997	0.997	0.997
$N30K10$	0.997	0.997	0.931
$N60K1$	0.678	0.654	0.670
$N60K2$	0.894	0.447	0.425
$N60K4$	0.997	0.997	0.997
$N60K10$	0.997	0.997	0.925

clients' behavior (connections/disconnections) than the real CDN. We split the original stream in four redundant sub-streams, with a total bandwidth consumption of 614 Kbps. Our results show that the broadcaster absorbs only 5 Mbps, and the peers the rest of the load (in average 0.6 Mbps per peer). The quality is not considerably degraded, with a worst case of $Q = 9.34$ on the average.

The numerical results obtained show the interest of employing a live-video P2P distribution system following a multi-source procedure.

8.5 Summary and Conclusions

In this chapter we present a preliminary design of a centralized tree-based overlay topology, for our P2PTV system.

The overlay is built choosing which peer will serve which other one. The assignment has to be done in a way to diminish the impact of peers disconnection on the quality. We model the overlay building as a mathematical programming problem, where the optimization objective is to maximize the global expected QoE of the network, calculated using the PSQA methodology. We present three algorithms for solving this problem, applied to cases study based on real life data.

This is just a first study that shows the possibilities of our approach. Future work is needed to improve the assignment algorithm. For instance, considering other enhancement criteria (like the available bandwidth), or using the presented greedy algorithm inside the GRASP. Finally, larger test cases have to be analyzed.

Part IV

CHALLENGES IN THE REALITY

Chapter 9

Measure and Monitor the Quality of Real Video Services.

Different network architectures are developed to deliver video transparently to end users. In our work we grouped these architectures under the name Video Delivery Networks (VDNs) (VDNs are presented in Section 2.2).

This chapter presents a full Video Delivery Network (VDN) monitoring suite. Our monitoring tool offers a new view of a VDN, a view based on the quality perceived by final users. We measure, in real time and automatically, the perceived quality at the client side by means of the PSQA methodology (see Chapters 4 and 5).

The developed monitoring suite is a completely free-software application, based on well-known technologies such as the Simple Network Management Protocol (SNMP) or the Round Robin Databases (RRD Tool), which can be executed in various operating systems. It is an extensible measurement framework that can be applied to most of the studied VDNs.

In this chapter we explain the tool implementation and we present some of the first experimental measurements performed with it. This work has been partially presented at the 7th IEEE International Workshop on IP Operations and Management (IPOM'07) [331] (listed at the Introduction as [ipom07audit]). The implementation of the suite has been completed with the contribution of J.L. Fernandez, D. De Vera and A. Chadarevian, who completed their engineering project at the Universidad de la República, Uruguay, in this subject.

9.1 Introduction

Video delivery systems may have many different architectures, each one with different points of failure and key components. For each architecture, monitoring and measurement are necessary in order to get a satisfactory Quality of Experience level with a minimal cost. To identify factors playing an important role on QoE, some specific quantitative aspects must be considered [87] (see also our discussion about this topic in Subsection 2.2.2). For video delivery services, the most important one is the perceptual video quality measure. Accurate video-quality measurement and monitoring is today an important requirement of industry. Service providers need to monitor the performance of various network layers and service layer elements, including those in the video head-end equipment (such as encoders and streaming servers) as well as at the home network (such as the home gateway and STB). Some solutions are being delivered by niche vendors with a specific focus in this area [93, 319], by large telecommunication infrastructure providers as part of an end-to-end VDN solution [9, 59, 309], or by a fully in-house development.

Monitoring tools can be classified into two different categories: active and passive. An active monitoring tool sends traffic through the network for performing its measurements. A passive one uses devices to watch the traffic as it passes through the measuring points.

This chapter describes a platform architecture belonging to the class of the active monitoring tools, that use probe nodes distributed in the network, with a centralized data collector using Simple Network Management Protocol (SNMP) [155]. The traditional way to monitor video quality [93, 319] in a VDN is a manual process, where experts observe the quality continuously in some displays located logically in different stages of the network (typically at the output of the encoding process and in a simulated client situated in the head-end of the network, where the experts are located). In a IP network with losses and congestion, it is necessary to be in the edge of the network to measure accurately the quality, but this is not possible because the perceived quality measure is actually a manual process. To avoid that, the usual approach to assess the performance of a VDN is to use a well chosen metric, that we know plays an important role in quality, such as the loss rate of packets, or of frames, and to analyze it in the system of interest. In this work we instead address the problem of directly measuring *perceived* quality by means of the Pseudo Subjective Quality Assessment (PSQA) technology

The chapter is organized as follows. Section 9.2 introduces the measurement methodology used in our platform. In Section 9.3, the architecture of the platform is described. In Section 9.4, we report on some experimental results allowing to illustrate the platform use. The main contributions of this work are then summarized in Section 9.5.

9.2 Methodological Considerations

Before describing our methodology, recall from Section 2.1, that in MPEG-2/4, the transmission units are the *frames*, which are of three main types: the Intra frames (I), the Predicted frames (P) and the Bidirectional or Interpolated frames (B). The frames are grouped in sets called groups of pictures (GOP) in MPEG-2 and groups of video object planes (GVOP) in MPEG-4. MPEG-2 and MPEG-4 also share a common concept called *user data*, which corresponds to byte sequences pertaining to an user application that can be inserted inside a stream. This can be done in many places, at the different abstraction levels defined in the specifications. The GOP header is the lowest one (this means that between two consecutive I frames we will find at most one piece of user data). As we will see in the following sections, the user data concept will be a fundamental piece in our audit platform design.

Some background information is needed to understand the methodology used in the measurements of our platform. This section briefly recalls the main needed concepts.

9.2.1 Simple Network Management Protocol

Simple Network Management Protocol (SNMP) [155] is a widely diffused protocol used to manage different resources and applications within a network. Network management systems contain two primary elements: the manager itself and the agents. The manager is the console through which the network administrator performs network management functions. Agents are the entities that interface to the actual device or application being managed. In the agent there should be one or more objects (parameters) to monitor. These objects should be defined on a management information base (MIB). The manager can read and/or modify the objects defined inside the MIB, carrying out this way the administration of each one of these objects.

SNMP has been used in previous situations for network monitoring by means of probes [152][1], with an integrated logging module [151], a scenario very similar to ours. Our application uses the SNMPv3 [155, 157] version and our designed MIB module is compliant with the SMIv2 [147, 149, 150].

9.2.2 Quality Measurements.

We want to measure the video quality in real–time, in particular the quality perceived by end users. As we know, traditional video quality assessments do not match well with this requirement. *Subjective tests* are very time-consuming and expensive in manpower, which makes them hard to repeat often. By definition, they cannot be a part of an automatic process. The other approach, called *objective metrics*, often provide assessments that do not correlate well with human perception, and thus their use as a replacement of subjective tests is limited. Furthermore, with a few exceptions, objective metrics needs the received and the original video sequences to compute the metric, so, it is not possible to use them in a real–time monitoring environment. A survey of traditional video quality assessments is presented in Chapter 3.

Here, we exploit the advantages of the Pseudo Subjective Quality Assessment (PSQA) hybrid approach explained in Chapter 4. PSQA gives us a mapping function from some quality-affecting parameters into the perceived quality. It is very easy to use in a monitoring suite: we need to measure the values of the chosen parameters at time t, and then to put them into the PSQA function $Q()$ to obtain an estimation of the *instantaneous* perceived quality at t.

Depending on the Video Delivery Network, our prototype can be adapted to different parameters, including network and application level parameters. In our tests, we use the PSQA *simple function* presented on Section 5.2.3.

9.2.3 An Extensible Measurement Framework.

We distinguish two main components within our measuring system: a set of probes (actually, video players) and a family of centralized monitoring servers (usually the service itself). The video players are an improved version of the VideoLan client software (VLC) [334]. Each VLC client performs the measuring tasks, and makes the measures available to the servers using the Simple Network Management Protocol (SNMP) [155]. A collector server polls the clients to obtain the SNMP MIB values of the measured parameters.

[1]Probes are remote network monitoring devices useful for the network management.

The parameters measured in the extended VLC client come from two different data sources: dynamically calculated information (e.g., video bitrate, I-Frame mean size, P-Frame mean size, B-Frame mean size, codecs detection and so on) and information included within the stream itself. As mentioned before, the *user data* defined in MPEG-2/4 allows to insert application's information inside the stream (using any server that complies with the MPEG standard). The measurement framework defines rules about how to tag a stream (for instance, what information should be inserted in the user data, how it should be formatted, and where it should be placed). The inserted information is captured and used by the extended VLC client in the parameters calculation. This flexibility allows our tool to evolve smoothly with time and technologies, adapting to changes in the network, the applications, or the users' behavior, by simply modifying the input parameters used to build the quality evaluation metric.

Table 9.1: Measurement framework parameters.

Frames Related Information
Losses per frame type
Frames expected to receive per frame type
Frames received per frame type
Mean size per frame type
Frames bitrate

Streams Related Information
Streaming server IP and port
Transport protocol
Container format
Video and audio codecs

Clients Related Information
Client active streams quantity
Time since the beginning of a stream execution

Table 9.1 shows the current parameters measured within our framework. The *user data* contains the number of I, P and B frames from the beginning of the stream. We send it at the beginning of each GOP. The extended VLC players count the frames received per class and, when the new user data arrives, compare them to their own counters. This way, frame losses are detected with high precision. In Figure 9.1 we show an example of the application of this algorithm. Assume that on a stream execution a B-Frame is lost or delayed. The B-Frame quantity counted in the VLC client would be seventeen. When comparing this with the user data information, there will be a difference of one frame. So until the next user data arrival, the client will report the lost of one B-Frame.

For the frame losses calculation we use the *user data* format defined in Table 9.2. The `user_data_start_code` field is defined in both MPEG-2 and MPEG-4 video standard

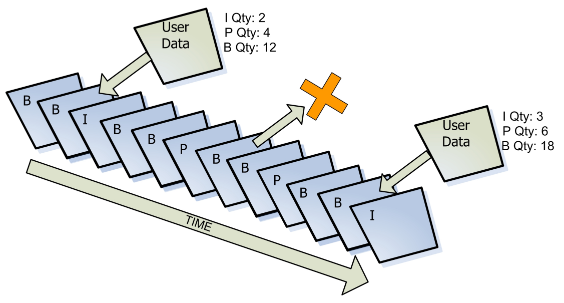


Figure 9.1: Frame losses calculation - VLC players count the frames received and when receiving the user data information, both are compared.

Table 9.2: User Data Format.

Bytes	Field	Value
0 - 4	user_data_start_code	0x000001B2
5 - 6	application_id	0xDDAB
7 - 10	iframes_count	-
11 - 14	pframes_count	-
15 - 18	bframes_count	-

specifications. The `application_id` field is used in order to identify our application (this is because other applications could also insert their own user data inside the stream). The `frames_count, pframes_count, bframes_count` counters refer to the I-Frames, P-Frames and B-Frames respectively. The counting process starts at the beginning of the Elementary Stream (ES) and goes until the next user data.

An important fact to consider is that the overhead added in the stream by the user data insertion is completely negligible (in our tests: 19 bytes in 22700 bytes in MPEG-2 and 19 bytes in 281700 bytes in MPEG-4).

9.3 The Audit Platform

In this section we explain the platform architecture and its components.

Inside the VDN we want to monitor, there are five basic components (Figure 9.2): the streaming server, the probes, the data collector server, the PSQA Tool and the Webstat application. The streaming server streams the video's content over the VDN. Probes are VLC players strategically located in the network, taking specific measures and sending reports using SNMP. The data collector server polls each probe of the VDN (using SNMP) in order to gather the probes' reports. The PSQA Tool is where the perceptual quality value is computed. Finally, Webstat provides a web interface for the probes' reports presentation.

Streaming Server. The function of the streaming server is to provide multimedia content to the network. This content can be coded using different video specifications (MPEG-2, MPEG-4...), audio specifications (MP3, FLAC...), container formats (MPEG-TS, MPEG-PS, OGM...)

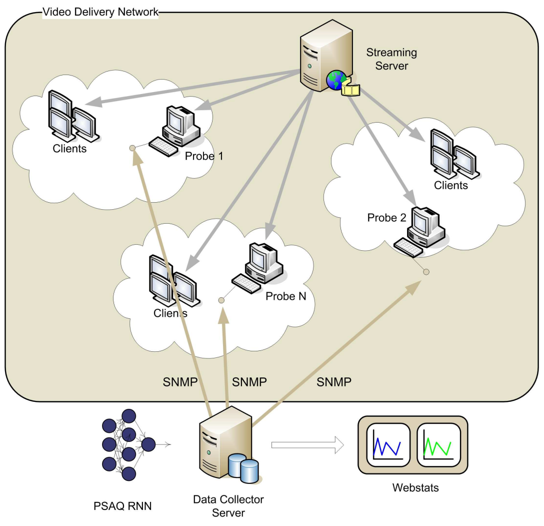


Figure 9.2: Global architecture - The streaming server, the probes, the data collector server, the PSQA Learning Tool and the Webstat.

and it can be streamed over different transport protocols (HTTP, UDP, RTP, MMS...). As we mentioned before, the measurement framework requires the user data insertion in the streamed content over the VDN. For this purpose, we have two different alternatives: inserting the user data in the streaming server on the fly, thus using a specific server, or inserting them in a post-encoding process, thus without any need for a specific streaming server. In our test scenario we chose the first option, by means of a modified VLC server. In a more realistic situation it may be not possible to use our own VLC servers; in that case, a preprocessed video (with user data) is needed to use with a generic stream server. Obviously this is a restriction of our method, but with our approach we win independence of container formats and transport protocols, which present high variability between different VDNs.

Probes (VLC players). Probes are VLC players modified in order to measure some specific information. They are strategically located inside the VDN. Basically, a probe is a VLC player with supplementary modules for coordination and data calculation, a SNMP module and a Logs module. See Figure 9.3 for a inside picture of a probe. Probes allow to capture and parse the user-data, to measure and calculate generic information (like the start and stop of the stream), to offer realtime reports through SNMP and to manage a set of rotating log files with all the relevant probe information.

Data Collector Server. The data collector server is in charge of gathering the probes's information. This application polls each one of the probes in the VDN (with some periodicity) and saves the data on a Round Robin Database (RRD). In this case, the data collector server polls the probes every 10 seconds and one RRD is created per active stream.

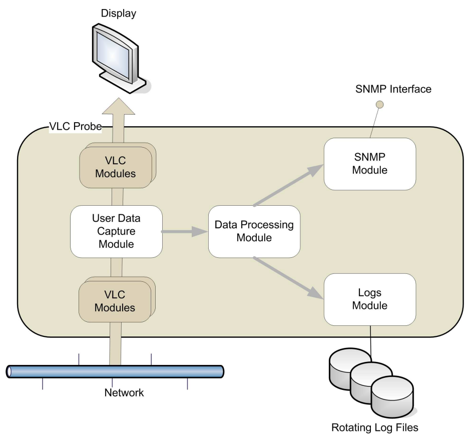


Figure 9.3: VLC Probe - A probe is a VLC player with the aggregate of the coordination and data calculation module, SNMP module and Logs module.

PSQA Tool. The PSQA Tool receives the quality–affecting parameters from the probes and computes the perceived quality of each one. In operation, the use of PSQA is very simple: the probes send statistical information about the quality–affecting parameters in a short period to the data collector, to be evaluated by the PSQA Tool. The period size can be arbitrarily defined for the specific application, and it is usually recommended to make it quite short, in order to use PSQA as an *instantaneous* quality value.

Webstat. Webstat is an application designed to present the data gathered by the data collector server to administrators and managers of the VDN. It offers reports at different levels and types of views. It is possible to generate reports focused on a particular stream or on a specific client (where there could be more than one active stream), or perhaps on the entire network (where there could be more than one client). Finally, it is possible to display the results at the frame level, possibly per frame type (I, P, B), or at the PSQA level.

Implementation. As mentioned before, both the probes and the streaming server are based on a VLC Player. The `libavcodec (ffmpeg)` library was used in order to work with MPEG-2 and MPEG-4 video specifications. As container format we worked with MPEG-TS using the functionalities provided by `libdvbps`. We streamed over HTTP and UDP using VLC internal modules. In order to report the probes's measures through SNMP we used the Net-SNMP library. We used the RRDtool to generate the statistical graphs. The data collector server was written in PHP and we used the following PHP extensions: `php4-snmp` and `php4-rrdtool`. Webstat is also written in PHP; it uses MySQL as relational database and it runs over Apache. All libraries and applications mentioned above are free-software and they can be executed in Microsoft Windows, Unix, Linux, etc.

9.4 Evaluating the platform in the Lab

To illustrate the platform possibilities, we tested it in some simple scenarios. We show here some examples of the obtained results. The scenarios concern the basic case of a VDN over the Internet.

Testing Scenarios. We simulated a VDN and we used our tool to monitor it. We used three streaming servers located at the same network, ten clients located at another network, one computer carrying out the routing function between both networks and a second computer where the data collector server and the Webstat application are run.

We considered three different scenarios: one of them without losses (NoLS), another one with server failures (SLS) and a last one with congestion (packet losses) in the network (NLS). See Figure 9.4 for details. The first scenario consisted of a streaming server who sends traffic

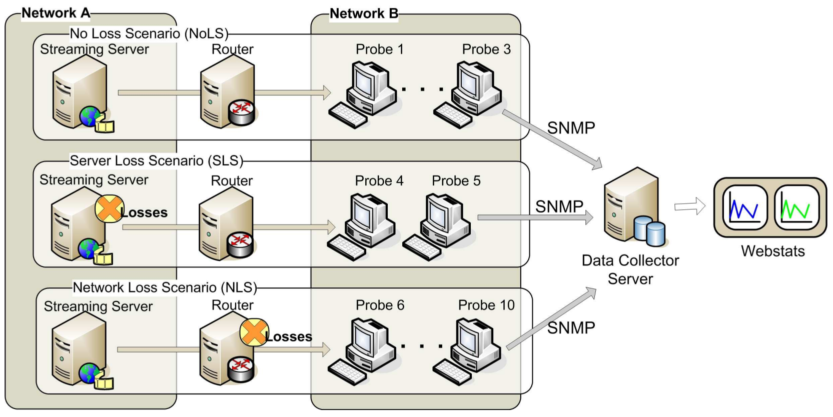


Figure 9.4: Testing Scenarios.

to three clients over a network without losses. In the second scenario, some video frames are eliminated from the stream by using a specific VLC server. This scenario was composed of one streaming server and two clients. Finally, in the third configuration we eliminated IP packets in the routing computer (congestion losses simulation). The losses were generated in an uniform way using the `netem` Linux module. This scenario was composed of one streaming server and five clients. Table 9.3 presents information about each of the streams used in the evaluations: the transport protocol, the video specification and the video bitrate of the streaming, the scenario where the test case was executed and, if it matters, the loss rate. When HTTP is the transport protocol, a maximum bandwidth is set in the network. This is in order to make the IP packet losses have a direct impact on the quality of the stream; otherwise the lost packets would be retransmitted and will only be affecting the bandwidth consumed in the network. In the case of sequences coded at 512 Kbps, a bandwidth of 717 Kbps is set in the network; for sequences coded at 1024 Kbps, the maximum bandwidth is 1280 Kbps.

Obtained Results. As mentioned in Section 9.3 and as shown in Figure 9.5, the developed application lets us analyze the results at different levels and providing different views.

Table 9.3: Scenarios Configuration. Each test case correspond to a client receiving a stream. Each scenario correspond to a failure situation: without losses (NoLS), with server failures (SLS) and with packet losses in the network (NLS).

Test Case	Protocol	Video Specification	Video Bitrate (Kbps)	Scenario	Loss Rate
1	UDP	MPEG-2	1024	NoLS	-
2	HTTP	MPEG-2	1024	NoLS	-
3	HTTP	MPEG-4	512	NoLS	-
4	HTTP	MPEG-4	512	SLS	0.50
5	HTTP	MPEG-4	1024	SLS	0.30
6	UDP	MPEG-2	512	NLS	0.30
7	UDP	MPEG-2	1024	NLS	0.30
8	UDP	MPEG-4	1024	NLS	0.30
9	HTTP	MPEG-2	1024	NLS	0.02
10	HTTP	MPEG-4	1024	NLS	0.04

Figure 9.5(a) shows a global view of the relative error measured during the evaluation at the frame level. This information can also be provided on a per-frame type basis (I-Frame, P-Frame, B-Frame).

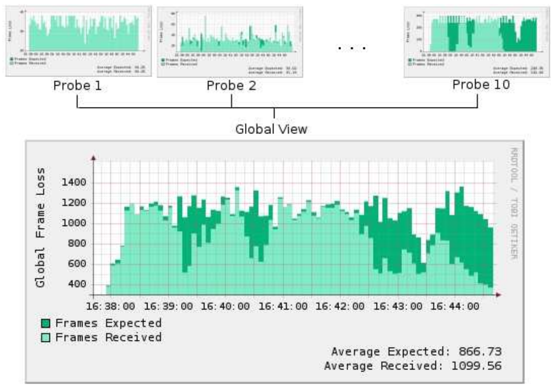
Figure 9.5(b) shows the evolution of global quality (PSQA) with time, normalized to numbers in the interval $[0, 1]$. We use the PSQA *simple function* presented on Section 5.2.3, that mapping LR and $MLBS$ into perceived quality. Table 9.4 presents some values obtained when

Table 9.4: Obtained Results.

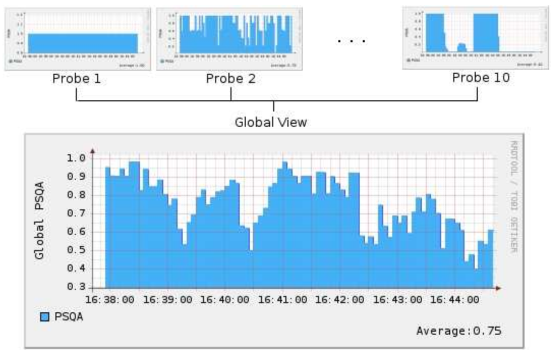
Test Case	Protocol	Specified Loss Rate	Measured Frame Loss Rate	Mean PSQA
1	UDP	-	-	10.0
2	HTTP	-	-	10.0
3	HTTP	-	-	10.0
4	HTTP	0.50 (Frame level)	0.47	4.2
5	HTTP	0.30 (Frame level)	0.29	6.1
6	UDP	0.30 (IP level)	0.09	7.9
7	UDP	0.30 (IP level)	0.19	6.6
8	UDP	0.30 (IP level)	0.33	3.8
9	HTTP	0.02 (IP level)	0.08	7.9
10	HTTP	0.04 (IP level)	0.07	8.9

executing the simulations. As it can be observed, there is no visible relationship between IP packet losses and frame losses. This is because this relationship depends on various factors: the video specification, the video bitrate, and the specific player software that processes the stream. However, there is a clear relationship between the loss rate and the PSQA value: the higher the loss rate the lower the quality perceived by the users.

Last, Figure 9.6(a) shows the frame losses measured during a test case where server failures occur and Figure 9.6(b) shows the frame losses measured during a test case with network losses.

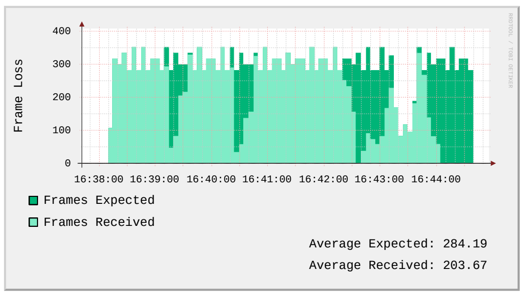


(a) Global frame losses.

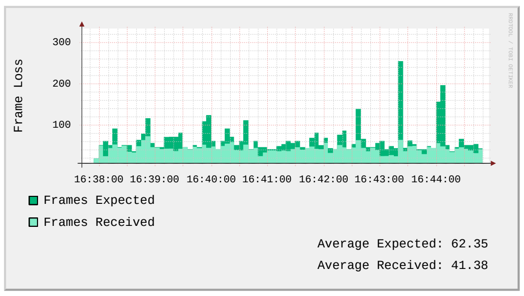


(b) Global perceptual quality (QoE).

Figure 9.5: Typical global view of the entire Video Delivery Network.



(a) Server failure example (5th test case).



(b) Network loss example (8th test case).

Figure 9.6: Typical stream view of the Video Delivery Network.

9.5 Summary and Conclusions

This chapter presents an effective monitoring and measuring tool that can be used by VDN managers and administrators to assess the streaming quality inside the network. With this tool it is possible to automatically monitor different sets of parameters of the streams, in real-time if necessary, including the perceived quality as seen by the final users, thanks to our improved version of the recently proposed PSQA technology. PSQA provides an accurate approximation of the QoE (Quality of Experience) and, to the best of our knowledge, our tool is the only one that is able to evaluate perceived quality continuously at arbitrarily chosen points in the network.

Another important feature of the presented suite is that it is not dependent on the considered VDN. It was designed as a generic implementation, in order to be able to use it with different VDN architectures. Moreover, it can be associated with most common management systems since it is built over the SNMP standard. Another feature is that the induced overhead is negligible. Finally, the tool is a free-software application that can be executed on several operating systems.

Chapter 10

GOL!P2P: A P2P Prototype for Robust Delivery High Quality Live Video

In this chapter we present our GOL!P2P prototype for live video streaming using a P2P network. Section 10.1 explains the core delivery system, i.e. the multi-source streaming implementation. This section has been partially presented at the 7th IEEE International Workshop on IP Operations and Management (IPOM'07) [285] (listed at the Introduction as [ipom07msource]). The implementation of the tool was made possible by the work of L. Stábile, who completed his engineering project at the Universidad de la República, Uruguay, in this subject. Section 10.2 presents the global architecture and main components of our prototype. In Section 10.3 we report on some integrated experimental results which illustrate the system performance in a real situation. The global prototype and the tests have been submitted [284] (listed at the Introduction as [submitted08a]). The chapter finalizes with general conclusions (Section 10.4).

10.1 Generic Multi-source Streaming Implementation

We will focus on the case of a P2P architecture where the video flow is decomposed into pieces and sent through the network. Each node receives the flow from different sources and builds the original stream before it is played. At the same time, it will, in general, send each of the substreams to a different client, acting then as a server. At the beginning, the (single) initial stream is split into several substreams, according to some specific scheme. With these goals, we need the two basic following services: (i) a flexible "splitter" where the original stream is decomposed into different substreams, and (ii) a module capable of reconstructing the original stream from the set of substreams, or a degraded version of it if some of the substreams had losses, or is simply missing. Associated with these services we want to design a very flexible transport scheme, allowing to transport the original information plus some redundancy, with high control on which server sends which part of the flow and of the redundant information (see below).

Let us now briefly describe these different components of the project. The splitter must be able to obey a general scheme where we decide how many substreams are to be used, and which of the frames are to be sent through which of the substreams. This generality allows us to send, for instance, most of the frames through the highest performing nodes in the network, or to balance the load among the different components, or to adopt a scheme where the type of frame is used to take the routing decision. It must also be possible to send an extra fraction r of the original stream, or of a specific population of the frames (for instance, the I frames) again according to any pre-specified scheme. If $r = 0$ there is no redundancy at all. If $r = 0.2$ we send 20% of supplementary redundant data, and $r = 1$ means that the original stream is actually sent twice to the receiver(s). We do not consider $r > 1$ because, in a real system, this would mean too much bandwidth consumption. At the client side, we must be able to reconstruct the original stream if we receive all the frames, but also if only a part of the stream arrives; this typically happens when some of the servers fails (that is, they disconnect from the P2P network).

We extend the VLC software to send and to receive video streaming from multiple sources. The multi-source streaming technique implemented is the most general one; for instance, the improved optimal version explained in Section 7.2 can be configured. In this section, we describe the method together with the design decisions and the operation characteristics of the tool. We illustrate the behavior of the tool by means of some tests.

10.1.1 Implementation Design

There are K servers and each server is identified by an index between 1 and K. The implementation consists of three VLC modules, one at the server side and two at the client side. They are called `msource-server`, `msource-bridge` and `msource-client` (see Figure 10.1).

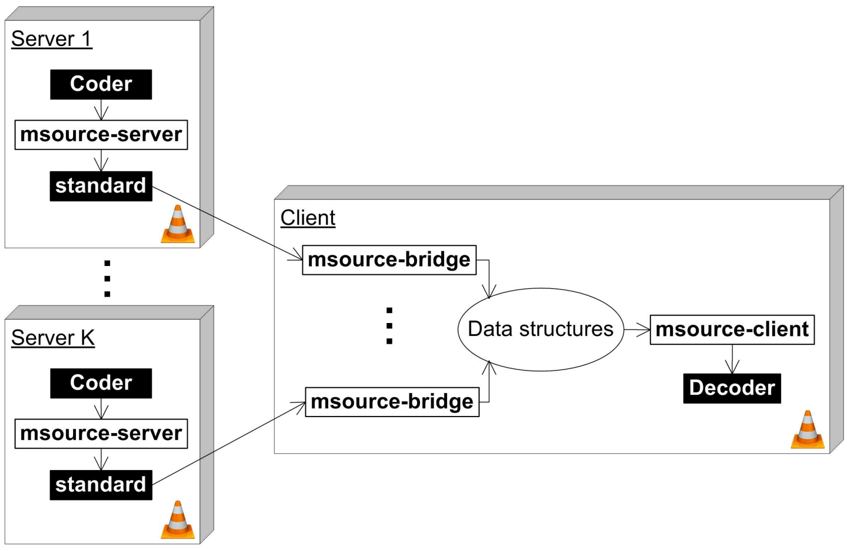


Figure 10.1: Architecture multiple-source in VLC

The module `msource-server`, located at the server, decides which frames are to be sent to the client. It builds a substream for audio and another one for video. The basic constraint to satisfy is, of course, that each frame in the original stream must be sent at least once by one of the servers. Once the frames selected by `msource-server`, the module `standard` of VLC sends them to the client.

At the client's side, there are K modules `msource-bridge`, one per server and each with its own buffer. The kth one receives the frames sent by server k. The last module, `msource-client`, receives the flows sent by the K modules `msource-bridge` and reconstructs the stream. This task consists of ordering the frames according to their decoding time (called DTS). The output of a module `msource-client` can be stored on the local disk, sent somewhere through the network (both tasks are performed by the module `standard`, which is a part of the VLC package) or played by the client.

The ideal case is when all the servers start their transmissions simultaneously and keep synchronized, but this never happens in practice. It means that the system must handle the possible shift between the servers. Moreover, since we don't assume that the servers are identical, we implement a strategy allowing to control the bandwidth used by each of them. Observe that these modules work at the frame level in the flow; it implies that our multi-source implementation is general and can be used with several video codecs, container formats and transport protocols. No extra signaling is used in the method.

10.1.1.1 Basic Server Algorithm

The strategy used by the servers allows to control the bandwidth used by each of them, for instance, to be able to send more information through the most reliable one, or through the one having the best performance. This control is implemented by acting on the percentage of frames of each type that the server must send. We must decide which fraction $p_k^{(c)}$ of the class c frames, $c \in \{\text{I, P, B, A}\}$ [1] is going to be sent through server k. We must then have

$$\sum_{k=1}^{K} p_k^{(c)} = 1, \quad c \in \{\text{I, P, B, A}\}. \tag{10.1}$$

The sending algorithm works as follows. All the servers run the same pseudo-random number generator and build a sequence of pseudo-random real numbers uniformly distributed on $[0, 1]$. Not only they run the same generator but they use the same seed. In this way, they obtain the same sequence of real numbers $(u_1, u_2, \cdots)$ behaving as a realization of the corresponding sequence of i.i.d. random variables uniformly distributed on $[0, 1]$.

Now, look at $(p_k^{(c)})_{k=1,\cdots,K}$, for fixed c, as a probability distribution, and call $X^{(c)}$ the corresponding random variable. It can be sampled from an uniform random variable U using the classical algorithm given by

$$X^{(c)} = k \Longleftrightarrow P_{k-1}^{(c)} \le U < P_k^{(c)}, \tag{10.2}$$

where $P_j^{(c)} = p_1^{(c)} + \cdots + p_j^{(c)}$, $j = 1, \cdots, K$, and $P_0^{(c)} = 0$.

[1] In MPEG, basically there are three classes of video frames (I, P and B), and one class of audio frames (A).

Now, let f_n be the nth frame of the stream, $n \geq 1$, and let $c_n \in \{\text{I, P, B, A}\}$ be the class of f_n. Any of the K servers will then run the same algorithm. They all sample the random variable $X^{(c)}$, and obtain the same value s_n, with $1 \leq s_n \leq K$. That is, they all look at the type of f_n, and use (10.2). Server s_n sends f_n and the remaining servers don't. This construction guarantees that one and only one server sends each frame, and since s_n behaves as a realization of random variable $X^{(c)}$, after sending many frames the fraction of type c frames sent by server k will be close to $p_k^{(c)}$. Observe that this method allows to control the bandwidth of each server in a scalable way (there is no limit on the number of servers nor on the distribution of the load among them).

Controlling the Redundancy Level. With our method we can send some redundancy to the client. This, together with our splitting procedure, adds robustness to the system, in order to face the problem of possible server failures (recall that we call failure any event causing the server to stop sending frames, for instance, because it simply left the network). As we will see in next subsection, redundancy also allows us to design a simple solution to the problem of synchronizing. We describe here the way we control the redundancy level in the system.

We allow the system to send redundant frames up to sending again the whole stream, and we provide a precise mechanism to control with high precision the redundancy level and the distribution of the supplementary load among the K servers. For this purpose, we implement a system where a given frame is sent either once or twice, and in the latter case, by two different servers. Our method allows to control the redundancy level per class, for instance, for each class of frames (I, P, B). We denote by $r^{(c)}$ the fraction of supplementary class-c frames that will be sent to the client (by the set of servers). So, each server k must decide if it sends frame f_n as the "original" frame, as a copy for redundancy, or not at all. The procedure described before allows to choose the server that sends the frame as the original one. For the redundancy, the implementation is also probabilistic. We proceed as follows. Together with the stream of pseudo-random numbers used to make the first assignment we described before, the set of servers build a second sequence $(v_1, v_2, \cdots)$ with the same characteristics (the same for all servers, the same seed). Suppose that frame f_n is of class c and that it is assigned to some other server j, $j \neq k$. Then, using the second sequence $(v_1, v_2, \cdots)$, server k samples a second random variable with values in the set $\{1, 2, \cdots, K\}$ to decide at the same time if a redundant copy of the frame is to be send and if it must send it. Let us call $Y_j^{(c)}$ this random variable. Its distribution is the following: $\Pr(Y_j^{(c)} = j) = 1 - r^{(c)}$ and if $m \neq j$,

$$\Pr(Y_j^{(c)} = m) = \frac{p_m^{(c)}}{\sum_{h:\, h \neq m} p_h^{(c)}} r^{(c)} = \frac{p_m^{(c)}}{1 - p_j^{(c)}} r^{(c)}.$$

If $Y_j^{(c)} = j$, no redundancy is sent. If $Y_j^{(c)} = m$, $m \neq j$, server m is the only one to send the frame as a redundant one.

10.1.1.2 Client Behavior

The client must reconstruct the stream using the flows received from the different servers. It will work even if some of the servers are missing (failed). Each server sends its streams marked with a time stamp indicating its playtime with respect to a specific reference (in VLC, 1/1/1970). Assuming all the servers synchronized, the algorithm is simple: it consists of selecting as the next frame the one having the smallest value of playtime. The problem is the possible shift between the servers (whose locations are different in the network). This can be due to the conditions encountered by the packets in their travelling through the network, or by the processing of the frames at the servers themselves. The key idea for knowing this shift in the transmission time of the servers consists of using the redundant frames.

First, let us look at the identification of the redundant frames. Each frame is sent with a time stamp corresponding to its playtime at the client side (computed by VLC). When receiving it, a Message-Digest algorithm 5 (MD5) [145] is computed and a dictionary is maintained to see if an arriving frame has already been received (necessarily by a different server). If the frame arrives for the first time, we store its MD5 together with its time stamp. Assume now that the same frame arrived from two different servers i and j, and call τ_i and τ_j the corresponding time stamps. The difference between these values is the (current) shift between the two servers. We denote it by Δ_{ij}, that is, $\Delta_{ij} = \tau_i - \tau_j$. Let us denote by Δ the $K \times K$ matrix (Δ_{ij}) where we define $\Delta_{ii} = 0$ for all server i. Observe that $\Delta_{ji} = -\Delta_{ij}$. Following these observations, we maintain such a matrix, initialized to 0, and we modify it each time a new redundant frame is detected (actually we do it less often, but this is a detail here). Observe that rows i and j in the matrix, $i \neq j$, are obtained by adding a constant element per element (or, equivalently, see that $\Delta_{ij} = \Delta_{ik} + \Delta_{kj}$). This constant is precisely the delay between the corresponding servers. The same happens with the columns. All this in other words: if we receive $K - 1$ redundant pairs corresponding to $K - 1$ different pairs of servers, we can build the whole matrix.

Each time we update matrix Δ, we can compute the index d of the most delayed server (if any), by choosing any row[2] in Δ and by computing its smallest element. Then, using for instance row 1, $d = \text{argmin}\{j : \Delta_{1j}\}$. When the client now looks for the next frame to choose, it scans the K buffers corresponding to the K servers. Let us denote by τ_k the time stamp of the first frame in the kth buffer (assume for the moment that no buffer is empty). Then, we first make the correcting operation $\tau'_k = \tau_k + \Delta_{dk}$, that is, we synchronize with the time of the most delayed server, and then we look for the server m where $m = \text{argmin}\{k : \tau'_k\}$. Next frame to be played will be the one in head of buffer m. This works as long as there are frames in buffer d. If it is empty after a play, we must wait for an arrival there, because we have no information about which server will be sending next frame to play. Of course, we must wait until some time out because if server d for some reason stopped sending, we block the system if we remain waiting for its frames. After some amount of time, we move to the frame having the smallest time to play using the previous procedure.

Observe that, at the price of an extra computation at the client's side, we are able to synchronize efficiently the substreams without any signalling traffic, and using the same data that protects the system against failures.

[2]Actually, we can choose any row if all the entries of Δ have been computed; otherwise, we choose the row having the largest number of computed entries.

10.1.2 Correctness Evaluation

For testing the correctness of our implementation, we implemented a program that collects complete traces of the sequences of frames sent by the servers and received by the client. These traces allow us to determine, for each frame, which server sent it and if it was played of not by the client. The program also collects some other data as the amount of frames sent, the used bandwidth, etc.

10.1.2.1 Testing the used bandwidth

The goal of the tests we will describe here is to measure the bandwidth used by the servers. This can be compared with the values provided by a theoretical analysis of the system, to check the consistency of the implementation. We will use two different models for distributing the load among the servers. The first one consists of sharing it uniformly among the K servers, that is, all of them send the same fraction $1/K$ (quantitatively speaking) of the global stream. In the second model, we use a geometric distribution: server i sends $1/2^i$th of the stream, for $i = 1, 2, \cdots, K-1$ and server K sends the fraction $1/2^{K-1}$.

Consider the uniform case, in which we send the stream with a redundancy level of r. If $BW^{unif}_{K,i}$ denotes the bandwidth used by server i, then clearly $BW^{unif}_{K,i} = (1+r)/K$. The case of our geometric load is a little bit more involved. If $BW^{geo}_{K,i}$ is the bandwidth used by server i in this case, we have

$$BW^{geo}_{K,i} = \begin{cases} \dfrac{1}{2^i} + \dfrac{r}{2^i}\left(1 - \dfrac{1}{2^i}\right) & \text{if } i \neq K \\ \dfrac{1}{2^{K-1}} + \dfrac{r}{2^{K-1}}\left(1 - \dfrac{1}{2^{K-1}}\right) & \text{if } i = K \end{cases}, \quad K \geq i \geq 1. \tag{10.3}$$

Table 10.1: Mean Squared Error between theoretical and observed values of the used (normalized) bandwidth, as a function of the total number K of servers. The error is computed summing on i, the server index, from 1 to K.

K	Uniform	Geometric
1	0.00000	0.00000
2	0.00005	0.00005
3	0.00013	0.00322
4	0.00006	0.00310
5	0.00008	0.00243
6	0.00005	0.00207
7	0.00006	0.00171
8	0.00004	0.00149
9	0.00009	0.00134
10	0.00011	0.00125

In our tests, we used the value $r = 0.5$. Table 10.1 shows the Mean Squared Error between the bandwidth measured during our experiments and the theoretical values in the two distribution models considered, uniform and geometric. The bandwidth measures are normalized by the total bandwidth for comparison effects. We evaluated the used bandwidth by extracting information automatically computed by the standard VLC modules. This consists of multiplying the number of frames sent in each class (I, P, ...) by the average size of these frames, and then by dividing by the used time. We then get an approximation of the used bandwidth. This, plus the losses (see next subsection) explains the differences between expected and observed values. Observe that we sum over all servers, which means that the effects of the random division of the stream between the different servers is not responsible of any part of the observed differences.

10.1.2.2 Measuring losses and Perceived quality

In our preliminary prototype, there are some frame losses at the beginning of the transmission, because we assume no specific effort is made to synchronize the K servers (this extreme situation is considered for testing purposes). There is, then, a transient phase during which the system will loose information until there have been enough redundancies allowing to synchronize the flows using the procedure described before. Then, during the transmission, in other stressing experimental situations, some of the servers may have very few frames to send, which can make the synchronization process again slower to converge, until redundant frames are finally sent.

Table 10.2: Estimated loss rates after synchronization, as a function of the number K of servers used, for the two load models considered

K	loss rate (uniform load)	loss rate (geometric load)
1	0.0000	0.0000
2	0.0049	0.0049
3	0.0070	0.0080
4	0.0083	0.0066
5	0.0080	0.0070
6	0.0080	0.0072
7	0.0083	0.0220
8	0.0093	0.0186
9	0.0090	0.0182
10	0.0129	0.0222

We computed the loss rates by comparing the sent and received frame sequences. We arbitrarily eliminated the first parts of the flows until observing 50 consecutive frames correctly sent, because this implies in general that the K servers are synchronized. Again, the goal is to check that even using this rough testing procedure, the observed loss ratios are small enough. We used the same loads as in Subsection 10.1.2.1. Table 10.2 shows the observed loss rates, during $2K$ experiments using k servers, $1 \leq k \leq K$, and both the uniform and geometric load distributions. Observe the fact that in the geometric model, we measure some "peak" values of the loss rates for high values of K. This is due to the fact that in this load model there are servers that send a small number of frames. Specifically, all the servers with index $i \geq 7$ send $(1/2^6)$th of the original video. These servers never succeed in synchronizing because of the lack of redundant frames sent. This implies that the decoding time of the frames sent by these servers will not be correctly computed because the shifts will not be known by the system. The final consequence of this is that those frames will be discarded.

We consider the perceived quality of the reconstructed stream. In Section 5.2.3, we apply the PSQA technology, obtaining a *simple function*, $Q(LR, MLBS)$, that maps the loss rate, LR, and the mean lost burst size, $MLBS$, into the perceived quality. Consider the loss rates obtained using the uniform load model, as shown in Table 10.2. The worst case is when $MLBS = 1$ (see Figure 5.6 that plot the PSQA *simple function*). Using this $MLBS$ value and the loss rates given in the first column of Table 10.2, we obtain, as perceived quality estimates, values going from 10 (maximal quality) to approximately 9, which is almost maximal. For the geometric load (somehow an extreme situation), we observe loss ratios up to 0.0222, which in the worst case of $MLBS = 1$, translates into a quality level of about 5 (this corresponds to the characterization "fair" in the subjective testing area). The reason was already commented: the servers having little to send have no time to synchronize. This is telling us that the losses due to the synchronization problem we observed, even if the situation which generated them is a little bit too pessimistic, should be diminished. This is commented in the last concluding section.

These measures show that the present implementation needs some improvements to reduce the losses due to the possible lack of synchronization (even if they are not very important in number). In order to diminish these losses, we are currently working on adding extra redundancy for a better synchronization (for instance, by sending at the beginning, all the frames by all the servers, during some pre-specified time or until a pre-chosen number of frames have been sent). Also, when a server has almost no frame to send, we also can force it to send redundant data, again for allowing the client to estimate the drifts with sufficient accuracy.

10.2 GOL!P2P Architecture: a Tree-based Structured Overlay Peer-to-Peer System

This section presents our full video delivery network based on a Peer-to-Peer architecture, called GOL!P2P. It is the integration effort of the methodologies developed in the previous chapters. Our prototype is a completely free-software application, based on well-known technologies which can be executed on various operating systems.

GOL!P2P is a new approach of P2PTV networks, based on the quality perceived by final users. We measure, in real time and automatically, the perceived quality at the client side by means of the PSQA technology. Moreover, we improve the efficiency and robustness of the network, in terms of the quality it delivers, considering the lifetime clients behavior of a real system.

Our design mitigates the impact of the peer' disconnection at two levels. In the long term the system places the peers most compromised with the network (those with largest lifetime and smallest bandwidth fluctuation) near to the broadcaster, in order to obtain a more robust overlay topology. This dynamics is formalized as a mathematical optimization problem which is solved in Chapter 8. In the short term, and once defined which peers will serve the streaming to each client, we study the best streaming technique that mitigates the peers' disconnections. Our delivery scheme was called *multi-source streaming technique*; an optimal configuration of the technique (considering the lifetime dynamics) is obtained in Section 7.2, while the detail of its implementation is presented in previous section (Section 10.1). From both the long and short term perspectives, the aim is the same: to improve the perceived average video quality at the end users.

Our approach allows for a large flexibility of the system, modulated by the dynamics of the network. In particular, it is in principle possible to increase the number of substreams composing the original flow and/or the amount of redundant information sent through the network; this is used as a tool to deal with the problem of nodes leaving the network in different potential applications or particular networks. We can also control the way the load is distributed among the different substreams.

The final goal is of course to deliver a satisfactory quality level to a maximal number of clients at a minimal cost. In our network we address directly the quality as perceived by the end users as the target to maximize. For this purpose, we use the PSQA *complex function* developed in Section 5.3 and the monitoring suite presented in Chapter 9, that together are able to automatically and accurately measure the perceived quality in real-time for each client in the network.

10.2.1 The GOL!P2P Prototype

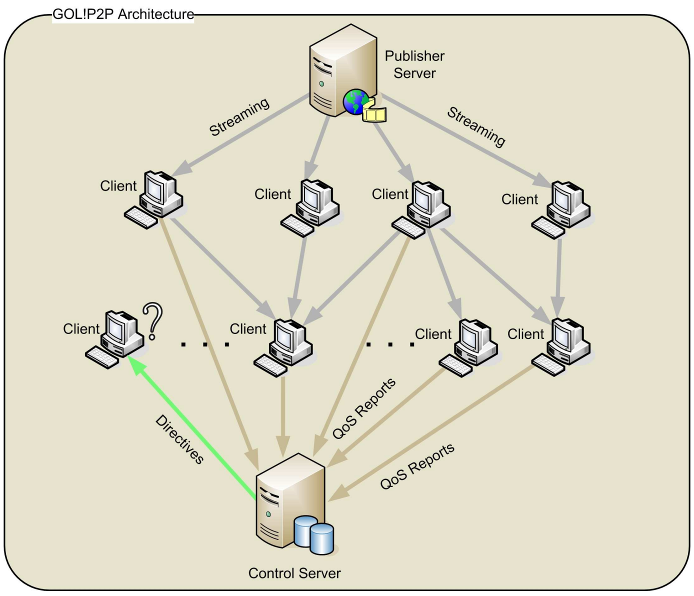


Figure 10.2: GOL!P2P architecture - The streaming server, the clients and the control server.

Inside the GOL!P2P network we can find three basic software components (Figure 10.2): the publisher server, the peers and the control server. The publisher server is an extended VLC server, where the original stream is decomposed into different substreams, which are streamed over the network. The peers are extended VLC players, who acts as a streaming client and server at the same time. Initially the peers take directives from the control server in order to know where to receive the substreams from. During the execution, each peer receives multiple substreams and using the multiple source approach it builds the original stream before playing it. Also the peers can send each of the substreams to other clients. The control server is the central manager of the P2P network, responsible in particular of its topology. The perceived quality in the network is audited automatically and in real-time at the control server, using periodical information sent by peers.

Publisher Server. As shown in Figure 10.3, we separate the publisher server into two different components, the broadcaster and the splitter. The broadcaster takes the content from a source and inserts user-data into the stream, which is used by the clients to measure the losses per frame type. All content sources are supported, such as a stored content or a live content or an Internet streaming. In our case we use a stored MPEG-4 content. The splitter decomposes the stream in sub-streams according to the previously described algorithm (Subsection 10.1.1.1). It also acts as the root node (also called node 0) inside the tree-based overlay. Both components are plugged VLC modules.

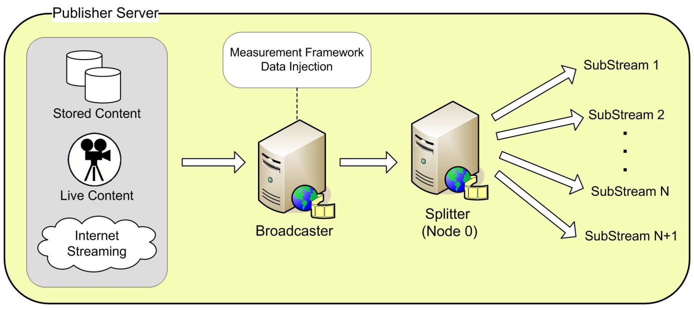


Figure 10.3: Publisher server architecture.

Peers Peers (clients) interact with the control server, implement the measurement procedures (for building the PSQA metric) and reconstruct the stream for playing it. As shown in Figure 10.4, three main modules were added to the VLC player for this purpose: the control module, the multisource module and the measurement module. The control module is in charge of the communication between the control server and the peers. It periodically reports (to the control server) the QoS parameters measured within a peer and also read directives from the control server (for instance, routing information). All communications between the peers and the control server are done over HTTP. The multisource module is capable of reconstructing the original stream (with the procedure described in Subsection 10.1.1.2) from the set of substreams, or a degraded version of it if some of the substreams had losses, or are simply missing. The measurement module is where all streams measurements are done. Inside this module the user-data inserted in the broadcaster server are parsed and used in order to calculate the loss rates. Finally in order to stream the substreams received to other clients, a VLC module called standard stream output is used.

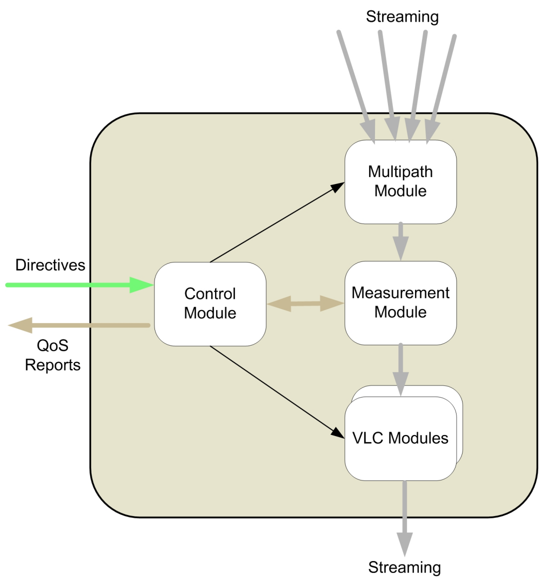


Figure 10.4: Peer server/client architecture.

Control Server. The control server is the central system manager. Its main function is to define the network topology. It also presents an interface to the administrators and managers of the P2P network. As presented in Figure 10.5, it has four components: the control server

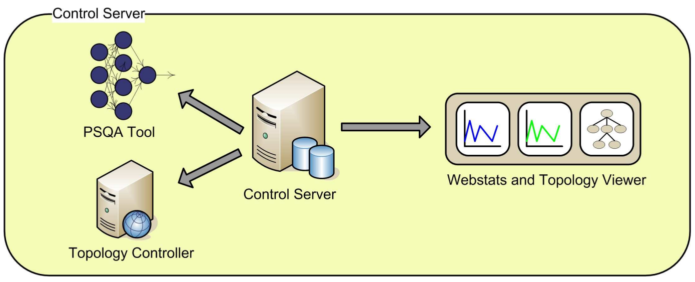


Figure 10.5: Control server architecture.

itself, a PSQA module, the topology controller and the webstats and topology viewer. The control server itself is the responsable of the communication with the peers. It reads peers' QoS reports and gives them routing directives. The information read from the peers is saved on a Round Robin Database (RRD) for its presentation. Using the peers' loss rate reports, the PSQA module gives us a measurement of the perceived quality at any instant, at any peer. The topology controller is the application responsible for the definition of network topology. This application is continuously running, calculating the (pseudo-)optimal topology per sub-stream. The topology is computed following a mathematical programing model that minimizes the impact of the peers' disconnection in the quality knowing the average lifetime behavior of each client (see the formal model and the heuristic solution used in Chapter 8). The results of this process are used by the control server to give the routing directives to the peers. The webstat and topology viewer is an application designed to present the data gathered by the control server and the network topology defined by the topology calculator to administrators and managers of the GOL!P2P network. It offers reports at different levels and types of views. It is possible to generate reports focused on a particular stream or on a specific client (where there could be more than one active stream), or perhaps on the entire network (where there could be more than one client). Finally it is possible to display the results at the frame level, possibly per frame type (I, P, B), or at the PSQA level.

Implementation. As mentioned before, clients and servers are based on VLC code. The `libavcodec (ffmpeg)` library was used in order to work with MPEG-2/4 video specifications. The container format was MPEG-TS, using the functionalities provided by `libdvbps`. We streamed over HTTP and UDP using VLC internal modules. The control server and webstat are written in PHP; they use MySQL and Oracle Berkeley Databases and both of them run over Apache. We used the RRDtool to generate the statistical graphs and Graphviz to visualize the network topology. As a difference with the monitoring suite presented in Chapter 9, the periodical reports sent by peers to the control server are delivered over HTTP as a piggybacking information of the directives, instead of SNMP[3]. The GOL!P2P application can be easily extended to support other container formats (MPEG-PS, OGM...) and transport protocols (RTP, MMS...). All libraries and applications mentioned above are free-software and they can be executed in Microsoft Windows, Unix, Linux, etc.

10.3 Extending Frontiers: Improving our Video Delivery Reference Service

In this section we test our GOL!P2P prototype. Using our P2P client and servers implementation, we deliver video on the Internet, and we verify the correct integration of each piece of software. The tests are divided into client-level and network-level tests. The former explore properties of the prototype from the user point of view, such as reconnection time, theoretical versus observed quality values, bandwidth consumption, etc. The latter refer to the global performance of the network, for instance the average behavior of the peers, the scalability of the network when the quantity of peers grows, etc.

All the tests were done in PlanetLab [259]. PlanetLab is a global research network that supports the development and test of new network services. It currently consists of 813 nodes at 401 sites. The peers' dynamics is based on the live-video service presented in Section 2.5.

To distribute the video, we use six different multi-source streaming configurations. We use the optimal configurations specified in Table 7.11, for $K \in \{1,2,3,4,5,8\}$ (see Section 7.2 for details about the procedure followed to obtain these configurations). We decide not to apply redundancy in these tests $r = 0\%$ because of strong limitations in the available bandwidth in each node of Planetlab. The bitrate of the original stream (and therefore, the overall bandwidth

[3]In this case we audit all the clients, not just a set of probes, an SNMP approach will not perform well.

considering all the sub-streams) is 512 kbps. There is no constraint on the upload bandwidth of the multi-source streaming configurations chosen. Table 10.3 shows the bandwidth consumed per sub-stream for each configuration.

Table 10.3: Upload bandwidth consumed on server i, for the multi-source streaming technique defined in Table 7.11, with K sub-streams and $r = 0\%$.

i/K	1	2	3	4	5
1	492.7	170.6	73.1	34.1	16.5
2		341.3	146.3	68.1	33.0
3			292.6	136.2	66.0
4				272.4	132.0
5					263.7
total	492.7	511.9	512.0	510.8	511.3

i/K	6	7	8	9	10
1	8.1	4.0	2.0	1.0	0.5
2	16.2	8.1	4.0	2.0	1.0
3	32.5	16.1	8.0	4.0	2.0
4	65.0	32.2	16.0	8.0	4.0
5	129.9	64.4	32.1	16.0	8.0
6	259.9	128.9	64.2	32.1	15.9
7		257.8	128.4	64.1	31.9
8			256.7	128.2	63.7
9				256.5	127.4
10					254.9
total	511.7	511.5	511.4	512.0	509.3

As in Section 8.4, we can compute a "theoretical" value of the perceived quality depending on which sub-stream is received by the user. For instance, if $K = 1$, an user has perfect quality when he/she receives the stream, and zero quality otherwise. We compute these "theoretical" values from the PSQA *complex function*, of Section 5.3.3, using the loss process model described in Section 7.2. Next subsections compare these values with the measured ones.

10.3.1 Client Performance Tests

We performed two families of tests concerning the clients' point of view: the first one concerns the impact of the peers' disconnections; the second one looks at the impact of the distance (the delay) in the network between the peer servers and the client. Each test instance has a duration of 4 minutes, and it uses the multi-source streaming technique specified in the fourth row ($K = 4$) of Table 7.11.

The peers' disconnection dynamics. In this test, we have four peers sending a different sub-stream each to a client. To measure the impact of peer disconnections in the perceived quality, we disconnect each of the sources separately and measure, in the client, the perceived quality and the reconnection time. Figure 10.6 shows the measured PSQA when one of the servers disconnects from the network. Quality degrades to 0.5 (PSQA scaled in $[0, 1]$), but this is for a short period because the network will quickly replace the disconnected peer with another connected peer.

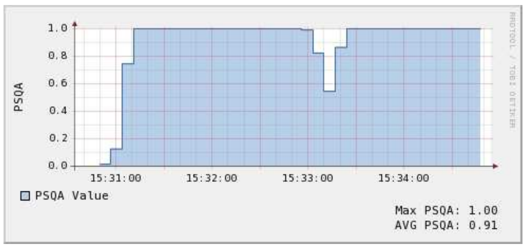


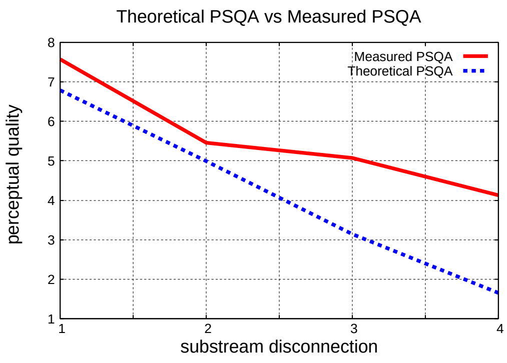
Figure 10.6: Measured PSQA when the peer that serves a given sub-stream (here, the 2nd one) disconnects from the network.

Table 10.4 and Figure 10.7(a) show the minimal PSQA value measured in the test for each peer disconnection. It also compares the measured PSQA with the theoretical value, showing that they conserve the same trend (with slightly higher values for the measured PSQA). See that the sub-streams with highest value (sent by the best nodes) have the major impact on quality (as we expect from the methodology presented in Section 7.2).

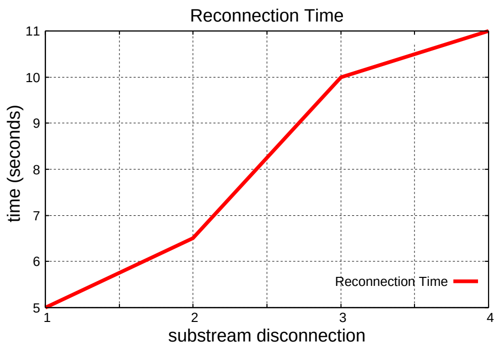
Table 10.4: Theoretical PSQA vs Measured PSQA when disconnections occur.

Substream	Theoretical PSQA	Measured PSQA
1	6.790	7.576
2	4.997	5.453
3	3.142	5.073
4	1.655	4.124

Finally, Figure 10.7(b) shows the reconnection time needed by the client to receive from another peer the lost sub-stream. It shows that the reconnection time depends on the sub-stream bandwidth: with higher bandwidth more time is needed for the reconnection (due to the buffer filling).



(a) Theoretical PSQA vs measured PSQA when disconnections occur.



(b) Reconnection time when disconnections occur.

Figure 10.7: Summary results of peer disconnection impact.

The peers' inter-delay. In this test, we have the same overlay topology and configuration as before. But we place the peers in two different geographical location scenarios, one close to the client, the other one far from it. The distances between the nodes are measured as the corresponding end-to-end delays, calculated by averaging the round trip time of ICMP packets. To measure the impact of the delay in the perceived quality, we disconnect each of the four servers separately and measure, at the client's side, the perceived quality and the reconnection time. The delay from the peers to the client of the two scenarios, called "close" and "far", are summarized in Table 10.5.

Table 10.5: Delay from the peers to the client in our two scenarios.

Peer	Distance (ms)	
	"close"	"far"
peer 1	21.0	266
peer 2	28.1	281
peer 3	28.1	281
peer 4	27.9	281
peer of reconnection 0	0.15	273

The mean/median/minimal perceived quality at the client side is measured with the PSQA technique. The connection and reconnection times of the multi-source streaming technique are also measured. The results of the tests show no significant impact of the delay in the perceived quality. The tests show no impact in the reconnection time neither. For instance, in Table 10.6 we show the results when peer "1" is disconnected. We show the average measurements in the period without disconnections and in the period with disconnections. In the two scenarios, "close" and "far", the quality measurements have negligible differences, and, as we expected, the connection time (streaming start-up) is shorter in the "close" scenario. But, surprisingly, the reconnection time is longer in the "close" scenario than in the "far" scenario. This behavior

Table 10.6: Peers' inter-delay measurements.

measure	Periods without disc.	Periods with disc.	total
	"close" scenario		
mean PSQA	9.947	6.543	8.812
median PSQA	10.000	6.454	10.000
min PSQA	9.209	4.678	4.678
theoretical PSQA	10.000	6.790	-
connection time (sec.)	-	-	8.5
reconnection time (sec.)	-	16	-
	"far" scenario		
mean PSQA	9.988	6.500	8.825
median PSQA	10.000	5.566	10.000
min PSQA	9.632	5.068	5.068
theoretical PSQA	10.000	6.790	-
connection time (sec.)	-	-	9
reconnection time (sec.)	-	12	-

is justified because the delay time is negligible with respect to other factors, as for instance with respect to the central network reconfiguration time, or to the buffering time.

10.3.2 Network Behavior Tests

We performed two tests that evaluated the performance and scalability of the global network. The first test evaluates the performance of the network when different configurations of the multi-source streaming technique are used. The second test determines the impact of the peers' connection dynamics in the global quality of the network. Each test instance has a duration of 400 seconds.

The number of sub-streams. In this test, we have 30 peers with a specific connection/disconnection dynamics extracted from a real situation. In Figure 10.8 we show the number of clients connected to the network, varying between 17 and 21 simultaneous connections. Each peer executes in a particular Planetlab node.

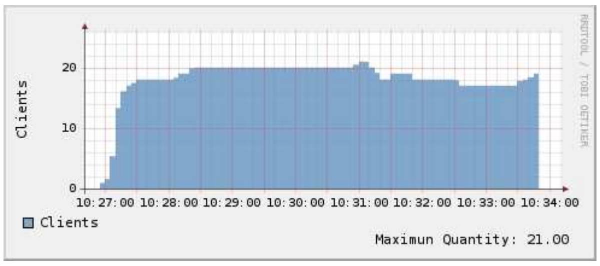


Figure 10.8: Number of clients connected to the network.

Using the same peers dynamics, we tried six different multi-source streaming configurations and evaluated the performance of each case. The configurations used correspond to the number of sub-streams in $K \in \{1, 2, 3, 4, 5, 8\}$, Table 7.11.

We evaluated the mean and the median PSQA of all the connected peers in the streaming period (the streaming has a duration of 400 seconds). Figure 10.9 shows the results. The schemes with two, three and four sub-streams show the best performance.

Figures 10.10(a) and 10.10(b) explain why the quality does not increase with the number of sub-streams. Figure 10.10(a) shows the number of clients that don't receive all the sub-streams because of the disconnection of another peer. Considering only 30 nodes in the network, with a fixed connection/disconnection dynamics, when the number of sub-streams increases the probability of been served from a peer that will be disconnected also increases. Therefore, when the number of sub-streams increases, more clients will probably have losses because of other peers' disconnections. But this doesn't imply a global degradation of the quality.

The topology of the network is computed (by the control server) by means of the greedy algorithm explained in Chapter 8. When the number of sub-streams increases, the number of possible topologies also increases. This means that, potentially, more intelligent assignments can be done, but also more computation time is needed at the control server for the topology calculation. Figure 10.10(b) shows the time needed by the control server to connect a client

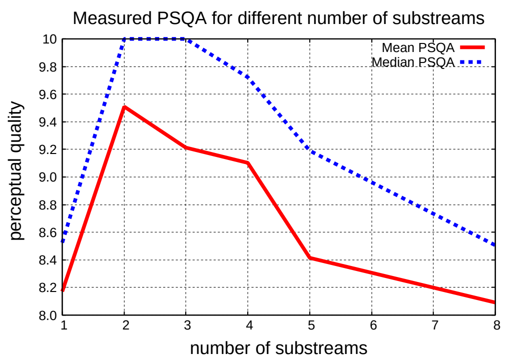
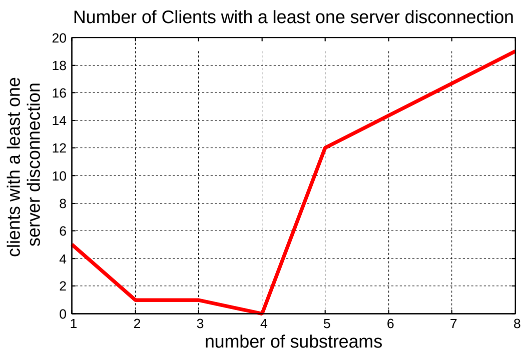
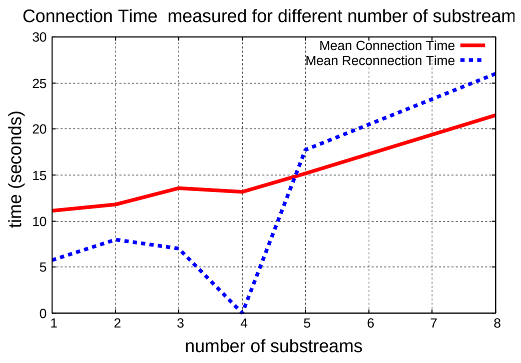


Figure 10.9: PSQA measured for each schema tested.



(a) [Number of clients with at least one disconnection peer.



(b) Connection time.

Figure 10.10: Connection time and number of clients with at least one disconnection peer per number of sub-streams.

(at the first time or after a peer' disconnection). This time basically consists in the topology computation time. When the original streaming is used (there is only one sub-stream), there are only five clients who loose the peer that serve their streaming, and therefore they do not receive anything (the other clients receive the streaming perfectly). The connection and reconnection time is also very low for this scheme (between 10 and 5 seconds). But the average global performance of the one sub-stream scheme is very poor because these five clients have the worst quality (zero) in the failure period. In the four sub-streams scheme only the leaves in the graph disconnect (and therefore there are no losses and reconnection time). But, when the number of sub-streams increases, the computation time (the connection and reconnection time) increases, and therefore the losses impact in a longer period. For the schemes with five sub-streams and more, there are too many disturbed clients because of other peers' disconnections, and despite of the fact that these disconnections only affect in part the streaming quality, the average global quality decreases.

The peers' connection/disconnection dynamics. In this test, we use again the multi-source streaming technique with the configuration given in the fourth row of the Table 7.11, which has four sub-streams. To measure the impact of the peers' connection/disconnection dynamics in the perceived quality, we evaluate four real scenarios with different dynamics. Table 10.7 shows the number of peers' connections and peers' disconnection for each scenario.

Table 10.7: Peers' connection/disconnection dynamics in the four network tests scenarios.

scenario	total client connections	total client disconnections
1	30	6
2	41	10
3	49	14
4	70	59

Each test instance has a duration of 400 seconds. For each test, we measure the average mean and median perceived quality for all the peers. Table 10.8 and Figure 10.11 present the results. As we expected, the average perceived quality decreases when the network' dynamics increases.

Table 10.8: Measured PSQA for each scenario.

Scenario	Mean PSQA	Median PSQA
1	8.7	9.3
2	8.0	8.8
3	7.9	8.1
4	6.8	7.1

Extending our Reference Service Network with our Hybrid GOL!P2P Prototype. The network behavior tests demonstrate the capability of our prototype to extend a Content Delivery Network infrastructure in real scenarios. The fresh clients bandwidth can be used to deliver

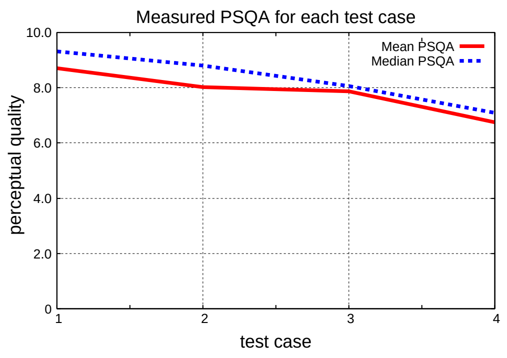


Figure 10.11: Measured PSQA for each scenario.

video to more clients or to improve the video quality of the stream. With very few clients (approximately thirty in our tests) we have an excellent quality on the average ($Q \geq 9$), with a very early version of the prototype.

10.4 Summary and Conclusions

This chapter discusses, at a high level, the implementation challenges behind the techniques and models developed in previous chapters. We propose a general architecture of a P2P live video streaming system, with a central controller, with a very low signaling overhead, and using the user perceived quality as the main quality criteria (perceived quality is the main component of QoE).

Our solution is based on a specific multi-source approach, and is extremely flexible: its configuration offers many parameters for a fine tuning. The main ones are the number of sub-streams (sources), the way we distribute the load among them, and the level of redundancy. We can also control them differently for each frame type (I, P, B). Using the information computed by analytical models, we define the optimal streaming configuration to be used, to ensure with a certain confidence (i.e., probability) a given QoE level[4], for a particular network dynamics.

The tree-based overlay topology is centrally maintained, using an efficient greedy algorithm.

The proposed method was prototyped and tested in a real environment using the PlanetLab platform. First, we show that a multiple source method for distributing live video can be efficiently implemented with a fine control on its redundant capacities. One of the results we obtained is that the redundancy of our system not only provides robustness against servers' failures (typically, due to nodes leaving the networks) but also allows to implement a method for synchronizing the flows without any other signalling process. The first measures we did show also that the present prototype needs some improvements to reduce the losses due to the possible lack of synchronization (even if they are not very important in number). In order to

[4]It is possible to address other tradeoffs.

diminish these losses, we are currently working on adding extra redundancy for a better synchronization (for instance, by sending at the beginning, all the frames by all the servers, during some pre-specified time or until a pre-chosen number of frames have been sent). Also, when a server has almost no frame to send, we can force it to send redundant data, again allowing the client to estimate the drifts with enough accuracy for avoid losses due to synchronization. Second, with respect to the global system, some preliminary results show the feasibility of the prototype capability for real life usage. Client level and network level tests validate the models and methodologies used. All the results show a very good behavior from the perceptual quality point of view, allowing us to think that extending a CDN with a P2P distribution will be a common feature in the near future.

Chapter 11

Challenges in VoD and MyTV services

In the previous chapters we studied different aspects of a broadcast video service. Live video streaming is one of the most challenging services, because of its bandwidth consumption and of its real-time restrictions. But present day Video Delivery Networks must provide a series of complementary services.

In this chapter, we study Video on Demand (VoD) and MyTV complementary services. In particular, we focus on a caching search strategy for these services, because the content discovery is the largest challenge here. In Section 11.2 we present a model of the impact that cache expiration times have on the total number of correct answers to queries in a content network, and on the bandwidth usage. We develop a mathematical programming formulation of the model, which is tested in Section 11.4 with a set of P2P test cases and a DNS system case coming from real traces.

The state of the art of content discovery and some models have been presented at three conferences: 33th Argentine Conference on Computer Science and Operational Research (JAIIO'04), Jornadas de Informática e Investigación Operativa (JIIO'04), and XIII Congreso Latino-Ibero-americano de Investigación Operativa (CLAIO'06) [276, 277, 279] (listed at the Introduction as [jaiio04] [jiio04][claio06]). The mathematical programming model of cache expiration times has been published at the International Network Optimization Conference (INOC'07) [280] (listed at the Introduction as [inoc07]). The file-sharing P2P system test scenario has been presented at the 3rd international IFIP/ACM Latin American conference on Networking (LANC'05) [278] (listed at the Introduction as [lanc05]). The Domain Name System (DNS) test scenario has been presented at the 3rd international conference on the Quantitative Evaluation of Systems (QEST'06) [42] (listed at the Introduction as [qest06]).

11.1 Video-on-Demand and MyTV services

Interactive TV has different meanings depending on the VDN capabilities and user behaviors. Some services are essentials for the next generation of VDN's, the most important are:

Electronic Program Guide (EPG). The Electronic Program Guide (EPG) is an on-screen guide to scheduled broadcast programs, allowing a viewer to navigate, select, and search content by time, title, channel, genre, etc.

In our P2P context, this service is very simple to implement: It can be implemented using a traditional Web Server-Client approach of some variants of Podcasting or Broadcatching (see section 11.1.1).

Time Shift TV (TSTV). Time shifting is the capability of recording, pausing, playing and rewinding a live broadcast streaming for short periods, giving the possibility to be viewed twice or at a time more convenient to the user.

Personal Video Recorder (PVR). Personal Video recorder, also know as Digital Video Recorder (DVR), it is the capability of recording a broadcast streaming for a long period to be played after by the user. As TSTV, PVR provides basic media player functionalities (pause, fast forward, fast rewind, slow forward, slow rewind, etc.). In some VDN's, instead of having the recording capability in the customer device (knows as a Set-top-box), these functionalities are implemented in the network. This special kind of PVR is knows as Network Personal Video Recorder (NPVR), and its development has been motivated by economical reasons.

To add TSTV and PVR functionalities to our prototype is an easy task. Because, in our P2P network, the user device is a computer, and it is possible to record a streaming with any multimedia player, particullary with the VideoLan Client. In this case, there is no need to implement NPVR.

Video on Demand (VoD). Video on Demand allow users to watch pre-stored movies or television programs that are available in the VDN. First, the user discovers the video that he wants to watch, selects it, and finally a unicast streaming or progressive download (depending on the kind of network) starts from the network to user. VoD services provide the user with traditional PVR functionality (pause, fast forward, fast rewind, slow forward, slow rewind, jump to previous/future chaper, etc.). In some VDN's is not possible to implement a unicast stream or progressive download, a variant of VoD called Near Video on Demand (NVoD) is used in these networks. NVoD is implemented with a multicast streaming, when a user selects a video to watch, he has to wait until the next multicast streaming of this video starts, implying that NVoD is only suitable for very popular videos. Push video on demand (PVoD) is used, in systems that lack interactivity, to emulate the VoD service. PVoD uses a Personal Video Recorder (PVR), at the user home, to record pre-defined videos (often transmitted at the night idle capacity). The user can then watch the downloaded videos at a time of their choosing.

VoD is a high bandwith resource consuming service. In spite of being very attractive for the users, its implementation is limited to some new VDN's. Essentially, VoD is a special case of file distribution, with very light real-time constraints. This allows that the VoD should be implemented in almost all Content Delivery Network (CDN) and Peer-to-Peer file sharing systems. In our context a P2P solution is the most suitable. Internet-based VoD services allow to the user of offering his own videos. The main technical problem is that peers connect and disconnect with high frequencies, and then, the videos that they publish are also highly variable. And thus, that to discover and to download a video of the network can be a difficult task. This is the main challenge in P2P design for VoD: to offer the content needed by the clients in a highly varying environment. Efficient download in a file-sharing P2P network was widely studied (see for instance [15, 47, 209, 216]). In this chapter we study the impact of the content dynamics in the search performance, i.e. the content discovery[1].

Pay per View (PPV). Pay per view is the service in which users can purchase broadcast events to be seen on TV and pay for the private view of that event to their homes. The event is shown at the same time to everyone ordering it, as opposed to video on demand service, which allows viewers to see the event at any time.

PPV adds the technological need for a Billing System and a Digital Right Management (i.e. a encrypted streaming method and the management of the associate encryption key). PPV implementation over a P2P network is possible, for instance Joost [185] and new BitTorrent [31] have this technology.

MyTV. MyTV (also know as ShareTV or selfcast) is the possibility to the user of offering it own broadcast channel. As an extension of the video-conference, MyTV is suitable for familiar events, or for small producers.

The Streaming of MyTV service can be high bandwidth consuming. From a technical point of view, it can be implemented like any broadcast streaming, therefore in our shared resource P2P approach it is not a problem. But MyTV service has the same problem than the VoD service, with high dynamics in the contents. In this chapter we also study the search performance for MyTV service.

Others. A lot of interactive services are developed today for the next generation VDN's. For example multi-player games, voting, interactive advertising, etc.

In the rest of the chapter we study the content discovery for the VoD and MyTV services.

11.1.1 Making Content Accessible: Video Podcast and Broadcatching are not enough

Video podcast[2] , also called vodcast, is a online announcing mechanism for video on demand (VoD), where the clients automatically discover new files. The client subscribes to a channel, where he receives regular programs. The client can view these programs at his own leisure. It can be considered as an Internet equivalent of a broadcast medium, and specifically of the Time Shift TV service[3].

In podcasting, the content publisher begins by making a video file available to delivery on the network through some known Uniform Resource Identifier (URI). Then, the content publisher announces the existence of that video file by referencing it in a web feed. The web feed is a XML file that provides a list of URIs with additional information (such as title, description,

[1]Specially in the case when the clients can submit their own videos.

[2]The word "PodCasting" is a merger of the words PoD (Portable on Demand) and broadcasting. The iPod, of Apple Inc., owes its name to the PoD concept. The first podcasting service was developed for the iPod. Today, podcast is no longer specifically related to the iPod.

[3]Some Personal Video Recorders (PVRs), such as TiVo, allow podcasting as a new service.

and publish date). The web feed is generally published via Really Simple Syndication (RSS), or via the proposed standard [160, 163] Atom Syndication Format (Atom).

Syndication of the video on demand has benefits for both publishers and clients. Publishers can automatically announce content for their subscribers. Moreover, they can provide dedicated content depending on each specific client interest. Clients can easily discover selective content, and merge content from different publishers (broadcatching allows a many-to-one communication instead the traditional one-to-many of the broadcasting).

Extending the video podcasting concept, **broadcatching** is the automated download of content that has been made available on the network using podcasting.

Broadcatching allows a complete and easy to use solution, for the discovery and distribution of the video on demand. The technology is easy adaptable to a P2P network. A first integration can be a centralized server syndication (using the standard RSS mechanism) where each URI listed in the web feed is actually an identifier of the file in a P2P distribution network (for instance a .torrent file in a Bittorrent network). A better integration can be the elimination of the inefficient polling of the RSS feed in the central server, and the distribution of the feed also with the P2P network. A light notification can be sent to each client (typically via XML-RPC) to announce the new web feed. Then the client downloads the web feed from the P2P network, and after, the client automatically downloads the content, from the P2P network, following the instructions in the feed.

But, as we see, broadcatching does not adapt well to very dynamic content. Today, a successful video delivery network has to allow users to publish their own content (not only the content of the content provider), already be in the form of video on demand (VoD) or live TV channels (MyTV). VoD of user content has the same design challenges that actual file-sharing P2P systems, where the content and the peers has high dynamics. A efficient distribution can be achieved using peer-to-peer technologies, but the discovery of the dynamic content can not be implemented with traditional announcement methods (like podcasting). Instead, search mechanisms have to be implemented in the network, in order to obtain an efficient discovery of the dynamic content. The same situation happen with the MyTV service, with the difference that the distribution on the P2P network is similar to the live TV distribution (instead of the file-sharing distribution).

11.1.2 Using a Search Caching approach: Network Structure Roles and Workload

As we have previously discussed (in Section 2.3), in a Content Network the addressing and routing are based on the content description, instead of on its location. This means that every content network is actually a knowledge network, where the knowledge is the information about the location of the nodes where each specific content is to be found: this is "meta-information", in the sense of being the information about the information contents themselves. The objective of the network is to be able to answer each content query with the most complete possible set of nodes where this content is to be found. This corresponds to discover the content location in the most effective and efficient possible way. There are two main strategies to discover the meta-information, namely publication and search. By publication we mean the process by which a network node unrequestedly sends meta-information it possesses to the remaining nodes (for instance, broadcatching). By search we mean the process by which a node asks the remaining ones to send it the meta-information they possess. By analogy with logistics, we can say that publication is an "information push" strategy, and search an "information pull" strategy. As both nodes and contents are continuously going in and out of the network, the task of maintaining updated the network meta-information is very difficult and represents an important communication cost. Both publishing and search can contribute towards this task, but their relative efficiency varies, so that there is a tradeoff between their frequency and modality of application. In this context, cache nodes are used to hold the available meta-information. As this information is continuously getting outdated, the cache nodes must decide when to discard it, which means increasing communication overhead for the sake of improving the quality of the answers.

In this chapter, we develop a simplified model of a content network, and in particular of the number of correct answers to a query as a function of the information expiration times used at the cache nodes, presented in Section 11.2. To the best of our knowledge, this is an aspect that has not been previously treated analytically in the literature (see our study of the related work on Section 11.3). This model gives rise to a mathematical programming formulation, which can be used to find the expiration times maximizing the correct answers to the queries received. Two numerical illustration are shown in Section 11.4. One is based on a file-sharing P2P system, that presents a similar dynamics to the expected dynamics in the VoD service (with content uploaded by the users). The other one is based on the DNS system, with a dynamics behavior similar to the MyTV service. The chapter concludes, in Section 11.5, with general results and discussion.

11.2 Making Video Libraries Efficiently Searchable

This section formalizes the problem of caching meta-information in a content network in order to maximize the number of correct answers to the queries, while respecting the bandwidth constraints; this will be our Content Caching Problem (CCP).

11.2.1 Preliminary Definitions

Network components description. We will look at the content network as composed of *source nodes* and *querying nodes* (which may be the same), of *cache nodes* (also called aggregation nodes), and of a *backbone* (which will not be further modeled); a graphical representation can be seen in Figure 11.1. This division is actually virtual, as a same physical node may act at the same time as a source node, a querying node, a cache node, and a backbone node. We will also separately model the contents of the network (which will belong to a set C). The content network is considered to be in steady state, so that we will not need to explicitly model the time; this assumption is justified by the fact that the time rate at which contents appear and disappear, and cache expiration times, are usually much faster than the times by which the statistical properties of the user population change.

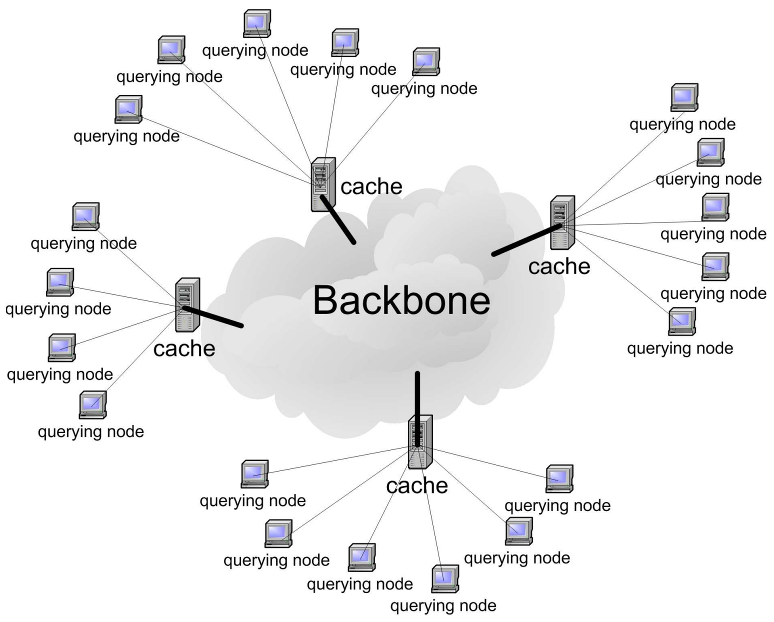


Figure 11.1: Simplified view of a content network.

The users of the network, the *querying nodes*, will query about each content k with a different query frequency f_k. We suppose that the number of users is large enough so that for each content, the queries follow a Poisson process of rate f_k. This means that $S_k(T)$, the number of queries for content k in a given time interval T, will have the following distribution:

$$p(S_k(T)=n)=\frac{(f_kT)^n e^{-f_kT}}{n!}, \forall k \in C, \forall n \in \mathbb{N}, \forall T \in \mathbb{R}^+.$$

Also T_{S_k}, the time between two consecutive queries, will be an exponentially distributed random variable with parameter f_k:

$$p(T_{S_k} \leq t)=\begin{cases}1-e^{-f_kt} & t \geq 0\\ 0 & t<0\end{cases}$$

$$\overline{T_{S_k}}=\mathrm{E}\{T_{S_k}\}=\frac{1}{f_k}.$$

The contents will be located in the *source nodes*; each source node decides when to start and when to end lodging the different contents. This leads to a different birth-and-death process (see Figure 11.2) for each content k, which we will suppose will be of $M/M/\infty$ type and parameters λ_k and μ_k (respectively, the rates of start and end of lodgment of content k at a source node); if we suppose that at moment t_0 the network is in stationary state, and $A_k(t_0)$ is the (random) number of source nodes lodging content k at t_0 we have that:

$$p(A_k(t_0)=n)=\frac{(\frac{\lambda_k}{\mu_k})^n e^{-\frac{\lambda_k}{\mu_k}}}{n!}, \forall k \in C, \forall n \in \mathbb{N}.$$

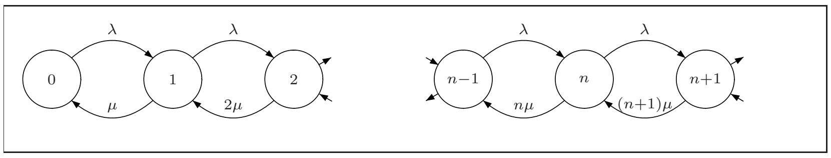


Figure 11.2: Birth and Death Process.

From this distribution, we can find the expected number of source nodes lodging content k (i.e., the expected number of times this content will be replicated in the network):

$$\overline{A_k} = \mathrm{E}\{\, A_k(t_0)\} = \sum_{n\geq 0} np(A_k(t_0) = n) = \sum_{n\geq 1} \frac{(\frac{\lambda_k}{\mu_k})^n e^{-\frac{\lambda_k}{\mu_k}}}{(n-1)!} = \frac{\lambda_k}{\mu_k}.$$

The only routing nodes we will consider are *cache nodes*. In general, querying nodes are not able to search directly in the backbone, and usually connect to at least one aggregation node in order to route their queries. The aggregation node concentrates all queries of its connected nodes and consults the backbone when it is not able to directly answer the queries received. One of the objectives of having aggregation nodes is to minimize the number of searches in the backbone; to do this, aggregation nodes maintain a cache of the results of recent queries, and are then also called cache nodes. The behavior of a cache node is very simple: when a query over content k arrives, if the answer is present in the cache it is returned; otherwise, the cache node starts a search in the backbone to obtain the information and answer the query; this information is then stored in the cache, for a prefixed time d_k, afterwards it expires.

One of the reasons for deleting out-dated information is that the results of a query will only be valid for a given time interval, as the nodes which hosted this content can disconnect or delete the content of interest, and new nodes can connect or start to publish the content. Suppose the cache node queried the backbone at time t_0, for content k, and received in answer the information about $A_k(t_0)$ source nodes which hosted this content at that time. From then on, we can consider that the number of valid locations for content k known to the cache node will evolve like a stochastic pure-death process (see Figure 11.3), with death parameter μ_k, as the source nodes will disconnect or delete the contents, until a new query is routed to the backbone.

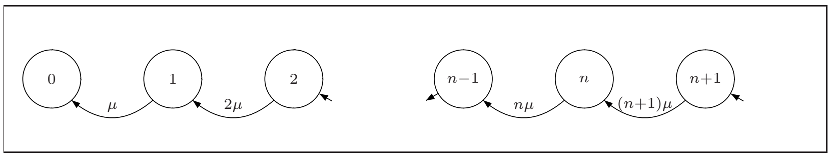


Figure 11.3: Death Process.

We can then compute the mean number of valid locations known by a cache node at time $t_0 + t$ when the last query answered by the backbone has been at time t_0:

$$\begin{aligned}
\left\{ \begin{array}{c} \text{mean number of valid content} \\ \text{locations } t \text{ time units after} \\ \text{the last backbone query} \end{array} \right\} &= \sum_{n \geq 0} np(A_k(t_0) = n)p(T_{V_k} > t_0 + t | T_{V_k} > t_0) \\
&= \sum_{n \geq 0} np(A_k(t_0) = n)p(T_{V_k} > t) \\
&= \sum_{n \geq 1} \frac{(\frac{\lambda_k}{\mu_k})^n e^{-\frac{\lambda_k}{\mu_k}}}{(n-1)!} e^{-\mu_k t} \\
&= \frac{\lambda_k}{\mu_k} e^{-\mu_k t}.
\end{aligned}$$

The behavior of a cache node is then essentially composed of a repeated cycle, which starts with a first query of content k, leading to a backbone search; then a period of fixed duration d_k, where all queries arriving are answered with the information contained in the cache memory; and then, after the expiration of the cache contents, a period of random duration, until a new query for content k arrives, re-starting all the cycle again. By the hypothesis of Poisson arrivals for queries, this last period follows an exponential distribution, of parameter f_k (the query frequency). Figure 11.4 shows a scheme of this cycle, where we denote by $T_{C_k} = d_k$ the

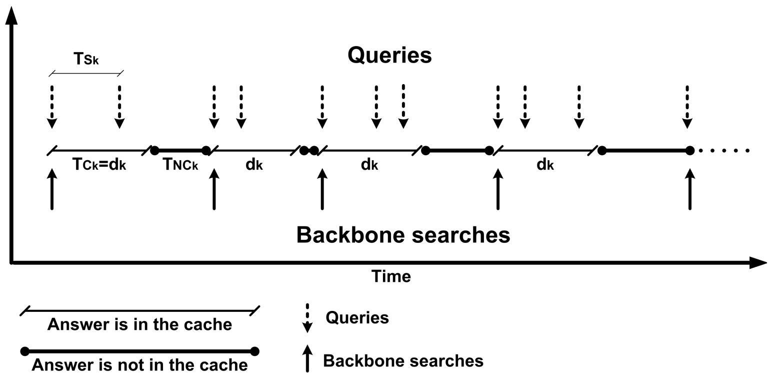


Figure 11.4: 2-cyclic behavior at cache nodes.

fixed period where the contents are cached, and by T_{NC_k}[4] the period where the contents are not cached. The mean length of the cycle is then $d_k + \frac{1}{f_k}$; in each cycle there is only a single search in the backbone (when the cycle starts), this can be used to compute the rate of backbone searches as follows:

$$\left\{ \begin{array}{c} \text{backbone searches} \\ \text{per time unit} \end{array} \right\} = \frac{\text{number of searches}}{\text{total cycle time}} = \frac{1}{d_k + \frac{1}{f_k}} = \frac{f_k}{1 + d_k f_k}.$$

[4]T_{NC_k} is an exponentially distributed random variable with parameter f_k, because the $p(T_{S_k} > s + t \mid T_{S_k} > t) = p(T_{S_k} > s)$ property.

As the query frequency is fixed externally, the only free variables we can adjust at cache nodes to define their behavior are the content expiration dates d_k for every content k.

Bandwidth constraints. Cache nodes have input and output bandwidth constraints, which can limit the number of queries they can receive, process, answer and eventually pass on to the backbone. We will try to formulate these constraints in terms of the previously defined parameters and of the free variables d_k. We denote by BW_{IN} and BW_{OUT} the maximum input and output bandwidth a cache node is able to employ. We suppose that each query the cache nodes receives employs β_S bytes in mean, and that its answer employs α_S bytes per location information to be sent (then, the answer varies in size depending the number of known node locations where a content is stored). We also use as additional parameters β_B, the message size of queries to be sent to the backbone, and α_B which is the message size per location of the answers received from the backbone.

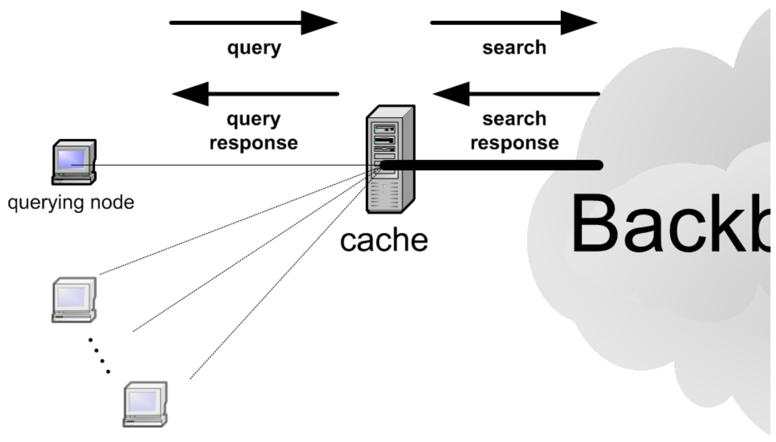


Figure 11.5: Bandwidth in cache nodes.

As we shown in Figure 11.5, the input bandwidth to be used by the cache node corresponds to the sum of the size of the queries received from the querying nodes (at a rate f_k per content k), and of the answers sent by the backbone when queried about a specific content. As we know that the backbone search frequency is $\frac{f_k}{1+d_k f_k}$, and the mean number of content k locations in the backbone is $\overline{A_k} = \frac{\lambda_k}{\mu_k}$, we arrive to the following formula for the input bandwidth:

$$\beta_S \sum_{k \in C} f_k + \alpha_B \sum_{k \in C} \frac{f_k}{1 + d_k f_k} \overline{A_k}.$$

Similarly, the output bandwidth corresponds to the sum of the queries transmitted to the backbone plus the content locations answered to the querying nodes in response to their queries, leading to the formulation of the output bandwidth:

$$\alpha_S \sum_{k \in C} f_k \overline{A_k} + \beta_B \sum_{k \in C} \frac{f_k}{1 + d_k f_k}.$$

We can then mathematically formulate the bandwidth constraints as follows:

$$\beta_S \sum_{k\in C} f_k + \alpha_B \sum_{k\in C} \frac{f_k}{1+d_k f_k}\overline{A_k} \leq BW_{IN},$$

$$\alpha_S \sum_{k\in C} f_k\overline{A_k} + \beta_B \sum_{k\in C} \frac{f_k}{1+d_k f_k} \leq BW_{OUT}.$$

Expected number of correct answers. The network primary objective is to be able to give the most complete correct information to the queries received. To formalize this objective, we develop an expression for the number of correct answers (i.e., the number of valid content locations) answered to the querying nodes. In particular, if we denote by R_k the random variable corresponding to the number of content locations answered to a query for content k, we want to compute its expected value $\overline{R_k}$. We know that during a cache node cycle, there will be at least one query (at the start of the cycle), and a random number of additional queries during the period where the content locations are stored in the cache, of duration d_k (as when the cache contents expire, the first query arriving will lead to the start of a new cycle). This leads to the following formulation for each content k:

$$\begin{aligned}\overline{R_k} &= \mathrm{E}\left\{R_k\right\} = \sum_{n\geq 0} \mathrm{E}\left\{R_k | n \text{ additional queries}\right\} p\left(n \text{ additional queries}\right)\\ &= \mathrm{E}\left\{R_{k,NC}\right\} p\left(0 \text{ additional queries}\right) + \\ &\sum_{n\geq 1}\left(\frac{\mathrm{E}\left\{R_{k,NC}\right\} + \sum_{m=1}^{n} \mathrm{E}\left\{R_{k,C_m} | n \text{ additional queries}\right\}}{n+1}\right\} p\left(n \text{ additional queries}\right).\end{aligned}$$

Where $R_{k,NC}$ is the answer to the initial query (transmitted to the backbone, and whose answers are stored in the cache), and $R_{k,C_1}, .., R_{k,C_n}$ are the answers to the following queries during the time period starting with the first query and of duration d_k. The expected number of correct responses to the first query is exactly the expected number of nodes hosting the contents, $\mathrm{E}\left\{R_{k,NC}\right\} = \overline{A_k} = \frac{\lambda_k}{\mu_k}$.

For the following queries, we use on one hand the fact that query arrivals follow a Poisson process of rate f_k, so that the probability of observing n arrivals during a time interval of length d_k is:

$$p(S_k(d_k) = n) = \frac{(f_k d_k)^n e^{-f_k d_k}}{n!}, \forall k \in C, \forall n \in \mathbb{N}, \forall d_k \in \mathbb{R}^+.$$

On the other hand, it is a well-known fact (see for instance the discussion in [178]) that the distribution of the arrivals of a Poisson process within a fixed interval follow an uniform distribution. This means that the expected mean value of the number of answers received to the queries during this interval will be equal to the expected value of valid content locations in the interval (i.e, the expectation over the queries will be equal to the expectation over the time interval, a PASTA -Poisson Arrivals See Time Averages- result). As the number of valid know locations known at time t after the last query is equal to $\frac{\lambda_k}{\mu_k} e^{-\mu_k t}$, then its expectation over the interval of duration d_k is:

$$\frac{\int_0^{d_k} \frac{\lambda_k}{\mu_k} e^{-\mu_k t} \delta t}{d_k} = \frac{\lambda_k}{\mu_k^2 d_k} \left(1 - e^{-\mu_k d_k}\right).$$

Then we have that:

$$\begin{aligned} \sum_{m=1}^{n} \mathrm{E}\left\{R_{k,C_m} | n \text{ queries}\right\} p(n \text{ queries}) &= n \left\{ \begin{array}{c} \text{mean number of valid locations} \\ \text{known to the cache node} \\ \text{in time interval } (t_0, t_0 + d_k] \end{array} \right\} \\ &= n \frac{\lambda_k}{\mu_k^2 d_k} \left(1 - e^{-\mu_k d_k}\right) \forall k \in C. \end{aligned}$$

Combining all these results, we find:

$$\begin{aligned} \overline{R_k} &= \mathrm{E}\left\{R_k\right\} = \sum_{n \geq 0} \mathrm{E}\left\{R_k | n \text{ additional queries}\right\} p\left(n \text{ additional queries}\right) \\ &= \mathrm{E}\left\{R_{k,NC}\right\} p\left(0 \text{ additional queries}\right) + \\ & \sum_{n \geq 1} \left(\frac{\mathrm{E}\left\{R_{k,NC}\right\} + \sum_{m=1}^{n} \mathrm{E}\left\{R_{k,C_m} | n \text{ additional queries}\right\}}{n+1} \right\} p\left(n \text{ additional queries}\right) \\ &= \frac{\lambda_k}{\mu_k^2 f_k d_k} \left[\mu_k \left(1 - e^{-f_k d_k}\right) + f_k \left(1 - e^{-\mu_k d_k}\right) - \frac{1}{d_k} \left(1 - e^{-f_k d_k}\right) \left(1 - e^{-\mu_k d_k}\right) \right]. \end{aligned}$$

Objective Function: Cache Effectiveness, ϵ. The *CCP* problem models the performance in a cache node, where one tries to maximize the effectiveness (understanding it as the probability of finding the sources of the contents) without neglecting the efficiency (limiting the consumption of bandwidth in the nodes).

Knowing $\overline{R_k}$, we can compute the expected number of correct answers in the network, taking into account all contents and its frequencies:

$$\sum_{k \in C} \overline{R_k} f_k = \sum_{k \in C} \frac{\lambda_k}{\mu_k^2 d_k} \left[\mu_k \left(1 - e^{-f_k d_k}\right) + f_k \left(1 - e^{-\mu_k d_k}\right) - \frac{1}{d_k} \left(1 - e^{-f_k d_k}\right) \left(1 - e^{-\mu_k d_k}\right) \right].$$

If we suppose that a cache node has infinite bandwidth, it is not necessary to cache any answers, and it is possible to search in the backbone for each user request. In this situation all sources of each content are answered, so $\overline{R_{k,\text{IDEAL}}} = \overline{A_k} = \frac{\lambda_k}{\mu_k}$.

Therefore, we can define the effectiveness of a cache node (comparing the average correct sources answered with regard to an ideal cache node):

$$\epsilon = \left\{ \begin{array}{c} \text{effectiveness} \\ \text{of a cache node} \end{array} \right\} = \frac{\sum_{k \in C} \overline{R_k} f_k}{\sum_{k \in C} \overline{R_{k,\text{IDEAL}}} f_k}.$$

The effectiveness ϵ is the function we would like to maximize.

$$\max_{d_k \in \mathbb{R}^+} \left\{ \frac{\sum_{k\in C} \frac{\lambda_k}{\mu_k^2 d_k} \left[\mu_k \left(1 - e^{-f_k d_k}\right) + f_k \left(1 - e^{-\mu_k d_k}\right) - \frac{1}{d_k} \left(1 - e^{-f_k d_k}\right)\left(1 - e^{-\mu_k d_k}\right)\right].}{\sum_{k\in C} f_k \frac{\lambda_k}{\mu_k}} \right\}$$

st:

$$\beta_S \sum_{k\in C} f_k + \alpha_B \sum_{k\in C} \frac{f_k}{1 + d_k f_k} \frac{\lambda_k}{\mu_k} \leq BW_{IN} \text{ // bandwidth capacity constraints}$$

$$\alpha_S \sum_{k\in C} f_k \frac{\lambda_k}{\mu_k} + \beta_B \sum_{k\in C} \frac{f_k}{1 + d_k f_k} \leq BW_{OUT} \text{ // bandwidth capacity constraints}$$

$$d_k \in \mathbb{R}^+ \forall k \in C, \text{ //decision variables}$$

$$f_k, \lambda_k, \mu_k \in \mathbb{R}^+ \forall k \in C$$

$$\alpha_S, \alpha_B, \beta_S, \beta_B, BW_{IN}, BW_{OUT} \in \mathbb{R}^+$$

Figure 11.6: CCP Mathematical Programming Model.

11.2.2 Mathematical programming formulation

If we put together the network objective and the bandwidth restrictions discussed in the previous section, we arrive to the formulation of our CCP problem in Figure 11.6.

This is a non-linear optimization problem, both in the restrictions and in the objective function. If we study it in detail, we can see that both the feasible solution space and the objective function are convex. As the problem is stated as a maximization one, a convex objective function will in general lead to multiple local optimum.

Content class based alternative formulation. In most cases, content networks manage a very large number of different contents. These means that the previous formulation will have a large class of decision variables d_k, an additional difficulty for the numerical solution of the problem. On the other hand, for simplicity design reasons, the networks will in general treat in the same way contents that have similar characteristics. It is then possible to group all contents in a certain number of content classes, such that all contents within a class have relatively homogeneous characteristics.

Formalizing, we suppose that all contents $c \in C$ are grouped into K content classes, such that if two contents belong to the same class, all their parameters are identical:

$$C = C_1 \cup C_2 ... \cup C_K$$

$$\forall i, j \in C_k, \forall k \in [1..K] \Rightarrow \begin{cases} f_i = f_j, \\ \lambda_i = \lambda_j, \\ \mu_i = \mu_j \end{cases}$$

The size of class k, denoted by l_k, is the number of contents of this class: $\|C_k\| = l_k \forall k \in [1..K]$. The total number of contents in the network is then: $\|C\| = \sum_{k\in K} \|C_k\| = \sum_{k\in K} l_k$. We now define the Content Class Caching Problem ($CCCP$). The problem is then formalized in Figure 11.7.

$$\max_{d_k \in \mathbb{R}^+} \left\{ \frac{\sum_{k\in C} \frac{l_k \lambda_k}{\mu_k^2 d_k} \left[\mu_k \left(1-e^{-f_k d_k}\right) + f_k \left(1-e^{-\mu_k d_k}\right) - \frac{1}{d_k}\left(1-e^{-f_k d_k}\right)\left(1-e^{-\mu_k d_k}\right)\right].}{\sum_{k\in C} l_k f_k \frac{\lambda_k}{\mu_k}} \right\}$$

st:

$$\beta_S \sum_{k\in C} l_k f_k + \alpha_B \sum_{k\in C} \frac{l_k f_k}{1+d_k f_k}\frac{\lambda_k}{\mu_k} \leq BW_{IN} \text{ // bandwidth capacity constraints}$$

$$\alpha_S \sum_{k\in C} l_k f_k \frac{\lambda_k}{\mu_k} + \beta_B \sum_{k\in C} \frac{l_k f_k}{1+d_k f_k} \leq BW_{OUT} \text{ // bandwidth capacity constraints}$$

$$d_k \in \mathbb{R}^+ \forall k \in C, \text{ //decision variables}$$

$$l_k, f_k, \lambda_k, \mu_k \in \mathbb{R}^+ \forall k \in C$$

$$\alpha_S, \alpha_B, \beta_S, \beta_B, BW_{IN}, BW_{OUT} \in \mathbb{R}^+$$

Figure 11.7: $CCCP$ Mathematical Programming Model.

We enumerate the parameters of the problem (and give their dimensional units between brackets):

- l_k: number of contents belonging to class k ($[l_k] = 1$).
- f_k: query rate for class k contents ($[f_k] = \text{sec}^{-1}$).
- λ_k: rate for source arrival for class k contents ($[\lambda_k] = \text{sec}^{-1}$) .
- μ_k: rate for content deletion in sources for class k contents. ($[\mu_k] = \text{sec}^{-1}$) .
- α_S: size per location answered in response to a content query ($[\alpha_S] = \text{bytes}$).
- α_B: size per location answered in response to a backbone search ($[\alpha_B] = \text{bytes}$).
- β_S: size of a content query packet ($[\beta_S] = \text{bytes}$).
- β_B: size of a backbone search packet ($[\beta_B] = \text{bytes}$).
- BW_{IN}, BW_{OUT}: input and output bandwidth restrictions in the cache node ($[BW_{IN}] = [BW_{OUT}] = \text{bytessec}^{-1}$).
- d_k: cache expiration times for class contents ($[d_k] = \text{bytes}$)

11.2.3 The problem has non trivial solutions: a simple example

In practical situations, the solutions of CCP and $CCCP$ are not intuitive. We will shown a simple application of the $CCCP$ model. Suppose that our content network has only five different contents:

$$K = \{1..5\}, l_k = 1 \forall k \in K.$$

The content request frequency and the lodging rate are inversely proportional, see Table 11.1 for the detail of the description of the type of content that lodges the network. Usually, the backbone searches consume more bandwidth than the one needed for the clients, the Table 11.2 shows that a backbone search employs three times the bandwidth of a client request.

Table 11.1: Content lodged in the example network. Only five different contents, where the content request frequency and the lodging rate are inversely proportional.

k	l_k	f_k (1/s)	λ_k (1/s)	μ_k (1/s)
1	1	0.01	100.00	1
2	1	0.10	10.00	1
3	1	1.00	1.00	1
4	1	10.00	0.10	1
5	1	100.00	0.01	1

Table 11.2: Communication in the example network. A backbone search needs three times the bandwidth of a client request.

β_S (bytes)	1
α_S (bytes)	1
β_B (bytes)	3
α_B (bytes)	3

The problem only presents restrictions in the bandwidth consumption. It is expected that, if the bandwidth is wide enough then the cache node will not cache any request as a way to maximize the correct answers. To see this effect in the solution we considered five different available bandwidths, $BW_{IN} \in \{112, 114, 117, 123, 127\}$. Table 11.3 shows the results for each bandwidth restriction[5].

It can be observed that the solution always reaches the maximum available bandwidth BW_{IN}, except in the instance 5th, where it is possible not to cache the queries without exceeding the available bandwidth. Since we expected, as more bandwidth is available better solutions are achieved: the efficiency changes from 20% ($\epsilon = 0.198563$) to 100% (when there is not caching) with regard to a network with perfect knowledge.

The solutions are less intuitive as the bandwidth is more restrictive. This simple example shows that, in general, not choosing a good caching policy can reduce drastically the efficiency of a cache node. A common wrong assumption is that the best caching policy is to store the most requested content, another wrong possible policy it is do it with the least changeable content. For example, in the first instance (the one most compromised in available bandwidth) it is convenient to have larger expiration times in contents (Figure 11.8 following a compromise between popularity of a content and its lodging rate.

[5]As we will see later, the results are obtained using the AMPL [13] software with the MINOS [315] library solver.

Table 11.3: Results for the five instances, each ones with different bandwidth restriction.

Instance	ϵ Normalized Objective function	BW_{IN} Available bandwidth (bytes/hr)	BW_{IN} Input bandwidth employed (bytes/hr)	BW_{OUT} Output bandwidth employed (bytes/hr)	d Expiration Time for each content class
1	0.198563	112	112	93.9766	1307750, 8639810, 19096900, 245619, 0.0237167
2	0.558249	114	114	60.3430	1075230, 8898270, 0.798618, 0.292056, 0.0551847
3	0.834110	117	117	40.6829	69.0388, 6.37234, 1.16435, 0.383453, 0.0975636
4	0.997607	123	123	153.3330	0.105319, 0.103796, 0.092098, 0.0564873, 0.0137929
5	0.999999	127	126.1097	338.3000	0, 0, 0, 0, 0

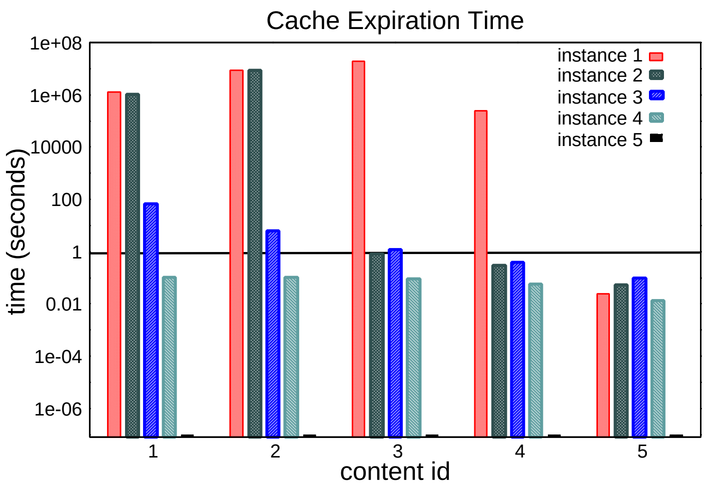


Figure 11.8: Expiration Times for the each instance, according to class of content.

11.2.4 Relationship betwen CCP and $CCCP$ models: symmetric solutions

Grouping the contents in classes allows to make the problem more treatable mathematically, because if the content is grouped in classes, then the number of variables (and the space of feasible solutions) diminishes drastically.

This implies that the $CCCP$ presents as feasible solutions only those where the cache applies the same treatment to the contents with identical parameters: $\forall i, j \in C \backslash f_i = f_j, \lambda_i = \lambda_j, \mu_i = \mu_j \Rightarrow d_i = d_j$. Whereas in the generic CCP problem it is possible to find: $i, j \in C \backslash f_i = f_j, \lambda_i = \lambda_j, \mu_i = \mu_j, d_i \neq d_j$.

This imposition of equity or symmetry in the solution of the $CCCP$ problem implies that some good solutions of the generic problem are ignored. We show it in the following simple example.

11.2.4.1 An Example of Loss of Solutions.

Let's suppose a network with two contents $C = \{c_1, c_2\}$, which belong to the same class $K = \{1\}$, with parameters: $l_1 = 2$, $f_1 = \lambda_1 = \mu_1 = 1$, $\alpha_S = \alpha_B = \beta_S = \beta_B = 1$, $BW_{IN} \geq BW_{OUT} = 2.25$.

Then, the CCP problem is defined as:

$$\begin{array}{ll} \max_{d_k \in \mathbb{R}^+} & \left\{ \left[\frac{1}{d_1} \left(1 - -e^{-d_1}\right) - \frac{1}{2d_1^2} \left(1 - -e^{-d_1}\right)^2 \right] + \left[\frac{1}{d_2} \left(1 - -e^{-d_2}\right) - \frac{1}{2d_2^2} \left(1 - -e^{-d_2}\right)^2 \right] \right\} \\ \text{st:} & \\ & 2 + \frac{1}{1+d_1} + \frac{1}{1+d_2} \leq BW_{OUT}, \text{ // bandwidth capacity constraints} \\ & d_1, d_2 \in \mathbb{R}^+ \text{ //decision variables} \end{array}$$

And the $CCCP$ problem is defined as:

$$\begin{array}{ll} \max_{d_1 \in \mathbb{R}^+} & \left\{ \frac{2}{d_1} \left(1 - -e^{-d_1}\right) - \frac{1}{d_1^2} \left(1 - -e^{-d_1}\right)^2 \right\} \\ \text{st:} & \\ & 2 + \frac{2}{1+d_1} \leq BW_{OUT}, \text{ // bandwidth capacity constrain} \\ & d_1 \in \mathbb{R}^+ \text{ //decision variable} \end{array}$$

The objective function of the $CCCP$ problem is decreasing monotonously, whereas the bandwidth restriction is increasing monotonously, therefore the optimal solution is reached when the cache node is consuming its maximum available bandwidth: $d_1 = 7$ and $\epsilon = 0.265083$. On the other hand, the CCP problem presents better performance if asymmetric solutions are considered, for example: $d_1 = 3.4032$, $d_2 = 42.6857$ and $\epsilon = 0.266874$.

11.2.5 Compromise between Publication and Search

In the chapter introduction (Section 11.1.2) we present the compromise between publication and search. In the cache nodes, this commitment is expressed by the time that the query answers are stored; i.e. the time of caching for any content k, called d_k. Querying nodes are connected to at least one aggregation node in order to route their queries. The aggregation node concentrates all queries of its connected nodes and consults the backbone when it is not able to directly answer the queries with its cache of the results of recent queries (i.e. expiration time of this content). The previous results of a content k are systematically deleted after a period d_k, because the nodes which publish this content can disconnect or delete it. Therefore, the optimal expiration time d_k depends of the dynamics in the lodged content process and the query frequency. If the lodgment of a content k is very dynamic, it will be expected that it will not be efficient to cache the answers (because they become out dated quickly), this implies a lower d_k. In the opposite situation, when a content is heavily queried, it will be expected that its queries will be cached a longer time (i.e. a high d_k). This is the compromise between publication and search, and we will show it in our CCP model.

High expiration time: If the expiration time d_k of a content grows, then the average number of correct answers diminishes to the point of which the cache node does not return any valid answer: $\lim_{d_k \to +\infty} \overline{R_k} = 0 \;\; \forall k \in C$.

Therefore, if an infinite expiration time is supposed for all the contents, the cache node does not answer correctly any valid location, using a minimal bandwidth:

$$\lim_{d_k \to +\infty} \epsilon = \inf_{d_k \in \mathbb{R}^+} \{\epsilon\} = 0 \;\; \forall k \in C$$

$$\lim_{d_k \to +\infty} \{\text{bandwidth in}\} = \inf_{d_k \in \mathbb{R}^+} \{\text{bandwidth in}\} = \beta_S \sum_{k \in C} f_k \;\; \forall k \in C$$

$$\lim_{d_k \to +\infty} \{\text{bandwidth out}\} = \inf_{d_k \in \mathbb{R}^+} \{\text{bandwidth out}\} = \alpha_S \sum_{k \in C} f_k \overline{A_k} \;\; \forall k \in C$$

Low expiration time: On the other hand, if the expiration time d_k of a content diminishes to very low values, the cache node answers perfectly all the requests (i.e. it answers all the sources that lodges the content): $\lim_{d_k \to 0^+} \overline{R_k} = \overline{A_k} = \frac{\lambda_k}{\mu_k} \;\; \forall k \in C$.

Therefore, with an instantaneous expiration time for all the contents, the cache node answers perfectly all the valid content locations, using the maximum available bandwidth:

$$\lim_{d_k \to 0^+} \epsilon = \sup_{d_k \in \mathbb{R}^+} \{\epsilon\} = 1 \;\; \forall k \in C$$

$$\lim_{d_k \to 0^+} \{\text{bandwidth in}\} = \sup_{d_k \in \mathbb{R}^+} \{\text{bandwidth in}\} = \beta_S \sum_{k \in C} f_k + \alpha_B \sum_{k \in C} f_k \overline{A_k} \;\; \forall k \in C$$

$$\lim_{d_k \to 0^+} \{\text{bandwidth out}\} = \sup_{d_k \in \mathbb{R}^+} \{\text{bandwidth out}\} = \alpha_S \sum_{k \in C} f_k \overline{A_k} + \beta_B \sum_{k \in C} f_k \;\; \forall k \in C$$

According to the dynamism that presents the content locations in the sources (expressed by λ_k and μ_k in our model) and the frequency of requests (f_k) we fix the optimal expiration time (d_k) in the cache node, restricted by the available bandwidth.

The same conclusions can be applied to the $CCCP$ model.

11.3 On the Application of the Efficient Search

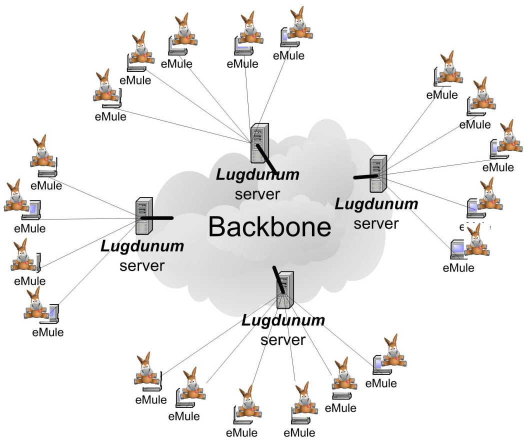
11.3.1 Content Networks

The Content Class Caching Problem ($CCCP$) (presented on previous section 11.2) can be applied to a wide range of Content Networks. In our previous work [274, 276, 277, 279] we present a taxonomy of Content Networks, based on their architectures. Some of the characteristics studied were the decentralization of the network, the content aggregation and the content placement. Also we analyze their behaviors, in terms of performance and scalability. Depending on the specific application of the content network, and mainly depending on the content dynamics in the network, different architectures design are used. Actually, the presence of cache nodes in content networks is apparently mandatory because of scalability reasons. Figure 11.9 shows a simplified view of well know content networks. New file-sharing P2P networks have cache nodes to improve their searching performance. In unstructured P2P networks with a hierarchical decentralization, usually the most compromised nodes have query caching facilities. For instance, eMule [89] (Figure 11.9(a)) has dedicated servers (typically with the Lugdunum software), which concentrate the queries of any peer connected to them. In eMule, each peer connects to a unique server, who will propagates the peer' queries in the network and return the consolidated answers to the peer. In KaZaA [190] (Figure 11.9(b)), the concept is identical, with the expection that the servers are also peers, called superpeers. For redundancy reasons, in KaZaA, each peer connects in average to three superpeers simultaneously (for simplification, not shown in the figure). Hybrid P2P systems, like Napster [230] (Figure 11.9(c)), does not need the query caching facility, because the central server has a global view of the content in the network. In pure peer-to-peer networks, like Gnutella [115] (Figure 11.9(d)) and Freenet, there are no cache node because all nodes have equal roles. Some P2P protocols, for example BitTorrent [30], does not provide a mechanism for content discovery, and therefore no query caching is done (in these networks, the content discovery is done outside the network, typically in the Web). The cache nodes are not used exclusively in peer-to-peer networks, for instance, they are present in the Domain Name System (DNS) [39, 143, 144, 242] (Figure 11.9(e)), where they are called recursive servers. Next we will present other work related to our $CCCP$ model.

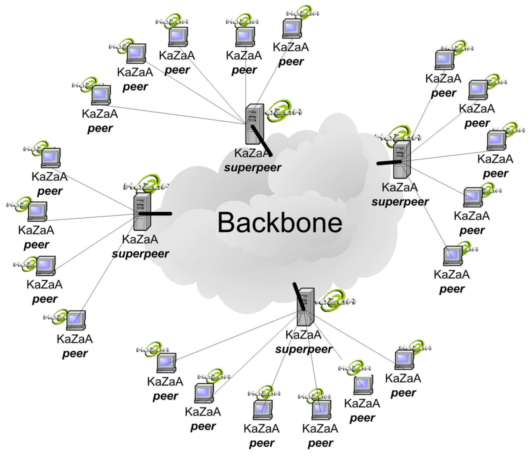
11.3.2 Other Approaches: Expiration Time in Cache Nodes

There are a lot of works related to the estimation of expiration times in the cache nodes of a content network. In general their objectives are to reduce the latency and the traffic in the delivery of the content to the final user. In this section we summarize the work related in this aspect.

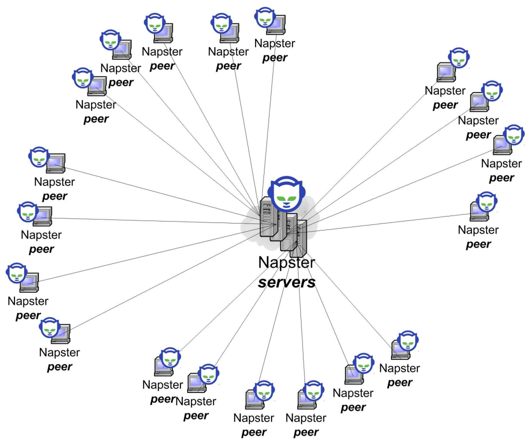
First, it is important to highlight that the cache technology comes out its application of the content networks. It is used for the memory access and disc access in the computers [22, 337], in the file systems (distributed or not) [5, 76, 232], in the routing protocols, in the distributed databases [241, 301], in mobile networks [44, 61], in the content networks (for the queries [29, 205, 208, 212, 297] or in the information lodging [95, 114, 132]), etc. According to each application, different models and technologies have been developed, we summarize in this section those ones related to content networks.



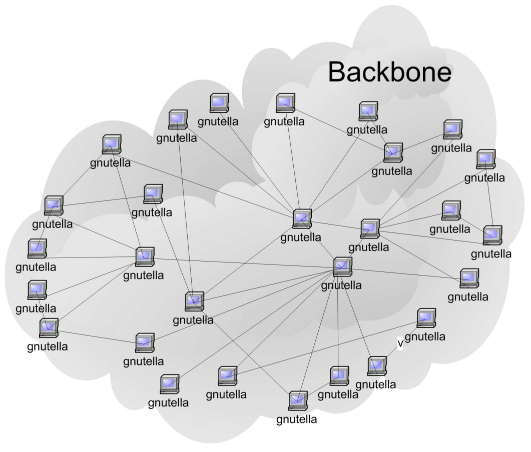
(a) eMule.



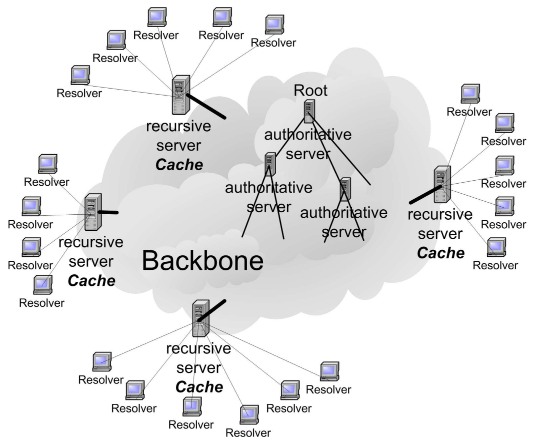
(b) KazAa.



(c) Napster.



(d) Gnutella.



(e) DNS.

Figure 11.9: Simplified view of well know content networks.

11.3.2.1 Consistency of the Information.

In many content networks the information consistency is crucial. In the cache nodes the inconsistency happens when they answer an invalid information, because the source has deleted the content (or has disconnected).

For example, in the DNS system it is extremely important to assure that the answers to the mapping among domain names and IPs should be correct, in another case there would arise some problems of safety (we will see later on that actually the DNS does not have the whole consistency wished).

11.3.2.2 Time-to-Live (TTL) Method.

When the information consistency is necessary, the network designer must include in the protocols some mechanisms to ensure it. If there are cache nodes, then the most used method is the TTL (time to live).

Each content is published by the sources with an associated time to live, the TTL. The TTL determines the maximum time in which a source assures that the content is going to remain unchanged and therefore it is the maximum expiration time allowed in the caches. In the first request of a content, the cache node obtains the answer of some source together with a TTL, for the following queries included in the interval TTL the cache can answer this source without consulting it again, after this period the cache node must erase the content of the cache and consult to the source again for the content.

Several networks use TTL's method, for example the World Wide Web and the DNS system. The cache nodes in the Web they are known as proxies Web, whereas in the DNS's network are known as recursive servers.

World Wide Web - Proxies Web. Most of the effort made in related works is in the study of the proxies cache for the WWW [4, 38, 53, 62, 65, 66, 124, 195–197, 217, 307, 340, 348, 350]. The TTL mechanism assures partially the information consistency in the HTTP caches, in general the previous studies on this topic are in two possible directions:

- To choose efficiently the contents to lodge in the cache, supposing a restriction in capacity, and
- to define policies (or algorithms) to lodge efficiently the contents (for example to choose the expiration times).

With respect to the contents to lodge in the cache, several techniques of prefetching have been proposed and proved [65, 66, 124, 340, 348, 350].

With respect to the policies definition, there are several works that test in a simulation context or with real data different algorithms. Some of these works are similar to our approach because they present optimization models who consider the dynamics of the content in the sources:

- Xing Y., in his MSc. work *"Caching on the Changing Web"* [355] presents an optimization model of the cache on proxies Web. The function to maximize is the benefit achieved by the decrease of latency in the final users and the cost is the space of available lodging in the cache. In his analysis, he classifies the contents in four classes depending on the request frequency and the lodging ratio.

- Gopalan P. et. al. in the work *"Caching with Expiration Times"* [117] show that some complex policies do not differ in performance with regard to other simple policies.
- Shim J. et. al. [307]. Their approach is very similar to Xing's work , they do not divide the contents in classes and they study other algorithms for the cache policies.
- Krishnamurthy B. et. al. [195–197] analyze different algorithms to estimate the expiration time in the caches depending on the query rate.
- Chen X. et. al. [53] study the expiration time of Web content. They classify the contents in four classes also; and they present a new algorithm to the expiration policy.

Domain Name System - Recursive servers. In spite of its importance, there are not so many works that study the cache performance in a recursive server. The caching policy must differ from other content networks (as the World Wide Web) given the nature of the content that is lodged:

- the contents have very small size and it is not a problem to lodge them;
- the search in the backbone (when the answer is not cached) cache has a high cost of latency in comparison to other systems;
- the expiration time is much less than the time between changes in the content;
- the content is generally lodged in redundant sources, and the access time differs among the sources;
- there are many authoritative servers badly configured (for example with LAME delegations), causing excessive delays in the search with time-out's.

The recursive servers incorporate a passive policy in the cache managing. The content is stored exclusively after a query node requests it, and is stored strictly according to the TTL offered by the source. Studies on this topic analyze improvements in different aspects [6]:

- to define policies (or algorithms) to lodge efficiently the contents (in some cases: to choose the expiration time);
- and to correlate the searches in DNS to the use of other networks (specially the WWW).

Most of the works tackle the definition of policies to store efficiently the queries in the recursive servers. E. Cohen and H. Kaplan seem to be very active in this area, developing studies in the DNS system [64] and in the World Wide Web [62, 63, 66]. The first improvement

[6]Since the small size that has the content in the DNS network, it is not necessary to choose efficiently which content will be stored in the cache, like the studies in WWW.

that appears is the pre-fetching search (or renovation policy): for very requested contents the contents of the cache refresh themselves generating unrequested searches, this eliminates the latency in the first resolution of a content.

Other improvements are done in the manipulation (in general the reduction) of the TTL stored in the cache. J. Jung et. al. model the behavior of the TTL mechanism [188] and its application to define policies on the DNS cache [189]. In general, the conclusion of their studies is that the influence of the policies of TTL managing are minimal and has no impact in the speed of navigation perceived by the final users [310].

Some authors discuss the utilization of joint policies between the World Wide Web and the DNS, to reduce the latency perceived by the final users. The underlying idea is to look for the cooperation and synchronism between DNS's caches and Web navigation. The pre-fetching needs predictive information on the future behavior of the query nodes. In general the caches use statistical information, supposing regularity among users (for example it is supposed that a content very requested previously will be very requested in the future). Nevertheless more knowledge on the applications that generate the queries in DNS provides a more refined prediction. A. S. Hughes and J. Touch study the traces of a proxy Web to predict the future navigations of the clients and they use this information to pre-fetch in the DNS [138, 139]. E. Cohen and H. Kaplan also study this idea, they analyze the pre-resolution of names [65], and also they study the simultaneous validation between the Web content and the DNS answers, achieving good results in small environments, when there are slow dynamics in the DNS sources [64].

On The strict Use of TTL In most of the networks that implement TTL mechanism, the TTL is not respected strictly. The clients of a network that offers a TTL mechanism often do not respect the cache time and keep the information as valid after its TTL expired. This behavior introduces serious disadvantages from the point of view of the manager of the network, but some advantages from the point of view of the final user: a strictly use of the TTL times is a restriction on the generic problem of diminishing the delays and the traffic in the delivery of the content to the final user, specially when there are slow dynamics in the content changes.

For example in the World Wide Web, all the Internet browsers (Microsoft Internet Explorer[7], Mozilla-Netscape[8] , etc.) do not respect the expiration time received in every HTTP connection.

These products are in competition to offer the most quick browsing experience, this is incompatible with a strict use of the standard in respect to the TTL, because it would generate more requests to the sources and therefore more delay in the display of the web pages. The problem arises in the fact that the TTL offered by the sources rarely imply a change in the content. If whenever the TTL is expired the source makes a change in the content unfailingly the Web browsers should respect the standard because in another case they will display wrong information, something that the users would perceive wrongly.

From the content sources point of view, in general they can not anticipate when they will update their contents, therefore the common behavior is to use the average time between changes for the TTL. This limitation is so clear that we do not know Web server implementations, or

[7]Internet Explorer. Home Page. http://www.microsoft.com/windows/ie/
[8]Mozilla Home Page. http://www.mozilla.org/

DNS authoritative server implementations, that can schedule future changes and reduce gradually the TTL to exactly inform to the cache nodes of the next change. Only few administrators of these systems know this level of detail and therefore in general the default TTL is used. Therefore, in practice, the TTL does not specify the minimal time of unchanged content, it just reports the average time between changes.

The recursive servers implementations and proxies Web implementations that belong to the public domain use the TTL mechanism strictly (for example BIND [229] and Squid [313]), nevertheless some proprietary implementations do not use it arguing improvements in the performance.

11.3.2.3 Other networks and methods - Peer-to-Peer networks.

The P2P file sharing networks do not use the TTL's mechanism, and its application would be inefficient because the indeterminate dynamism in the lodged content.

Some applications use a P2P infrastructure to offer a decentralized and anonymous proxy Web (for example Squirrel [179] that uses Pastry [287] as network infrastructure).

The P2P traffic represents an important percentage of the bandwidth consumption in the international links of every ISP. These links in general are costly and the growth in their usage is a worry for the ISP.

Several companies offer equipment that guarantees a reduction of the bandwidth consumption in the international links due to P2P applications. This equipment stores and guides the traffic of the most popular P2P applications in order to use efficiently the link. These equipments are cache nodes in these networks, with a particular policy in respect to the expiration time of contents out side the ISP network. Examples of these equipments are: PacketShaper[9], Peer-to-Peer-Element[10] and PeerCache[11].

11.3.2.4 CCP, $CCCP$ models and the TTL mechanism.

The CCP and $CCCP$ models presented in this chapter do not contemplate the TTL mechanism that several content networks include to preserve the information consistency. The TTL can be added to the CCP problem directly if we add the restriction in the expiration time of every content. The same directly extension is possible in the $CCCP$ problem, where every content in a class has the same TTL. It is potentially a very strong restriction, because the TTL of a class has to be the minimal TTL among the contents of the class. Therefore, in this case it is necessary to study in depth the impact of the TTL restriction on the solution.

It is necessary a comparative study of the the information consistency between the current networks with TTL and those who arise of the use of the CCP and $CCCP$ models.

In the following section, we apply the models to two real network cases: a file sharing P2P network (without TTL's mechanism), and a DNS recursive server (that includes TTL). We expect a similar behavior in the workload of the P2P network with respect to the Video on

[9]Packeteer's PacketShaper® Home Page http://www.packeteer.com/products/packetshaper.cfm.
[10]Sandvine Incorporated. Home Page. http://www.sandvine.com/.
[11]PeerCache de Joltid. Home Page. http://www.joltid.com/index.php/peercache.

Demand service of our GOL!P2P prototype. Also we expect a similar behavior between the DNS study and the MyTV service.

11.4 Estimate the Performance Using "Similar" Real Data

11.4.1 Caching VoD searches

In this section we present a numerical illustration of the $CCCP$ approach over a case study, where the data was generated with information available in different literature sources especially referring to Gnutella or similar peer-to-peer (P2P) file sharing networks [57, 123, 211, 300, 359]. We have chosen file-sharing P2P networks, because we expect similar user behavior for the VoD service, and also because there is also much quantitative information available for P2P networks.

Table 11.4: Parameter values for a file sharing case study (a similar behavior is expected in VoD).

Parameter	Value
T: time units	1 hour
C: number of different contents	878691
$\overline{f}$: average content query rate	0.037938251 hr-1
f_{max}: maximum content query rate	1000 hr-1
$\overline{\lambda}$: average content storage rate	11.09749966 hr-1
$\overline{\mu}$: average content location validity rate	1 hr-1
$\left(\frac{\lambda}{\mu}\right)_{max}$: maximum allowed number of locations answered in response to a content query	200
α_S: size of a the answer to a content query	100 bytes
α_B: size of the answer of a backbone search	310 bytes
β_S: size of a content query	94 bytes
β_B: size of a backbone search packet	291.4 bytes
BW_{IN}: input bandwidth	921600000 bytes/hr.
BW_{OUT}: output bandwidth	460800000 bytes/hr.

Table 11.4 summarizes the main parameters of the case study. We generated ten CCP instances of this detailed case study (using a random number generator with different seeds), including the data for the 878691 different contents (which correspond to the number of Gnutella contents in the study by Chu [57]), where the distributions for the query frequency follow a modified Pareto distribution law taking into account the "fetch-at-most-once" effect (see [123] for a discussion of this observed network behavior). Regarding the frequency of arrival of new storage locations for each content, we suppose that it is linearly related to the query frequency, following the hypothesis mostly used in the literature (an exception is the work by Qin [211] which also studies a square root dependency). For the bandwidth constraints, we suppose that the cache nodes will be equipped with an ADSL 2/1 Mbps connection as reference value. The average packet sizes in Gnutella was measured by Yang [359].

11.4.1.1 Number of content classes in the $CCCP$.

As we discussed in section 11.2.2, it is next to impossible to directly solve the CCP problem generated, a non-linear problem in 878691 independent variables (one for each content). As an alternative, we cluster the contents into a small number of homogeneous content classes, and solve the resulting $CCCP$ problem. As it is not a-priori clear what is the best number of classes to use, we experimented with five different values, namely 2, 8, 16, 32, and 128 classes, for each of the ten different CCP problems generated.

The problem was programmed using the AMPL modeling language; AMPL [13, 263] is a software with a algebraic modelling language for doing mathematical programming, which can be used to easily represent a non-linear mathematical programming problem such as the $CCCP$. We solved the problem using AMPL in conjunction with MINOS [226, 315] (version 5.5), an optimization solver[12].

All experiments were run on a PIII 800 MHz computer, with 320 Mb RAM space. The ten instances results obtained per class are averaged in Table 11.5. Among other observations, we can see that when the number of classes grow, the available resources (i.e. the BW_{OUT} employed) are being increasingly used. Also, the computational times required to solve the model grow, albeit they remain very modest.

Table 11.5: Average results for 10 (randomly generated) cases of a file sharing case study (a similar behavior is expected in VoD).

Number of content classes	Normalized Objective function ϵ	Execution time (secs.)	Input bandwidth employed (bytes/hr)	Output bandwidth employed (bytes/hr)
2	0.99888275	0.00625	921599933	334157325
8	0.98845330	0.07400	921599993	406282138
16	0.98759660	0.25800	921599993	413490851
32	0.99659690	0.21100	921599993	415901600
128	0.99924110	0.44900	921600793	416588930

As in $CCCP$ we are dealing with aggregated data, For comparing reasons, it is important to translate back the results into the terms of the original CCP problem. In particular, we now consider again the 878691 different contents, and we evaluate the number of correct answers to queries if we use for each content the cache expiration times given by the optimization models. Table 11.6 summarizes this comparison. From this table, we can see that if the number of classes is too low, then the approximation error incurred in the aggregated model is very large, and the percentage of correct answers to queries in the real problem will be much below the nominal values computed by the optimization procedure. This discrepancy gets very quickly irrelevant when the number of classes increase, when we have 128 classes the results coincide.

Each content network has its own profile in respect to the query frequency and store dynamics. This profile influences on the degree of similarity of the contents in the network, from our model point of view, and therefore how many $CCCP$ classes are needed to compute with a negligible approximation error. In the P2P instances evaluated, with 32 classes or 128 classes,

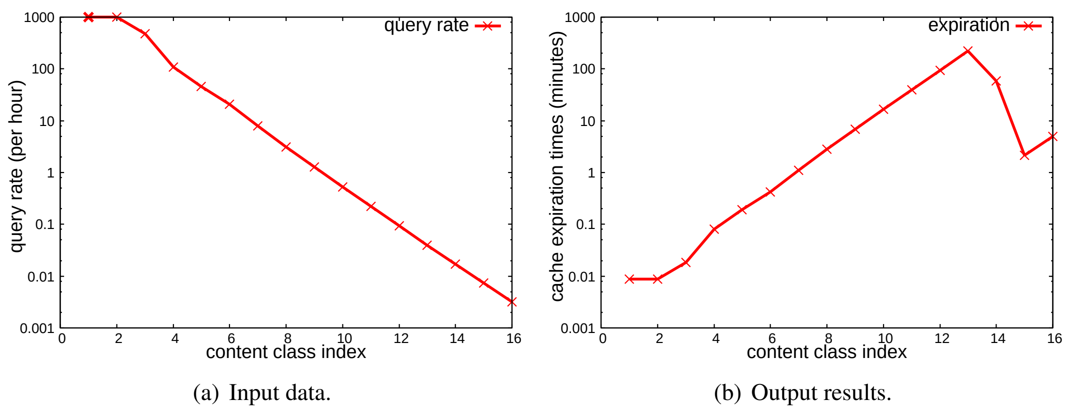
[12]Other optimization solvers were tested, see [274] for details.

Table 11.6: Discrepancies between objective functions for original and aggregated models in the P2P case study (a similar behavior is expected in VoD).

Number of classes	Normalized objective function ϵ for the aggregated $CCCP$ problem	Normalized objective function ϵ for the original CCP problem
2	0.99888275	0.86598051
8	0.98845330	0.98248558
16	0.98759660	0.98621038
32	0.99659690	0.99653966
128	0.99924110	0.99924080

we have very good precision and low computation time.

We have also looked in detail at the solutions given by the optimization model. As a representative case, we can look at the results of one of the instances of the 16 class $CCCP$ model. In Figure 11.10, we can see on the left the distribution (in logarithmic scale) of the query rates for the different content classes; the difference between query rates go across 6 magnitude orders. On the left, we can see (also in logarithmic scale) the results of the optimization, namely the values of the cache expiration dates for each of the 16 content classes. It is clear that, although here we can also appreciate wide differences in scale, there is no direct relation with the input data shown on the left.



(a) Input data. (b) Output results.

Figure 11.10: Input data and Output results for a 16 class $CCCP$ instance.

11.4.1.2 Comparing our approach with other cache expiration dates policies and with other optimization solvers

In order to analyze the $CCCP$ performance, we model different policies for fixing the cache expiration dates for the information about the contents' location. In particular, we give mathematical programming formulations which represent the case where the cache expiration dates are equal for all contents, the case where the cache expiration dates are proportional to the

$$\max_{d\in\mathbb{R}+}\left\{\frac{\sum_{k\in C}\frac{l_k\lambda_k}{\mu_k^2 d}\left[\mu_k\left(1-e^{-f_k d}\right)+f_k\left(1-e^{-\mu_k d}\right)-\frac{1}{d}\left(1-e^{-f_k d}\right)\left(1-e^{-\mu_k d}\right)\right].}{\sum_{k\in C} l_k f_k \frac{\lambda_k}{\mu_k}}\right\}$$

st:

$$\beta_S\sum_{k\in C} l_k f_k + \alpha_B \sum_{k\in C}\frac{l_k f_k}{1+df_k}\frac{\lambda_k}{\mu_k} \le BW_{IN} \text{// bandwidth capacity constraints}$$

$$\alpha_S\sum_{k\in C} l_k f_k \frac{\lambda_k}{\mu_k} + \beta_B \sum_{k\in C}\frac{l_k f_k}{1+df_k} \le BW_{OUT} \text{// bandwidth capacity constraints}$$

$$d \in \mathbb{R}^+ \forall k \in C, \text{//decision variable}$$

$$l_k, f_k, \lambda_k, \mu_k \in \mathbb{R}^+ \forall k \in C$$

$$\alpha_S, \alpha_B, \beta_S, \beta_B, BW_{IN}, BW_{OUT} \in \mathbb{R}^+$$

Figure 11.11: $SETP$ Mathematical Programming Model.

query frequency for a content, and the case where all cache expiration dates can be fixed independently, i.e. the $CCCP$.

Policies for fixing Cache expiration dates. Based on the previous model, we now discuss three alternative ways to fix the cache expiration dates.

- **Single expiration time policy (SETP).** The simplest option for solving the problem is setting the same expiration time for all contents. In this case, the aggregation node defines a fixed expiration time d; for every content location query, the information is stored in the cache during this time, and then deleted. This amounts to having all variables $d_k = d$ in the previous formula, leading to the mathematical programming formulation in Figure 11.11.

- **Expiration time proportional to query rates policy (PETP).** The query rate has an important impact in the behavior of the different contents. It can be reasonable to assume that high query rates correspond to contents with high impact and that their expiration times must then be kept longer; the easiest way is to impose a linear dependency. In this case, there is a single coefficient e such that for every content k, $d_k = ef_k$. Then, the mathematical programming formulation is shown in Figure 11.12.

- **Optimal policy ($CCCP$).** If all values d_k are free, unrelated variables, solving the optimization model will give the theoretical optimum for the cache expiration policy problem. The $CCCP$ formulation is then the most general one.

The three problems are non-linear optimization problems, both in the restrictions and in the objective function. We can see that both the feasible solution space and the objective function are convex. As the problem is stated as a maximization one, a convex objective function will in general lead to multiple local optimum.

$$\max_{e\in\mathbb{R}^+}\left\{\frac{\sum_{k\in C}\frac{l_k\lambda_k}{e\mu_k^2 f_k}\left[\mu_k\left(1-e^{-ef_k^2}\right)+f_k\left(1-e^{-e\mu_k f_k}\right)-\frac{1}{ef_k}\left(1-e^{-ef_k^2}\right)\left(1-e^{-e\mu_k f_k}\right)\right].}{\sum_{k\in C} l_k f_k \frac{\lambda_k}{\mu_k}}\right\}$$

st:

$$\beta_S \sum_{k\in C} l_k f_k + \alpha_B \sum_{k\in C} \frac{l_k f_k}{1+ef_k^2}\frac{\lambda_k}{\mu_k} \leq BW_{IN} \text{// bandwidth capacity constraints}$$

$$\alpha_S \sum_{k\in C} l_k f_k \frac{\lambda_k}{\mu_k} + \beta_B \sum_{k\in C} \frac{l_k f_k}{1+ef_k^2} \leq BW_{OUT} \text{// bandwidth capacity constraints}$$

$$e \in \mathbb{R}^+ \forall k \in C, \text{//decision variable}$$

$$l_k, f_k, \lambda_k, \mu_k \in \mathbb{R}^+ \forall k \in C$$

$$\alpha_S, \alpha_B, \beta_S, \beta_B, BW_{IN}, BW_{OUT} \in \mathbb{R}^+$$

Figure 11.12: $PETP$ Mathematical Programming Model.

Numerical Illustration. In this section we present a numerical illustration over a case study. The instance we discuss here was generated using the parameters of the Table 11.4. We solved the three different problems formulated using AMPL software with the different solvers available at NEOS [73, 122]. The particular optimization solver used are MINOS, IPOPT [165], KNITRO [365], LANCELOT [201], PENNON [254], and SNOPT [315]. The better performance was achieved using the SNOPT solver.

We compare the three optimization policies above described. As the full problem generated has a large number of contents (878691), we used the 16 content classes instance, and we solve this reduced problem. The results are then cast back in terms of the original problem[13].

Table 11.7: Problem solutions, for 16 content classes instance, with the three policies: $CCCP$, $SETP$ and $PETP$.

	$CCCP$		$SETP$		$PETP$	
Optimization Solver	Objective Function ϵ	Execution Time (s)	Objective Function ϵ	Execution Time (s)	Objective Function ϵ	Execution Time (s)
MINOS	0.99610320	0.520	0.99995210	0.020	0.99993000	0.020
IPOPT	0.99995470	0.030	0.99995210	0.040	0.99992800	0.040
KNITRO	0.99995310	0.640	0.99995190	0.140	0.99658890	0.050
LANCELOT	0.99975880	0.180	0.99995210	0.010	0.01368548	0.030
PENNON	0.99955790	0.130	0.99995210	0.030	0.99993000	0.030
SNOPT	0.99995470	0.010	0.99995210	0.030	0.99993000	0.030

Table 11.7 shows the objective values and the computing times for the three policies, in the 16 class problem. The execution times are short, even if it can be seen that the $CCCP$ policy takes longest to compute to optimality. Table 11.8 shows the objective function values, as computed in the original CCP case. As expected, it can be seen that the $CCCP$ policy

[13]The original CCP problem could not be solved directly on NEOS due to its file size limitations in the used optimization interfaces.

Table 11.8: Solutions transformed to the original problem. For the policies: $CCCP$, $SETP$ and $PETP$.

Optimization Solver	$CCCP$ Aggregated Objective Function ϵ	$CCCP$ Original Objective Function ϵ	$SETP$ Aggregated Objective Function ϵ	$SETP$ Original Objective Function ϵ	$PETP$ Aggregated Objective Function ϵ	$PETP$ Original Objective Function ϵ
MINOS	0.99610320	0.99553881	0.99995210	0.99995200	0.99993000	0.99993040
IPOPT	0.99995470	0.99995404	0.99995210	0.99995200	0.99992800	0.99992847
KNITRO	0.99995310	0.99995215	0.99995190	0.99995174	0.99658890	0.99661043
LANCELOT	0.99975880	0.99974659	0.99995210	0.99995200	0.01368548	0.01393887
PENNON	0.99955790	0.99953126	0.99995210	0.99995199	0.99993000	0.99993039
SNOPT	0.99995470	0.99995405	0.99995210	0.99995200	0.99993000	0.99993040

obtained the best results, with $\epsilon = 0.99995405$ for the original problem. All the same, the other policies resulted in values not far away from this optimum.

11.4.2 Caching MyTV searches

In order to apply our framework to the MyTV service, it is necessary to obtain from measurements or previous studies the information about the parameters of the class contents. For it, we look at the DNS (Domain Name System) [39, 143, 144, 242], which is the system used on Internet in order to map symbolic domain names (such as www.fing.edu.uy) into IP addresses corresponding to actual computers in the network (such as 164.73.32.3). We expect a similar behavior in the query frequency of the DNS and the MyTV service. The network actually modeled corresponds to the *.uy* (Uruguay) subdomain of Internet, and the parameters used are based on real data obtained thanks to the support of ANTEL[14], which is a state-owned company, and the largest telco in Uruguay.

11.4.2.1 Parameters for the DNS network.

DNS (Domain Name System) can be seen as a content network with hierarchical distribution, where the information consistency is one of the most important objectives. DNS is based on a network of recursive servers, which pass on queries until finding authoritative answers, which minimize the probability of information inconsistency (even if the correctness cannot be completely guaranteed). In our case, we are interested in recursive servers, which correspond to the aggregation nodes of our general model. We collected real data from the recursive servers at ANTEL, which daily serve hundreds of thousands users, with daily peak query rates of approximately 1800 queries per second. The parameters of the model are summarized in Table 11.9.

As previously discussed, in our study we only took into account the behavior of the Uruguayan domains, i.e., those whose names finish by *.uy*. We collected ten consecutive days of recursive server logs, in order to estimate the total number of contents and the distribution of query rates. ANTEL DNS infrastructure employs the BIND software[15], which was very useful as the logs

[14]ANTEL - Administración Nacional de Telecomunicaciones, http://www.antel.com.uy.

[15]Berkeley Internet name domain, by Internet Software Consortium, http://www.isc.org/products/BIND.

Table 11.9: Parameter values for a domain name system case study (a similar behavior is expected in MyTV).

Parameter	Value
C: number of different contents	220107
$\overline{f}$: average content query rate	Empirical distribution (heavy tailed, see Figure 11.13)
$\overline{\lambda}$: average content storage rate	0.8361 hr-1
$\overline{\mu}$: average content location validity rate	0.5158hr-1
α_S: size of a the answer to a content query	169.6 bytes
α_B: size of the answer of a backbone search	1150.3 bytes
β_S: size of a content query	80.45 bytes
β_B: size of a backbone search packet	385.6 bytes
BW_{IN}: input bandwidth	1.933×10^8 bytes/hr.
BW_{OUT}: output bandwidth	3.445×10^8 bytes/hr.

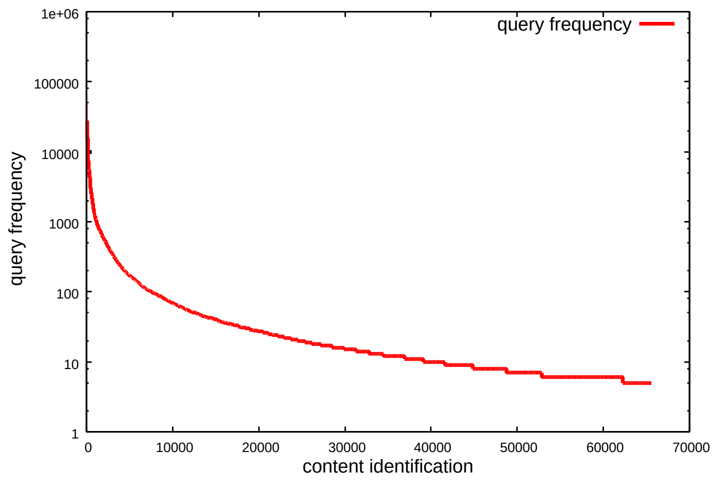


Figure 11.13: Domain query rates distribution (tail cut off).

it collects contain very detailed information. The log files were processed using a BerkeleyDB data base[16] to obtain the statistics of the domain queries. The largest part of the *.uy* domains actually belongs to the *.com.uy* zone, which is administered by ANTEL. We took the historical information of the domain changes between october 2003 and october 2005; this information was used to compute overall storage and validity rates for the contents. To compute the mean packet sizes, we also collected information about the packets transmitted and received by a DNS recursive server (in this case, a 15 minutes detailed sample provided us with enough information). Table 11.10 shows the measurements, which were the basis for computing the $\alpha's$ and $\beta's$ parameters shown on Table 11.9.

Table 11.10: Statistics for DNS packet sizes.

	Number	Total bytes transmitted	Number of registers
Queries	366148	29458160	402098
Answer to queries	168530	40089683	236386
Backbone searches	61481	5594012	293552
Answers to backbone searches	56442	10774381	293552

Finally, in the case of ANTEL recursive servers, the bandwidth limitation is actually driven by the CPU processing power, which limits the number of queries that may be processed by time unit. In our case, the empirical measurements result in $BW_{IN} = 419.5\text{Kbps} = 193305600\text{bytes/hr}$ and $BW_{OUT} = 747.6\text{Kbps} = 344494080\text{bytes/hr}$.

11.4.2.2 Numerical results.

The data collection discussed in the previous section resulted in obtaining detailed query rate information for each of the 220107 contents observed. We cluster the contents into a small number of content classes, where in each class we will include contents with equal or at least similar query rates. As it is not a-priori clear what is the best number of classes to use, we experimented with five different values, namely 2, 8, 16, 32, and 128 classes. In order to solve the different problems formulated, we used AMPL in conjunction with MINOS. All experiments were run on a PIII 800 MHz computer, with 320 Mb RAM space. The results obtained are summarized in Table 11.11. As in the VoD service study, when the number of classes grows, the available bandwidth are being increasingly used. Also, the computational times required to solve the model grow, albeit they remain very modest.

We continue our analysis, as in the VoD service, translating back the results, of the aggregated $CCCP$ data, into the terms of the original CCP problem. In particular, we now consider again the 220107 different contents, and we evaluate the number of correct answers to queries if we use for each content the cache expiration times given by the optimization models. Table 11.12 summarizes this comparison.

In this table, we saw the same behavior than the VoD service case. If the number of classes is too low, then the approximation error incurred in the aggregated model is very large, and the percentage of correct answers to queries in the real problem will be much below the nominal

[16]Sleepycat Software Inc.. http://www.sleepycat.com/.

Table 11.11: Results for different number of content classes of the DNS case study (a similar behavior is expected in MyTV).

Number of content classes	Normalized Objective function ϵ	Execution time (secs.)	Input bandwidth employed (bytes/hr)	Output bandwidth employed (bytes/hr)
2	0. 969623	0.000	193305400	77163700
8	0. 982216	0.020	193305400	77163700
16	0. 997218	0.060	193305400	77163700
32	0. 999742	0.120	193305400	77163700
128	0. 999919	0.347	193305400	77163700

Table 11.12: Discrepancies between objective functions for original and aggregated models in the DNS case study (a similar behavior is expected in MyTV).

Number of classes	Normalized objective function ϵ for the aggregated $CCCP$ problem	Normalized objective function ϵ for the original CCP problem
2	0.96962300	0.88249787
8	0.98221600	0.95332949
16	0.99721800	0.99412569
32	0.99974200	0.99966787
128	0.99991900	0.99991900

values computed by the optimization procedure. This discrepancy gets very quickly irrelevant when the number of classes increase, when we have 128 classes the results coincide.

11.5 Summary and Conclusions

In addition to the live video streaming service, Video Delivery Network must have a series of complementary services. In this chapter, we study the Video on Demand (VoD) service and the MyTV service, and in particular a caching search strategy for these services. We have developed a model of the impact that cache expiration times have on the total number of correct answers to queries in a content network, and on the bandwidth usage. This model has been used to develop a mathematical programming formulation, which allows to find optimal values for the cache expiration times in order to maximize the number of correct answers, subject to bandwidth limitations. In order to cope with the explosion of free variables, we have also developed an alternative formulation based on treating identically groups of similar contents. To show the feasibility of employing the mathematical programming formulation, we used a set of P2P test cases generated randomly in such a way that they comply with previously published information about existing networks. Also, we used a comprehensive data collection program to instantiate the optimization model in the DNS system case. The results show that the computational requirements are modest, and that the model results can lead to non-intuitive solutions giving high performance levels.

Also, to study the robustness of our procedure, we compare the model with two other alternative cache expiration time policies, which in this case show an small advantage of the optimal policy with respect to the two simplest ones. Moreover, we analyze different available optimization solvers.

We think that models of this kind lead to improved understanding of the behavior of content networks, and can be used to test their performance in a wide variety of potential scenarios, which are difficult to test in practice.

Future work includes studying more advanced non-linear programming solution methods which could be used to solve directly the problem with a large number of variables; improving the model to take into account other features of content networks; and implementing the caching policies at a cache node in a real network, in order to study the benefits in practice (and the possible side effects) of these policies with respect to current implementations. Also, we need to use the model with test cases corresponding to content network of different characteristics, and in particular for instances that come from real data of VoD and MyTV services. It is also possible to refine the model to take into account additional features. For example, the search answer packet sizes could be divided into a fixed part plus a variable, per location answered, part; additional constraints could be added to represent particular features of specific networks, in particular the time-to-live, TTL. Another interesting point is doing a more detailed analysis of the impact of the number of content classes chosen on the quality of the results obtained, as well as on the computational requirements imposed by the solution methods. Finally, a more difficult challenge is to integrate backbone behavior details into this model, in order to have a more wide perspective on the tradeoffs between information publication and search in a content network.

Chapter 12

General Conclusions and Perspectives

12.1 Conclusions

In this thesis, we presented a a quality-centric design of a live-video peer-to-peer distribution system.

We developed a novel mechanism to stream live video, called multi-source streaming technique, whose most important feature is a very low signalling cost (overhead), in contrast with Bittorrent-like approaches. Our mechanism decomposes the video stream into different redundant flows that travel independently through the network. We analyzed models that allow us to configure the number of flows, their rates, and the amount of redundancy transported by each of them, in order to obtain (and also statistical ensure) a pre-specified level of quality. Moreover, we provided a methodology to maximize the delivered perceived quality based on the heterogeneous peers' connectivity dynamics. The proposed streaming technique is directly applicable to networks with high probability of distribution failures (such as our P2P system) or when a very strict level of quality is desired.

The multi-source streaming technique is configured at the server side of the network for a specific quality-bandwidth trade-off. We also studied the protection strategy at the client side. A client can be protected from distribution failures increasing its buffer size (which also increases the delay). We obtained an expression of the buffer size that, in particular failure conditions, ensures a pre-specified quality level.

Multi-source streaming technique and buffer protection strategy make the servers and the clients of the network more robust to resource fluctuations, especially to peers' disconnection dynamics. These techniques act at each node, isolated from other decisions on the network. But our design also mitigates the impact of the peer' disconnection at the network level, placing the peers most commmitted to the network (those with largest lifetime and smallest bandwidth fluctuation) near to the broadcaster, in order to obtain a more robust overlay topology. We developed a preliminary design of a centralized tree-based overlay topology, for our P2P system. With very low signaling overhead, the overlay is built choosing which peer will serve which other node. We model the overlay topology design as a mathematical programming problem, where the optimization objective is to maximize the global expected perceived video quality on the network. To solve this problem, we studied different centralized algorithms. To compute a solution, these algorithms need to know the instantaneous quality in each node of the network. We developed a generic monitoring and measuring tool to provide this measure. This tool can be used by managers and administrators to assess the current streaming quality inside a generic video delivery network (not just our particular P2P design).

Observe that, from both the nodes and network perspectives, the aim is the same: to improve the average video quality perceived by the end users. With the main goal of a P2P distribution design, we provided a novel global design methodology that addresses the Quality-of-Experience (QoE), which is the ultimate target. The main component of the QoE in video delivery networks is the perceptual video quality. To measure the perceived video quality, automatically and with high accuracy, we extended the Pseudo-Subjective Quality Assessment (PSQA) technique in different ways. We studied the effects of distribution failures[1] on the perceived video quality, analyzing the losses at frame level instead of packet level studied in all previous works. We showed that the packet loss process does not correlate well with quality in a general context, because of its dependency of the protocol used. Instead, our frame loss analysis has general applicability to other network performance evaluations. We also studied the influence of video's motion on quality. Evaluating different motion activity metrics, we concluded that the perceived quality does not seem to have an important dependence on this source factor.

The output of applying the PSQA technique is a function able to mimic, somehow, the way that an average human assesses the quality of a stream. We applied the methodology three times, and we obtained three mapping quality functions that allowed us to quantify, in a generic way, *how* the frame losses affect the perceived quality. These result can be directly used in the performance evaluation of other networks.

The optimal multi-source streaming configuration, the tree-based overlay, and the monitoring suite are integrated in a prototype, called GOL!P2P. This prototype is a open-source application based on well proven technologies, and it is independent of operating systems.

GOL!P2P was designed as a generic implementation. The multi-source streaming implementation allows different configurations, codecs (MPEG-2/4), transport protocols (HTTP, RTP,...), and container formats (OGG, ASF,...). The monitoring suite can be used in different architectures, and it can be associated with most common management systems since it is built over the SNMP standard. The tree-based overlay accepts different solvers.

Our global prototype covers the most important aspects of the design. The system was configured and tested using real data from a video delivery reference service. The prototype showed the feasibility of a quality-centric design approach.

Present day Video Delivery Networks must provide a series of complementary interactive services. For instance, Video on Demand (VoD) and MyTV services allow the clients to submit content. In a very dynamic environment, efficient distribution of live video is the most challenging services, because of its bandwidth consumption and of its real-time restrictions. A

[1]Network congestion or servers failures.

different situation occurs when the contents themselves exhibit high dynamics. The discovery of very dynamic content can not be solved with traditional techniques, like publications by video podcast or broadcatching. We studied the design of efficient searches in video libraries for Video on Demand (VoD) and MyTV services. We developed a mathematical programming model of the impact of cache strategies on the total number of correct answers to queries on these services, and on the bandwidth usage. We applied the model in order to maximize the number of correct answers subject to bandwidth limitations in two scenarios with similar behavior to the studied services.

12.2 Perspectives

The addition of new interactive services, such as VoD and MyTV, is only one of the necessary improvements in order to convert our GOL!P2P prototype into a massive system. For instance, security and access control issues have not been studied yet. Simple improvements are:

- An enhanced synchronism mechanism is needed in our multi-source streaming technique to diminish the connection delay.
- An analysis extension in the buffering strategy, at the client side, in order to model the generic multi-source technique.
- A deeper study of the overlay construction; for instance, the exploration of other algorithms, in particular the distributed ones.

A more significant work has been started in the structured overlay methodology. We are now exploring a mesh-based overlay instead of the tree-based proposed. This will help us to compare the two approaches and to show the extensibility of our implementation. The new prototype is called Goalbit [2] and today it concentrates our main research efforts.

With Goalbit, we are showing the applicability of our quality-centric design to other contexts. In spite of the fact that the methodology is generic, every Video Delivery Network has its own particularities that impact into the quality of experience and they must be considered in the global design. For instance, in a IPTV system a fluid interactivity with the remote control is very important for the experience.

To understand the demand of a video service, and in particular the users behavior, is crucial to supply the expected quality level using the available resources. For that reason it is very important to have real data about the service that is designed. The users behavior dynamics is difficult to be considered in the overall design. Basically, we modeled the system in two independent time scales: in the short term we studied the best streaming technique that mitigates the peers' disconnections, and in the long term we built a more robust overlay topology placing the peers most committed to the network near to the broadcaster. A multi-time scale analysis is very complex, and we think that our design division is very adapted to these kind of networks.

[2]http://goalbit.sourceforge.net/

We expect that all or part of our methodology will be applied to future networks designs. In particular, the analysis of coupling between the available network resources and the users behavior, and the quality mapping functions are directly applied to other video delivery networks.

The success of any Video Delivery Network depends principally on the popularity of contents offered, but also in the scalability and QoE of the system. The choosing of a particular architecture by a Service Provider is based in a business plan that considers costs and strategic aspects. In the Internet context, there are mature commercial CDNs that offer live video streaming, but as we explained, with high costs. Also, some proprietary commercial P2P networks have been recently developed for live video. As far as we know, there isn't other open protocol and an open-source implementation of a P2P system for live video. This is a perfect opportunity for the diffusion of our prototype as an alternative for commercial and community services. We expect, with the open-source community help, to improve the system in the near future.

Appendix A

Comparing our Video Source Motion Measurement with Other Metrics

In this appendix we study the impact of video motion activity factor on the PSQA accuracy. This work has been done with the participation of G. Capdehourat, who completed an internship at the Irisa, France, in this subject.

These results extends the study presented in Section 5.3 about the video source motion effect on quality, where two simple source coding parameters was used to estimate the motion activity. In order to evaluate the impact of adding a video motion effect parameter, we apply the PSQA methodology to obtain a function of four inputs, three of them are network parameters, and represents the losses per kind of frame (LR_I, LR_P, LR_B); and the fourth one is the motion activity of each ten second sequence. We present different motion activity measures: based on histograms, based on motion vectors, and low level methods (see Section A.2). Motion activity is computed for the original videos, without frame losses. In Section A.4 we evaluate the performance of each measure. The appendix finalizes with general conclusions on Section A.5.

A.1 Motion Activity

Motion Activity refers to the human perception of movement in a video sequence. Examples of high "activity" include scenes such as "race driven" and "goal scoring in a soccer match". On the other hand, examples of low "activity" include scenes such as "weather report" and "interview show". Activity descriptors can be defined to capture this intuitive notion of "intensity of action" or "pace of action" in a video sequence. As we explain below, activity descriptors are applied in the literature for applications such as video compression, video summarization, etc.

One of the motion activity descriptors defined in MPEG-7 [51], in particular the one we are interested in, is the *Intensity of Activity* [182]. It is expressed by an integer lying in the range 1-5. A high value indicates high activity while a low value indicates low activity, activity defined in the way before mentioned. The principal steps of the measurement system are shown in Figure A.1. The first one refers to the particular method implemented for measuring the instantaneous measure, taking into account each frame transition of the sequence. At least two frames are necessary to take this measure, but more can be used also. After having a value of the instantaneous motion activity, the measure for the whole sequence is calculated. One possible way to do this is to take the average of the instantaneous measure. Several ways where tested for measuring the amount of motion, which are detailed in Section A.2. Finally a quantization step maps the amount of motion, which is a real value, to the desired range 1-5 in the integer domain.

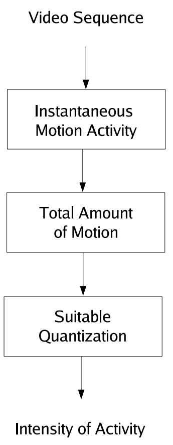


Figure A.1: Automatic measurement of motion activity.

There is one difference between the *Intensity of Activity*, motion activity descriptor defined in MPEG-7 [51], and the amount of motion used for the PSQA system. While the first one is an integer lying in the range 1-5, the one used in PSQA, as another input to the RNN is a real value, so the quantization step is obviated in this case.

A.1.1 Related work.

Motion analysis in video sequences is not a new area, with a lot of research developed during last twenty years. Several techniques have been applied for many different applications, motion estimation for video compression and motion detection for video surveillance are only some examples. In particular, this notion of motion activity as described in the MPEG-7 standard [51], has been already used for many different purposes; some of them are summarized next.

Video content characterization. Motion activity can be used to characterize video in terms of its content. This enables applications as content-based retrieval, where you can filter a lot of different video sequences taking into account things like the degree of action, presence of violence or sex scenes. As an example, Vasconcelos et al. [330] classified movies in terms of the genre, separating the romance and the action ones. The amount of variation in the scene on each frame transition was estimated through the tangent distance between images. Peker et al. [253] used MPEG motion vectors to estimate motion activity to achieve close-ups detection in sport videos and high activity shots retrieval.

Key frame selection. A key frame summarizes the content of a video sequence, so it is useful for example to make storyboards for movies, used by directors to summarize scenes of the story. Wolf [349] utilized the motion activity, calculated as the mean of the motion vectors magnitude, in order to identify key frames as the local minima of motion on a shot, related to the stillness in the sequence. The motion vectors correspond to the optical flow, which was obtained through the Horn and Schunck algorithm [133]. A different approach was presented by Narashima et al. [231], using the MPEG motion vectors in order to estimate not only motion activity but also its spatial distribution.

Video indexing and retrieval. Larger video databases are more common everyday, so an appropriate way to index them, in order to be able to make quicker queries, is also necessary. Several measures were used for this purpose related to the idea of motion activity, most of them using motion vector information. Two examples of this work are presented by Ardizzone et al. [20] and Akutsu et al. [8].

Video summarization. Summarising video data is essential to enable content-based video indexing and retrieval. Generation of sports video highlights, news summaries and movie trailers, are some examples of the applications for video summarization. Divakaran et al. [83] used the hypothesis that the intensity of motion activity indicates the difficulty of summarization of a video shot in order to develop a summarization algorithm.

A.2 Motion Activity Measurement

Three different kinds of methods to measure the motion activity were studied in the literature. The first type, based on low level calculations on the image sequences, allows a very fast and simple measure. The second one is based on the motion vectors, in our case calculated with the Horn & Schunck optical flow algorithm [133]. The third and last kind of methods tested were the histogram-based ones.

A.2.1 Low level methods.

This kind of methods try to estimate the motion activity taking into account differences between frames at transitions, looking the images at the pixel level, and getting a measure that indicates the motion activity based on few operations.

Two principal methods were tested, both of them consisting of very simple calculations on each image transition. Both of them consider only luminance information of each image, which is the same as use only greyscale images information of the sequence.

Motion Activity Matrix. This method was introduced by Oh et al. in [239] for the computation of motion activity (MA). A matrix called Motion Activity Matrix (MAM) is defined, with size equal to the video frame size. Basically, this matrix is calculated by looking at each frame transition and increasing each value of the matrix if the pixels corresponding to that position in the two referred images are different. Nothing is done if they are equal. Starting from a null matrix and after repeating the process for each frame transition, finally the MAM for the hole video sequence is obtained. In order to compute the amount of motion of the hole video segment, called total motion (TM), the average of all the matrix values is calculated.

$$MA = TH = \frac{1}{N}\frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W} MAM_{ij} \tag{A.1}$$

where MA stands for Motion Activity and TH for Total Motion, N is the number of frames of the sequence, W is the width and H the height of each frame. All the values dividing in the equation A.1 are for normalization purposes, in order to obtain a measure value independent of the frame and the sequence size.

A slight variation of this method is to consider only pixel intensity differences greater than a fixed value, which is like a quantization to a lower dimension space (i.e. from 256 to 64 or 32 levels).

Absolute Difference. In this method, the instantaneous motion activity is measured at each frame transition by computing the mean of the absolute difference between the two images. Then, in order to obtain the motion activity (MA) for the whole sequence, the average of all the computed values is calculated.

$$MA = \frac{1}{N}\sum_{T=1}^{N-1}\frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W}|I_{T+1}(i,j) - I_T(i,j)|$$

where the notation is the same as in equation A.1; I_{T+1} and I_T are the images corresponding to frames T and T+1 respectively.

A similar variation to the previous case was tested, taking into account only the differences greater than a minimum value (7) and below a maximum value (100). The ones below the minimum are considered 0 and the ones over the maximum are fixed to the maximum value (in our case 100) in order to eliminate high intensity differences influence into the motion measure. This method is referred as DIF2 in the results.

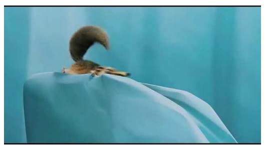
A.2.2 Motion vectors based methods.

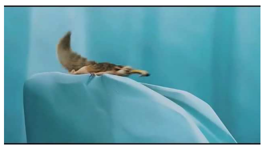
The estimation of motion vectors implies more computational cost than the previous kind of methods. This fact is due to the need to solve a partial differential equation (PDE) via optimization, in order to obtain the motion vectors field. Most of the recent video codecs use motion vectors for compression purposes, so in many cases those ones could be used to avoid the computation.

(a) frame 1

(b) frame 25

(c) frame 50


(d) frame 75


(e) frame 100

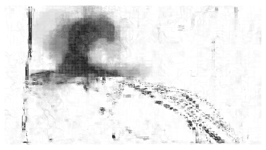
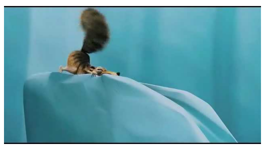
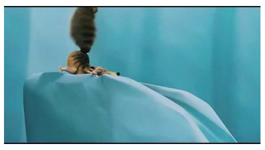

(f) MAM

Figure A.2: Frames 1, 25, 50, 75, 100 and the MAM for the whole sequence.


(a) frame 1


(b) frame 2

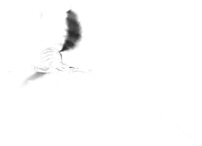

(c) Difference

Figure A.3: Frames 1 and 2, and the absolute difference between them.

The optical flow between image pairs at each frame transition is computed with the Horn & Shunck algorithm [133]. After having the motion vectors for each transition, the motion activity is calculated by two different ways. These methods as the previous ones, only consider grayscale information of the image sequences.

Average Magnitude. For each frame transition, the instantaneous motion activity is computed as the average of all the motion vector magnitudes. In order to calculate the motion activity for the whole sequence, the mean of the instantaneous motion activity during the sequence is computed.

Magnitude Standard Deviation. In this case the standard deviation of the motion vector magnitudes is used to compute the instantaneous motion activity. This measure is based on the observation that the perceived motion activity is higher when the motion is not uniform. This procedure was chosen in [182] to compute the MPEG-7 *intensity of activity* descriptor. The extension to the whole sequence is the same than in the previous case, the mean of all the values for each frame transition is calculated.

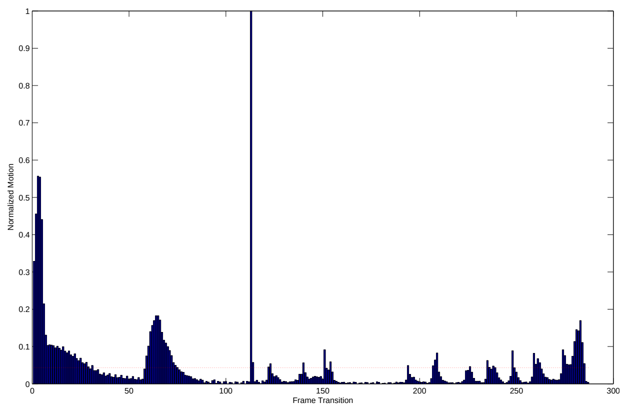


Figure A.4: Instantaneous Motion Activity for the same sequence of Figure A.2. Motion Activity for the whole sequence (mean) is in red.

A.2.3 Histogram based methods.

Although these methods do not consider spatial information, because it is not used for histogram computation, it is based on the hypothesis that in a video sequence, it is not probable [1] that one frame has nothing to do with the next one. In that sense, histogram differences are used to measure changes between one frame and the next one, which can be assumed as the instantaneous amount of motion occurred in the sequence.

The three different methods tested based on histogram calculations follow the same procedure. For each frame transition, the histogram is computed for both images and then the

[1]The probability that it happens is very small

histogram intersection between them is computed following the formula shown below.

$$I(H_1, H_2) = \sum_{i=1}^{B} min(H_1(i), H_2(i))$$

where B stands for the number of bins used in the histogram computation.

The bigger is the histogram intersection, the less is the change between the frames, so the instantaneous motion activity considered has to be inversely proportional to this value. To do this, the negative of the obtained value is used as motion activity. Finally the average is computed in order to obtain the total motion activity for the hole sequence.

$$MA = -\frac{1}{N}\frac{1}{H \times W}\sum_{T=1}^{N}\sum_{i=1}^{B} min(H_{T+1}(i), H_T(i))$$

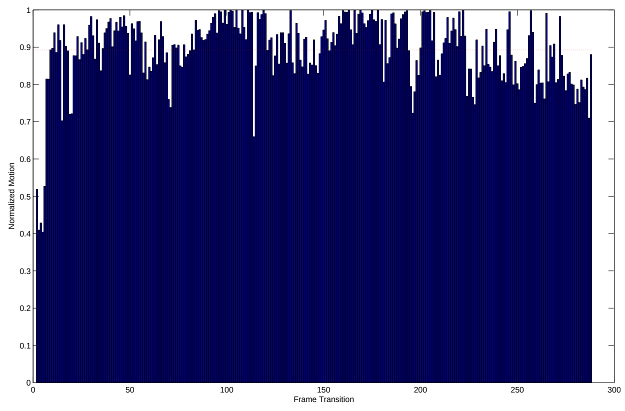


Figure A.5: Instantaneous Motion Activity for the same sequence of Figure A.2. Motion Activity for the hole sequence (mean) is in red.

Three options were used, which differ only in the information taken into account from each image of the sequence.

Luminance. Only the luminance information is used, so the histogram of each greyscale image is computed.

RGB color. The three color channels RGB of each image are considered. In order to get only one histogram vector per image, the three histograms are concatenated.

YCbCr color. In this case the same procedure of the previous option is followed, but the three channels considered are YCbCr, taking the image to that color space before the histogram computation.

A.3 Evaluation and Results

In order to evaluate the impact of adding a precise motion metric parameter, four inputs to the RNN were used, three of them are the same as in NET (loss rate parameters: LR_I, LR_P and LR_B), and the fourth one is the motion activity of each ten second sequence. Motion activity is computed for the original videos, without packets loss.

We compare the performance of PSQA with several motion metrics. Tested methods are identified by the labels of Table A.1. A One-man subjective measure of the motion activity is used. Subjectively, a expert defines a value for the amount of motion in each original video sequence. This subjective motion activity is different from the other objective measures because it is a discrete value.

Table A.1: Motion Activity Metrics. We train a RNN with each metric as a parameter (and the loss rate parameters).

	Low-level methods
MAM	Motion Activity Matrix
MAM2	MAM modified
DIF	Absolute difference
DIF2	DIF modified
	Motion vectors methods
OPTHS	Average magnitude of the motion vectors
OPTHS2	Standard deviation of the motion vectors
	Histogram based methods
HIS	Histogram intersection of grayscale images
HIS3	Histogram intersection of RGB images
HIS3b	Histogram intersection of YCbCr images
	Subjective
SUB	Subjective

Best results for the different cases are resumed in the Table A.2. There is no great difference between each motion activity metric. After the analysis of all the results, the best

Table A.2: Best RNN for each input motion metric parameter.

Name	Hidden neurons	MSE validation
MAM	3	0.0272313
MAM2	8	0.0255118
DIF	8	0.0177780
DIF2	3	0.0296392
OPTHS	7	0.0189175
OPTHS2	4	0.0282770
HIS	15	0.0262564
HIS3	15	0.0262928
HIS3b	3	0.0248390
SUB	3	0.0275326

performance of the PSQA system is reached using the motion activity computed by the DIF algorithm, which takes into account absolute difference between images to estimate motion activity.

Any motion activity metric improves the precision with respect to the network without motion parameters (NET). Table A.3 compares the system with the motion activity as input with the other options, the improvement is considerable in terms of the absolute standard deviation enhance with respect to an average subject (the deviation in our test campaign). As it can be

Table A.3: RNN Comparison between an Average Subject, our Complex Functions and the Best performance Motion Metric parameters.

Name	MSE validation	Deviation δ	Improvement (%)
human	0.0207025	0.153000	
NET	0.0296717	0.172255	-1.9
NET2	0.0288654	0.169898	-1.7
NETSRC	0.0269109	0.164045	-1.1
NETSRC2	0.0254898	0.159655	-0.7
DIF	0.0177780	0.133334	2.0
OPTHS	0.0189175	0.137541	1.6
HIS3b	0.0248390	0.157604	-0.5
SUB	0.0275326	0.165930	-1.3

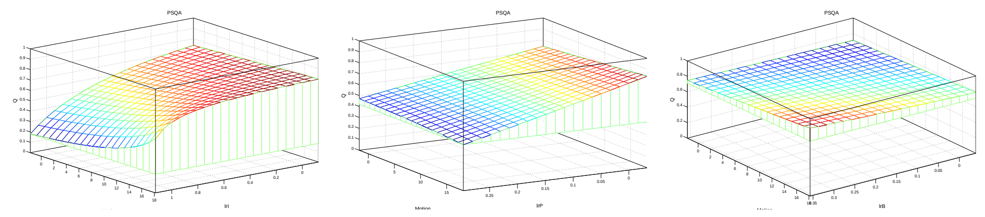
seen, the absolute improvement is very small. In particular, the DIF and OPTHS algorithms are the only ones which are below than the human error, so they are the best results, even better than an arbitrary human. Apparently, the quality is not very sensible to the motion activity, and therefore when two videos with different motion face the same fault, the quality perception of them are approximately the same one.

A.4 Motion Activity Influence in the Perceptual Quality

Next, the best two methods for computing the motion activity (in terms of lower MSE) are considered: DIF and OPTHS. The resulting functions (after the RNN training) are shown, particularly analyzing the dependence on each parameter.

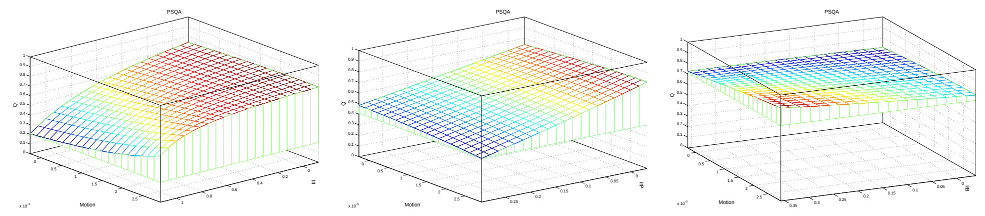
To observe the effect of motion into the quality, we follow the same procedure used in the *complex function*. We show the motion parameters impacts with respect to each kind of frame loss rate independently. Figures A.6 and A.7 show the perceived quality as a function of the motion parameter and one kind of frame loss, the other two losses parameters (hidden in the figures) are fixed to zero.

For the cases of I and P frames loss rates, both are quite similar with slight differences. In both cases, quality falls down while loss rates for both type of frames grow, which is as expected. For the lower values of the loss rates, the quality rises while the motion activity grows, but it is not a big increment, only around one unit of quality. The behavior changes for higher loss rates, where in the case of I-frames, quality still grows with motion but for P-frames, if the loss rate is more than 15% quality starts to go down as motion grows. It can be stated that motion activity has not a big influence in the perceived quality, based on the fact that



(a) I-Frames loss rate, while the other loss rates are 0. (b) P-Frames loss rate, while the other loss rates are 0. (c) B-Frames loss rate, while the other loss rates are 0.

Figure A.6: Estimated subjective quality as function of the Motion Activity (DIF) and loss rates.



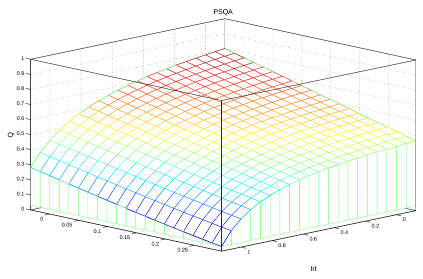
(a) I-Frames loss rate, while the other loss rates are 0. (b) P-Frames loss rate, while the other loss rates are 0. (c) B-Frames loss rate, while the other loss rates are 0.

Figure A.7: Estimated subjective quality as function of the Motion Activity (OPTHS) and loss rates.

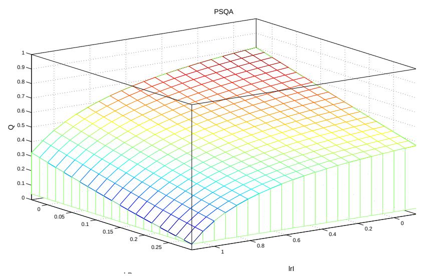
the biggest variation of quality with the motion activity is never bigger than one unit, less than the error humans do when qualifying the videos.

The behavior is really different for the case of the B-frames loss rate, where almost none influence of both parameters is seen. Moreover, a slight growth of quality while the B-frames loss rate rises can be seen, but it is very small to be considered.

One interesting point in the analysis is that if both cases are compared, the one which considers motion activity computed with the DIF algorithm and the other with the OPTHS one, there is almost no difference between them, which is a really an important result. It indicates, not only that the two different measures are consistent for this application, but also that DIF algorithm, much cheaper in terms of computational cost gets the same performance as OPTHS, a much more expensive one. This similar behavior also happens if we observe only the relationship between kind of losses. Figures A.8, A.9 and A.10 present the dependence on the loss rates shown in pairs, all of the cases for an average value of the motion activity and loss rate 0 for the one left out on each case. Identical to our *complex function*, Figures show that quality falls down as loss rates of I and P frames grow. However, B-frames loss rate does not present an impact on the perceived quality.

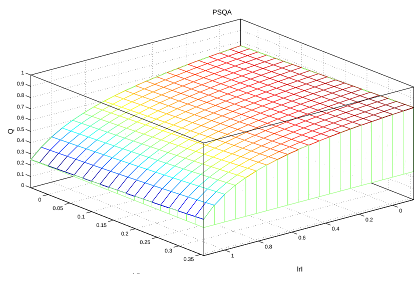


(a) the average value of the motion activity (DIF).

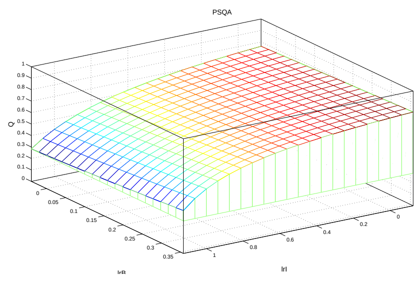


(b) the average value of the motion activity (OPTHS).

Figure A.8: Estimated subjective quality as function of the I-Frames and P-Frames loss rates.

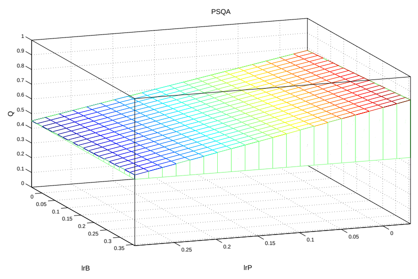


(a) the average value of the motion activity (DIF).

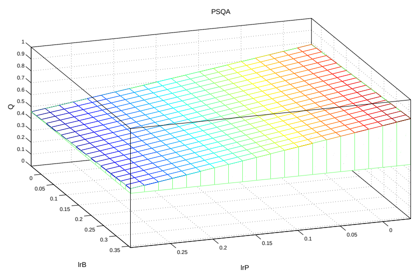


(b) the average value of the motion activity (OPTHS).

Figure A.9: Estimated subjective quality as function of the I-Frames and B-Frames loss rates.



(a) the average value of the motion activity (DIF).



(b) the average value of the motion activity (OPTHS).

Figure A.10: Estimated subjective quality as function of the P-Frames and B-Frames loss rates.

A.5 Summary and Conclusions

PSQA performance was tested, with the addition of a parameter related to the nature of the scene, as it is the intensity of motion activity in the sequence. The results achieved by the approximated functions that take into account motion parameters are quite similar to the functions without these parameters in terms of the MSE. An insignificant improvement of 2% is obtained by adding this input to the system. Also analyzing the qualitative behavior of the best motion functions, the perceived quality does not seem to have an important dependence on the motion activity.

The *Complex Function*, presented in Section 5.3.3, uses simple source parameters related indirectly with the nature of the scene, based on the encoder decisions who takes into account the content of the video sequence to encode the stream. The precision improvement of the quality assessment does not justify the use of complex metrics to compute the motion activity. The use of simple source parameters, as the used in our *Complex Function*, are enough in terms of precision and complexity for a real–time application.

Bibliography

[1] 3GPP Specifications Home Page. http://www.3gpp.org/specs/specs.htm, 2007.

[2] Abacast home page. http://www.abacast.com/, 2007.

[3] Abilene Internet2 Network home page. http://abilene.internet2.edu/, 2007.

[4] Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Edward A. Fox, and Stephen Williams. Removal policies in network caches for world-wide web documents. *SIGCOMM Comput. Commun. Rev.*, 26(4):293–305, 1996.

[5] Anant Agarwal, Richard Simoni, John Hennessy, and Mark Horowitz. An evaluation of directory schemes for cache coherence. In *ISCA '98: 25 years of the international symposia on Computer architecture (selected papers)*, pages 353–362, New York, NY, USA, 1998. ACM.

[6] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory.*, 2000.

[7] Akamai website. http://www.akamai.com/, 2007.

[8] A. Akutsu, Y. Tonomura, H. Hashimoto, and Y. Ohba. Video indexing using motion vectors. In P. Maragos, editor, *Proc. SPIE Vol. 1818, p. 1522-1530, Visual Communications and Image Processing '92, Petros Maragos; Ed.*, volume 1818 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 1522–1530, November 1992.

[9] Alcatel-Lucent Home. http://www.alcatel-lucent.com/, 2007.

[10] Shahzad Ali, Anket Mathur, and Hui Zhang. Measurement of commercial peer-to-peer live video streaming. In *In Proc. of ICST Workshop on Recent Advances in Peer-to-Peer Streaming*, Weaterloo, Canadda, 2006.

[11] Allcast home page. http://www.allcast.com/, 2007.

[12] Stephen Alstrup and Theis Rauhe. Introducing octoshape - a new technology for large-scale streaming over the internet. Technical report, European Broadcasting Union (EBU), 2005.

[13] AMPL - A Modeling Language for Mathematical Programming. Home Page. http://www.ampl.com/, 2007.

[14] Globule: an Open-Source CDN Home page. http://www.globule.org/, 2007.

[15] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, 2004.

[16] J. Apostolopoulos and S. Wee. Unbalanced multiple description video communication using path diversity. In *IEEE International Conference on Image Processing*, pages 966–969, October 2001.

[17] J.G. Apostolopoulos. Reliable video communication over lossy packet networks using multiple state encoding and path diversity. In *Proc. Visual Communication and Image Processing, VCIP '01*, pages 392–409, January 2001.

[18] J.G. Apostolopoulos and M.D. Trott. Path diversity for enhanced media streaming. *IEEE Communications Magazine*, 42(8):80–87, August 2004.

[19] John G. Apostolopoulos, Wai tian Tan, and Susie J. Wee. Video streaming: Concepts, algorithms, and systems. Technical Report HPL-2002-260, HP Labs, 2002.

[20] E. Ardizzone, M. La Cascia, A. Avanzato, and A. Bruna. Video indexing using mpeg motion compensation vectors. In *ICMCS '99: Proceedings of the IEEE International Conference on Multimedia Computing and Systems Volume II-Volume 2*, page 725, Washington, DC, USA, 1999. IEEE Computer Society.

[21] W. Ashmawi, R. Guerin, S. Wolf, and M. Pinson. On the impact of policing and rate guarantees in diff-serv networks: A video streaming application perspective. *Proceedings of ACM SIGCOMM'2001*, pages 83–95, 2001.

[22] Todd M. Austin, Dionisios N. Pnevmatikatos, and Gurindar S. Sohi. Streamlining data cache access with fast address calculation. *SIGARCH Comput. Archit. News*, 23(2):369–380, 1995.

[23] Aviel and Lorrie F. Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proc. 9th USENIX Security Symposium*, pages 59–72, August 2000.

[24] H. Bakircioglu and T. Kocak. Survey of Random Neural Network Applications. *European Journal of Operational Research*, 126(2):319–330, 2000.

[25] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. Scalable application layer multicast. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 205–217, New York, NY, USA, 2002. ACM Press.

[26] A. Basso, I. Dalgic, F. Tobagi, and C. Lambrecht. Study of mpeg-2 coding performance based on a perceptual quality metric. In *Proceedings of PCS'96*, pages 263–268, March 1996.

[27] BBC: iMP website. http://www.bbc.co.uk/imp/, 2007.

[28] BBC Multicast initiative. http://support.bbc.co.uk/multicast/, 2007.

[29] B. Bhattacharjee, S. Chawathe, V. Gopalakrishnan, P. Keleher, and B. Silaghi. Efficient peer-to-peer searches using result-caching. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, volume 2735/2003, February 2003.

[30] Bittorrent home page. http://www.bittorrent.org, 2007.

[31] Bittorrent movie site. http://www.bittorrent.com, 2007.

[32] Chonggang Wang Bo. Peer-to-peer overlay networks: A survey.

[33] G. A. Bolcer, M. Gorlick, A. S. Hitomi, P. J. Kammer, B. Morrow, P. Oreizy, and R. N. Taylor. Peer-to-Peer Architectures and the Magi Open Source Infrastructure. Technical report, Endeavors technologies (www.endeavors.com), 2000.

[34] Gregory Alan Bolcer. Magi: An architecture for mobile and disconnected workflow. *IEEE Internet Computing*, 4(3):46–54, 2000.

[35] J-C. Bolot, S. Fosse-Parisis, and D.F. Towsley. Adaptive FEC–Based Error Control for Internet Telephony. In *Proceedings of INFOCOM '99*, pages 1453–1460, New York, NY, USA, March 1999.

[36] A. Bouch and M. A. Sasse. Why Value is Everything: a User-Centered Approach to Internet Quality of Service and Pricing. *Lecture Notes in Computer Science*, 2092, 2001.

[37] A. Bouch, M. A. Sasse, and H. DeMeer. Of Packets and People: a User-centered Approach to Quality of Service. *Proceedings of IWQoS2000*, pages 189–197, 2000.

[38] L. Breslau, Pei Cao, Li Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: evidence and implications. *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1:126–134 vol.1, 21-25 Mar 1999.

[39] Liu C. *DNS and BIND Cookbook.* O'Reilly, New York, October 2002.

[40] H. Cancela, F. Robledo, and G. Rubino. A GRASP algorithm with RNN based local search for designing a WAN access network. *Electronic Notes on Discrete Mathematics*, 18C:53–58, 2004.

[41] Hector Cancela, Franco Robledo Amoza, Pablo Rodríguez-Bocca, Gerardo Rubino, and Ariel Sabiguero. A robust P2P streaming architecture and its application to a high quality live-video service. *Electronic Notes in Discrete Mathematics.*, 30C:219–224, 2008.

[42] Hector Cancela and Pablo Rodríguez-Bocca. Optimization of cache expiration dates in content networks. In *Proceedings of the 3rd international conference on the Quantitative Evaluation of Systems (QEST'06)*, pages 269–278, Washington, DC, USA, September 11-14 2006. IEEE Computer Society.

[43] Bengt Carlsson and Rune Gustavsson. The rise and fall of napster - an evolutionary approach. In *AMT '01: Proceedings of the 6th International Computer Science Conference on Active Media Technology*, pages 347–354, London, UK, 2001. Springer-Verlag.

[44] Tamra Carpenter, Robert Carter, Munir Cochinwala, and Martin Eiger. Client-server caching with expiration timestamps. *Distrib. Parallel Databases*, 10(1):5–22, 2001.

[45] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.*, 19(3):332–383, 2001.

[46] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 20(8):1489–1499, 2002.

[47] Miguel Castro, Manuel Costa, and Antony Rowstron. Debunking some myths about structured and unstructured overlays. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 85–98, Berkeley, CA, USA, 2005. USENIX Association.

[48] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. Splitstream: High-bandwidth content distribution in a cooperative environment. In *IPTPS '03: Proceedings of the Second International Workshop on Peer-to-Peer Systems*, page 6, 2003.

[49] CastUP home page. http://www.castup.net/, 2007.

[50] CDNetworks home page. http://www.cdnetworks.com, 2007.

[51] Shih-Fu Chang, Thomas Sikora, and Atul Puri. Overview of the mpeg-7 standard. *IEEE Trans. Circuits Syst. Video Techn.*, 11(6):688–695, 2001.

[52] Yatin Chawathe. Scattercast: an adaptable broadcast distribution framework. *Multimedia Syst.*, 9(1):104–118, 2003.

[53] Xiangping Chen and Prasant Mohapatra. Lifetime behavior and its impact on web caching. In *WIAPP '99: Proceedings of the 1999 IEEE Workshop on Internet Applications*, page 54, Washington, DC, USA, 1999. IEEE Computer Society.

[54] Dah M. Chiu, R. W. Yeung, Jiaqing Huang, and Bin Fan. Can network coding help in p2p networks? In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2006 4th International Symposium on*, pages 1–5, 2006.

[55] P. A. Chou, Y. Wu, and K. Jain. Practical network coding. *41st Conference on Communication Control and Computing.*, Oct 2003.

[56] Philip Chou, Yunnan Wu, and Kamal Jain. Network coding for the internet. *IEEE Communication Theory Workshop, Capri, IEEE.*, 2004.

[57] J. Chu, K. Labonte, and B. Levine. Availability and locality measurements of peer-to-peer file systems. In *Proceedings of SPIE. ITCom: Scalability and Traffic Control in IP Networks*, volume 4868, july 2002.

[58] Cisco Content Networking Architecture. Cisco Systems, Inc. http://www.cisco.com/go/cdn/, 2005.

[59] Cisco Systems, Inc. http://www.cisco.com/, 2007.

[60] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46–??, 2001.

[61] Munir Cochinwala. Database performance for next generation telecommunications. In *Proceedings of the 17th International Conference on Data Engineering*, pages 295–298, Washington, DC, USA, 2001. IEEE Computer Society.

[62] Edith Cohen and Haim Kaplan. Exploiting regularities in web traffic patterns for cache replacement. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 109–118, New York, NY, USA, 1999. ACM.

[63] Edith Cohen and Haim Kaplan. The age penalty and its effect on cache performance. In *USITS'01: Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems*, pages 7–7, Berkeley, CA, USA, 2001. USENIX Association.

[64] Edith Cohen and Haim Kaplan. Proactive caching of dns records: Addressing a performance bottleneck. In *SAINT '01: Proceedings of the 2001 Symposium on Applications and the Internet (SAINT 2001)*, page 85, Washington, DC, USA, 2001. IEEE Computer Society.

[65] H. Cohen, E.; Kaplan. Prefetching the means for document transfer: a new approach for reducing web latency. *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2:854–863 vol.2, 2000.

[66] H. Cohen, E.; Kaplan. Refreshment policies for web content caches. *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 3:1398–1406 vol.3, 2001.

[67] C. Colomes, M. Varela, and J-C. Gicquel. Subjective Audio Tests: Quality of Some Codecs When Used in IP Networks. In *Proceedings of the Measurement of Speech and Audio Quality in Networks workshop (MESAQIN'04)*, Prague, Czech Republic, June 2004.

[68] CoolStreaming home page. http://www.coolstreaming.us, 2007.

[69] Avaki Corporation. Avaki 2.0 concepts and architecture. Technical report, Avaki Corporation, October 2001.

[70] C. Cramer and E. Gelenbe. Video Quality and Traffic QoS in Learning-Based Sub-Sampled and Receiver-Interpolated Video Sequences. *IEEE Journal on Selected Areas in Communications*, 18(2):150–167, 2000.

[71] C. Cramer, E. Gelenbe, and H. Bakircioglu. Low bit rate video compression with neural networks and temporal sub-sampling. *Proceedings of the IEEE*, 84(10):1529–1543, 1996.

[72] Charles D. Cranor, Matthew Green, Chuck Kalmanek, David Shur, Sandeep Sibal, Jacobus E. Van der Merwe, and Cormac J. Sreenan. Enhanced streaming services in a content distribution network. *IEEE Internet Computing*, 5(4):66–75, 2001.

[73] Joseph Czyzyk, Michael P. Mesnier, and Jorge J. Moré. The neos server. *IEEE Comput. Sci. Eng.*, 5(3):68–75, 1998.

[74] Ana Paula Couto da Silva, Pablo Rodríguez-Bocca, and Gerardo Rubino. Coupling QoE with dependability through models with failures. In *8th International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS'07)*, Edinburgh, Scotland, September 20-21 2007.

[75] Ana Paula Couto da Silva, Pablo Rodríguez-Bocca, and Gerardo Rubino. Optimal quality–of–experience design for a p2p multi-source video streaming. In *IEEE International Conference on Communications (ICC 2008)*, Beijing, China, 19-23 May 2008.

[76] Michael D. Dahlin, Clifford J. Mather, Randolph Y. Wang, Thomas E. Anderson, and David A. Patterson. A quantitative analysis of cache policies for scalable network file systems. *SIGMETRICS Perform. Eval. Rev.*, 22(1):150–160, 1994.

[77] Niranjan Damera-Venkata, Thomas D. Kite, Wilson S. Geisler, Brian L. Evans, and Alan C. Bovik. Image quality assessment based on a degradation model. *IEEE Transactions on Image Processing*, 9(4):636–650, April 2000.

[78] C. Dana, Danjue Li, D. Harrison, and Chen-Nee Chuah. Bass: Bittorrent assisted streaming system for video-on-demand. In *Multimedia Signal Processing, 2005 IEEE 7th Workshop on*, pages 1–4, 2005.

[79] J. Van der Merwe, S. Sen, and C. Kalmanek. Streaming video traffic: Characterization and network impact. In *Proceedings of the 7th International Workshop on Web Content Caching and Distribution, Boulder, CO, USA*, August 2002.

[80] H. Deshpande, M. Bawa, and H. Garcia-Molina. Streaming live media over peer-to-peer network. Technical report, Stanford University, 2001., 2001.

[81] John Dilley, Bruce Maggs, Jay Parikh, Harald Prokop, Ramesh Sitaraman, and Bill Weihl. Globally distributed content delivery. *IEEE Internet Computing*, 6(5):50–58, 2002.

[82] Direct Connect home page. http://www.neo-modus.com., 2005.

[83] A. Divakaran, K. Peker, R. Radhakrishnan, Z. Xiong, and R. Cabasson. Video summarization using mpeg-7 motion activity and audio descriptors. Technical Report Report TR2003-34, Mitsubishi Electric Research Laboratories, May 2003.

[84] Doom Game home page. http://www.idsoftware.com/games/doom/doom-final/, 2005.

[85] P. Druschel and A. Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In *Eighth Workshop on Hot Topics in Operating Systems (HotOS'2001)*, pages 75–80, 2001.

[86] DSL Forum home page. http://www.dslforum.org/, 2007.

[87] DSL Forum Technical Work WT-126. Video services quality of experience (qoe) requirements and mechansims, Apr 2007.

[88] Edutella home page. http://edutella.jxta.org, 2005.

[89] eMule home page. http://www.emule-project.net, 2007.

[90] End System Multicast home page. http://esm.cs.cmu.edu/, 2007.

[91] Hans Eriksson. Mbone: the multicast backbone. *Commun. ACM*, 37(8):54–60, 1994.

[92] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: scalable coordination in sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 263–270, New York, NY, USA, 1999. ACM.

[93] Evertz Microsystems Ltd. - Manufacturer of High Quality Film and Digital Broadcast Equipment. http://www.evertz.com/, 2007.

[94] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary cache: a scalable wide-area Web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3):281–293, 2000.

[95] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Trans. Netw.*, 8(3):281–293, 2000.

[96] Folding@home home page. http://foldingathome.stanford.edu., 2005.

[97] CoDeeN: A Content Distribution Network for PlanetLab Home page. http://codeen.cs.princeton.edu/, 2007.

[98] Eran Gabber, Phillip B. Gibbons, David M. Kristol, Yossi Matias, and Alain Mayer. Consistent, yet anonymous, web access with lpwa. *Commun. ACM*, 42(2):42–47, 1999.

[99] Z. Ge, D. R. Figueiredo, Sharad Jaiswal, J. Kurose, and D. Towsley. Modeling peer-peer file sharing systems. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2003. IEEE*, volume 3, pages 2188–2198, March 2003.

[100] E. Gelenbe. Random Neural Networks with Negative and Positive Signals and Product Form Solution. *Neural Computation*, 1(4):502–511, 1989.

[101] E. Gelenbe. Stability of the Random Neural Network Model. In *Proc. of Neural Computation Workshop*, pages 56–68, Berlin, West Germany, February 1990.

[102] E. Gelenbe. Learning in the Recurrent Random Neural Network. *Neural Computation*, 5(1):154–511, 1993.

[103] E. Gelenbe. G-Networks: new Queueing Models with Additional Control Capabilities. In *Proceedings of the 1995 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 58–59, Ottawa, Ontario, Canada, 1995.

[104] E. Gelenbe. Towards Networks with Cognitive Packets. In *Proc. IEEE MASCOTS Conference*, pages 3–12, San Francisco, CA, 2000.

[105] E. Gelenbe. Reliable networks with cognitive packets. In *International Symposium on Design of Reliable Communication Networks*, pages 28–35, Budapest, Hungary, October 2001. Invited Keynote Paper.

[106] E. Gelenbe. Cognitive packet networks: QoS and performance. In *Proc. IEEE Computer Society MASCOTS02 Symposium*, pages 3–12, Fort Worth, TX, October 2002. Opening Keynote.

[107] E. Gelenbe and F. Batty. Minimum cost graph covering with the random neural network. *Computer Science and Operations Research*, 1:139–147, 1992.

[108] E. Gelenbe, C. Cramer, M. Sungur, and P. Gelenbe. Traffic and Video Quality in Adaptive Neural Compression. *Multimedia Systems*, 4(6):357–369, 1996.

[109] E. Gelenbe and J.M. Fourneau. Random Neural Networks with Multiple Classes of Signals. *Neural Computation*, 11(3):953–963, 1999.

[110] E. Gelenbe, V. Koubi, and F. Pekergin. Dynamical random neural network approach to the traveling salesman problem. *Systems, Man and Cybernetics, 1993. 'Systems Engineering in the Service of Humans', Conference Proceedings., International Conference on*, 2:630–635, 17-20 Oct 1993.

[111] E. Gelenbe, R. Lent, and Z. Xu. Design and Performance of Cognitive Packet Networks. *Performance Evaluation*, 46:155–176, 2001.

[112] Genome@home home page. http://genomeathome.stanford.edu., 2005.

[113] E. Gilbert. Capacity of a burst–loss channel. *Bell Systems Technical Journal*, 5(39), September 1960.

[114] Steven Glassman. A caching relay for the world wide web. *Comput. Netw. ISDN Syst.*, 27(2):165–173, 1994.

[115] Gnutella home page. http://gnutella.wego.com/, 2005.

[116] Google home page. http://www.google.com/, 2005.

[117] Parikshit Gopalan, Howard Karloff, Aranyak Mehta, Milena Mihail, and Nisheeth Vishnoi. Caching with expiration times. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 540–547, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[118] Vivek K Goyal. Multiple description coding: Compression meets the network. *IEEE Signal Processing Magazine*, September 2001.

[119] Vivek K. Goyal. Multiple description coding: Compression meets the network. *IEEE Signal Processing Magazine*, 18(5):74–93, September 2001.

[120] Graphics & Media laboratory of Computer Science department of MSU. http://graphics.cs.msu.ru, 2007.

[121] Groove home page. http://www.groove.net, 2005.

[122] W. Gropp and J. Mor'e. Optimization environments and the neos server. approximation theory and optimization. Technical report, M. D. Buhmann and A. Iserles, eds.Cambridge University Press, 1997.

[123] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 314–329, New York, NY, USA, 2003. ACM.

[124] James Gwertzman and Margo I. Seltzer. World wide web cache consistency. In *USENIX Annual Technical Conference*, pages 141–152, 1996.

[125] D. Hands and M. Wilkins. A Study of the Impact of Network Loss and Burst Size on Video Streaming Quality and Acceptability. In *Interactive Distributed Multimedia Systems and Telecommunication Services Workshop*, October 1999.

[126] Mohamed Hefeeda, Ahsan Habib, Boyan Botev, Dongyan Xu, and Bharat Bhargava. Promise: peer-to-peer media streaming using collectcast. In *MULTIMEDIA'03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 45–54, New York, NY, USA, 2003. ACM Press.

[127] Mohamed Hefeeda, Ahsan Habib, Boyan Botev, Dongyan Xu, and Bharat Bhargava. Collectcast: A peer-to-peer service for media streaming. *Multimedia Systems*, 11(1):68–81, November 2005.

[128] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross. Insights into pplive: A measurement study of a large-scale p2p iptv system. In *In Proc. of IPTV Workshop, International World Wide Web Conference*, 2006.

[129] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 146–159, New York, NY, USA, 2001. ACM.

[130] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185, New York, NY, USA, 1999. ACM.

[131] Geoffrey E. Hinton and Terrence Joseph Sejnowski. *Unsupervised Learning: Foundations of Neural Computation.* MIT Press, June 1999.

[132] Vegard Holmedahl, Ben Smith, and Tao Yang. Cooperative caching of dynamic content on a distributed web server. In *HPDC '98: Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing*, page 243, Washington, DC, USA, 1998. IEEE Computer Society.

[133] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. pages 389–407, 1992.

[134] Yang hua Chu, Aditya Ganjam, T. S. Eugene Ng, Sanjay G. Rao, Kunwadee Sripanidkulchai, Jibin Zhan, and Hui Zhang. Early experience with an internet broadcast system based on overlay multicast. In *ATEC'04: Proceedings of the USENIX Annual Technical Conference 2004 on USENIX Annual Technical Conference*, pages 12–12, Berkeley, CA, USA, 2004. USENIX Association.

[135] Yang hua Chu, Sanjay G. Rao, Srinivasan Seshan, and Hui Zhang. Enabling conferencing applications on the internet using an overlay multicast architecture. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 55–68, 2001.

[136] Yang hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast. In *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 1–12, New York, NY, USA, 2000. ACM Press.

[137] Qi Huang, Hai Jin, and Xiaofei Liao. P2p live streaming with tree-mesh based hybrid overlay. In *ICPPW '07: Proceedings of the 2007 International Conference on Parallel*

Processing Workshops (ICPPW 2007), page 55, Washington, DC, USA, 2007. IEEE Computer Society.

[138] A. Hughes and J. Touch. Cross-domain cache cooperation for small clients. In *In Proceedings of NetStore '99*, 1999.

[139] A. Hughes and J. Touch. Expanding the scope of prefetching through inter-application cooperation. In *Proceedings of the Sixth International Web Content Caching and Distribution Workshop (WCW'01)*, 2001.

[140] Scott Hull. *Content Delivery Networks*. McGraw-Hill., New York, February 2002.

[141] ICQ home page. http://www.icq.com/, 2005.

[142] IETF Network Working Group. User Datagram Protocol (RFC 768), August 1980.

[143] IETF Network Working Group. Internet Domain Name System Standard: Domain names - concepts and facilities (RFC 1034), 1987.

[144] IETF Network Working Group. Internet Domain Name System Standard: Domain names - implementation and specification (RFC 1035), 1987.

[145] IETF Network Working Group. The MD5 Message-Digest Algorithm (RFC 1321), April 1992.

[146] IETF Network Working Group. A Transport Protocol for Real-Time Applications (RFC 1889), January 1996.

[147] IETF Network Working Group. Conformance Statements for SMIv2 (RFC 2580), April 1999.

[148] IETF Network Working Group. Hypertext Transfer Protocol – HTTP/1.1 (RFC 2616), June 1999.

[149] IETF Network Working Group. Structure of Management Information Version 2 (RFC 2578), April 1999.

[150] IETF Network Working Group. Textual Conventions for SMIv2 (RFC 2579), April 1999.

[151] IETF Network Working Group. Notification Log MIB (RFC 3014), November 2000.

[152] IETF Network Working Group. Remote Network Monitoring Management Information Base (RFC 2819), May 2000.

[153] IETF Network Working Group. IANA Guidelines for IPv4 Multicast Address Assignments (RFC 3171), August 2001.

[154] IETF Network Working Group. IP Multicast Applications: Challenges and Solutions (RFC 3170), September 2001.

[155] IETF Network Working Group. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks (RFC 3411), December 2002.

[156] IETF Network Working Group. Internet Group Management Protocol, Version 3 (RFC 3376), October 2002.

[157] IETF Network Working Group. Management Information Base (MIB) for the Simple Network Management Protocol (SNMP) (RFC 3418), Dic 2002.

[158] IETF Network Working Group. An Overview of Source-Specific Multicast (SSM) (RFC 3569), July 2003.

[159] IETF Network Working Group. Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised) (RFC 3973), January 2005.

[160] IETF Network Working Group. The Atom Syndication Format (RFC 4287), December 2005.

[161] IETF Network Working Group. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised) (RFC 4601), August 2006.

[162] IETF Network Working Group. Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast (RFC 4604), August 2006.

[163] IETF Network Working Group. The Atom Publishing Protocol (RFC 5023), October 2007.

[164] Institut für Rundfunktechnik. http://www.irt.de, 2007.

[165] Interior Point OPTimizer (I-P-Opt) home page. http://projects.coin-or.org/Ipopt, 2007.

[166] iRate website. http://irate.sourceforge.net/, 2007.

[167] ISO/IEC 13818 Standard - MPEG-2. Information technology: generic coding of moving pictures and associated audio information, January 2005.

[168] ISO/IEC 14496-10 Standard - MPEG-4 Part 10. Advanced video coding, 2003.

[169] ISO/IEC 14496-2 Standard - MPEG-4 Part 2. Coding of audio-visual objects - part 2: Visual, 2001.

[170] ITU-R Document 6Q/57-E. Draft new recommendation for subjective assessment of streaming multimedia images by non-expert viewers, April 2004.

[171] ITU-R Recommendation BT.500-11. Methodology for the subjective assessment of the quality of television pictures, June 2002.

[172] ITU-T Recommendation G.107. The E-model, a Computational Model for Use in Transmission Planning, 2003.

[173] ITU-T Recommendation P.800. Methods for Subjective Determination of Transmission Quality, 1996.

[174] ITU-T Recommendation P.910. Subjective video quality assessment methods for multimedia applications, 2000.

[175] ITU-T Recommendation P.920. Interactive test methods for audiovisual communications, 2000.

[176] ITU-T Standard H.261. Video codec for audiovisual services at px64 kbit/s, January 1993.

[177] ITU-T Standard H.263. Video coding for low bit rate communication, January 1998.

[178] Villy Bæk Iversen. Teletraffic engineering handbook, draft-version. Technical report, ITC, in cooperation with ITU-D SG2, February 2005.

[179] S. Iyer, A. Rowstron, and P. Druschel. Squirrel: A decentralized peer-to-peer web cache. In *In Proc. 21st ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, 2002.

[180] Jabber home page. http://www.jabber.com/, 2005.

[181] Java Anon Proxy home page. http://anon.inf.tu-dresden.de/, 2005.

[182] Sylvie Jeannin and Ajay Divakaran. Mpeg-7 visual motion descriptors. *IEEE Trans. Circuits Syst. Video Techn.*, 11(6):720–724, 2001.

[183] Ingjerd Straand Jevnaker. Rawflow - using p2p to create virtual “multicasts”. Technical report, European Broadcasting Union (EBU), 2006.

[184] K. L. Johnson, J. F. Carr, M. S. Day, and Kaashoek M. F. The measured performance of content distribution networks. In *Proceedings of the 5th International Web Caching and Content Delivery Workshop*, 2000.

[185] Joost Home page. http://www.joost.com/, 2007.

[186] M. Jovanovic, F. Annexstein, and K. Berman. Scalability issues in large peer-to-peer networks - a case study of gnutella. Technical report, University of Cincinnati, January 2001.

[187] jumpTV Home page. http://www.jumptv.com/, 2007.

[188] J. Jung, A.W. Berger, and Hari Balakrishnan. Modeling ttl-based internet caches. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 1:417–426 vol.1, 30 March-3 April 2003.

[189] Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. Dns performance and the effectiveness of caching. *IEEE/ACM Trans. Netw.*, 10(5):589–603, 2002.

[190] KaZaA home page. http://www.kazaa.com, 2007.

[191] S. Kim and S.-U. Lee. Multiple description motion coding algorithm for robust video transmission. *IEEE Int. Symp. Circuits and Systems. Lausanne, Switzerland, vol. 4, pp. 717-720.*, May 2000.

[192] Kontiki website. http://www.kontiki.com/, 2007.

[193] L. Kontothanassis, R. Sitaraman, J. Wein, D. Hong, R. Kleinberg, B. Mancuso, D. Shaw, and D. Stodolsky. A Transport Layer for Live Streaming in a Content Delivery Network. In *Proceedings of the IEEE, Special Issue on Evolution of Internet Technologies*, pages 92(9):1408–1419, sep 2004.

[194] Franz Kozamernik, Paola Sunna, Emmanuel Wyckens, and Dag Inge Pettersen. Subjective quality of internet video codecs - phase 2 evaluations using samviq. Technical report, EBU Technical Review, 2005.

[195] Balachander Krishnamurthy and Craig E. Wills. Study of piggyback cache validation for proxy caches in the world wide web. In *USENIX Symposium on Internet Technologies and Systems*, 1997.

[196] Balachander Krishnamurthy and Craig E. Wills. Piggyback server invalidation for proxy cache coherency. In *WWW7: Proceedings of the seventh international conference on World Wide Web 7*, pages 185–193, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[197] Balachander Krishnamurthy and Craig E. Wills. Proxy cache coherency and replacement - towards a more complete picture. In *ICDCS '99: Proceedings of the 19th IEEE International Conference on Distributed Computing Systems*, page 332, Washington, DC, USA, 1999. IEEE Computer Society.

[198] John Kubiatowicz, David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westly Weimer, Christopher Wells, and Ben Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS*. ACM, November 2000.

[199] H. Kung. Motusnet: A content network. Technical report, Harvard University, 2001.

[200] H. Kung and C. Wu. Content networks: Taxonomy and new approaches. In Kihong Park and Walter Willinger, editors, *The Internet as a Largescale Complex System*, Santa Fe Institute Series. Oxford university Press, 2002.

[201] LANCELOT Home Page. http://www.cse.clrc.ac.uk/Activity/LANCELOT/, 2007.

[202] Last.fm website. http://www.last.fm, 2007.

[203] I. Lazar and W. Terrill. Exploring content delivering networking. Technical report, IT Professional, aug 2001.

[204] W.S. Lee, M.R. Pickering, M.R. Frater, and J.F. Arnold. A robust codec for transmission of very low bit-rate video over channels with bursty errors. *IEEE Trans. Circuits Syst. Video Technol., vol. 10, pp. 1403-1412.*, Dec 2000.

[205] J. Li, B. Loo, J. Hellerstein, F. Kaashoek, D. Karged, and R. Morris. On the feasibility of peer-to-peer web indexing and search. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, volume 2735/2003, February 2003.

[206] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng. Anysee: Peer-to-peer live streaming. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–10, 2006.

[207] Aristidis Likas and Andreas Stafylopatis. Training the Random Neural Network Using Quasi-Newton Methods. *European Journal of Operations Research*, 126(2):331–339, 2000.

[208] Boon Thau Loo, Joseph M. Hellerstein, Ryan Huebsch, Scott Shenker, and Ion Stoica. Enhancing p2p file-sharing with an internet-scale query processor. In *vldb'2004: Proceedings of the Thirtieth international conference on Very large data bases*, pages 432–443. VLDB Endowment, 2004.

[209] Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE*, pages 72–93, 2005.

[210] Lugdunum home page. http://lugdunum2k.free.fr, 2007.

[211] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proceedings of the 16th annual ACM international conference on Supercomputing*, pages 84–95, New York, NY, USA, 2002. ACM.

[212] Evangelos P. Markatos. On caching search engine query results. *Computer Communications*, 24(2):137–143, 2001.

[213] Marcelo Martínez, Alexis Morón, Franco Robledo, Pablo Rodríguez-Bocca, Héctor Cancela, and Gerardo Rubino. A GRASP algorithm using RNN for solving dynamics in a P2P live video streaming network. Submitted.

[214] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65, London, UK, 2002. Springer-Verlag.

[215] Milan Milenkovic, Scott H. Robinson, Rob C. Knauerhase, David Barkai, Sharad Garg, Vijay Tewari, Todd A. Anderson, and Mic Bowman. Toward internet distributed computing. *IEEE Computer*, 36(5):38–46, 2003.

[216] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-peer computing. Technical Report HPL-2002-57, HP Labs, 2002.

[217] Jeffrey C. Mogul. Hinted caching in the web. In *EW 7: Proceedings of the 7th workshop on ACM SIGOPS European workshop*, pages 103–108, New York, NY, USA, 1996. ACM.

[218] S. Mohamed. *Automatic Evaluation of Real-Time Multimedia Quality: a Neural Network Approach*. PhD thesis, INRIA/IRISA, Univ. Rennes I, Rennes, France, jan 2003.

[219] S. Mohamed, F. Cervantes, and H. Afifi. Audio Quality Assessment in Packet Networks: an Inter–Subjective Neural Network Model. In *Proc. of IEEE International Conference on Information Networking (ICOIN-15)*, pages 579 –586, Beppu City, Oita, Japan, January 2001.

[220] S. Mohamed, F. Cervantes, and H. Afifi. Integrating Networks Measurements and Speech Quality Subjective Scores for Control Purposes. In *Proceedings of IEEE INFOCOM'01*, pages 641–649, Anchorage, AK, USA, April 2001.

[221] S Mohamed and G. Rubino. A Study of Real–time Packet Video Quality Using Random Neural Networks. *IEEE Transactions On Circuits and Systems for Video Technology*, 12(12):1071 –1083, December 2002.

[222] S. Mohamed, G. Rubino, H. Afifi, and F. Cervantes. Real–time Video Quality Assessment in Packet Networks: A Neural Network Model. In *Proceedings of International Conference on Parallel and Distibuted Processing Techniques and Applications (PDPTA'01)*, Las Vegas, Nevada, USA, June 2001.

[223] S. Mohamed, G. Rubino, and M. Varela. A method for quantitative evaluation of audio quality over packet networks and its comparison with existing techniques. In *Proceedings of the Measurement of Speech and Audio Quality in Networks workshop (MESAQIN'04)*, Prague, Czech Republic, June 2004.

[224] S. Mohamed, G. Rubino, and M. Varela. Performance evaluation of real-time speech through a packet network: a Random Neural Networks-based approach. *Performance Evaluation*, 57(2):141–162, May 2004.

[225] msnTV website. http://www.msntv.com, 2007.

[226] B. A. Murtagh and M. A. Saunders. Minos 5.4 user's guide. report sol 83-20r. Technical report, Systems Optimization Laboratory, Stanford University, February 1995.

[227] musicSearch website. http://search.mercora.com, 2007.

[228] MyTVPal Home page. Free Internet TV. http://www.mytvpal.com/, 2007.

[229] BIND Berkeley Internet name domain. Internet Software Consortium. http://www.isc.org/products/BIND, 2004.

[230] Napster home page. http://www.napster.com/, 2005.

[231] R. Narashima, A. Savakis, R. M. Rao, and R. De Queiroz. Key frame extraction using mpeg-7 motion descriptors. *Signals, Systems and Computers, 2003. Conference Record of the Thirty-Seventh Asilomar Conference on*, 2:1575–1579 Vol.2, 9-12 Nov. 2003.

[232] Michael N. Nelson, Brent B. Welch, and John K. Ousterhout. Caching in the sprite network file system. *ACM Trans. Comput. Syst.*, 6(1):134–154, 1988.

[233] T. Nguyen and A. Zakhor. Distributed video streaming with forward error correction. In *Proc. Packet Video Workshop, Pittsburgh, USA*, April 2002.

[234] T. P. Nguyen and A. Zakhor. Distributed video streaming over Internet. In *Proc. SPIE, Multimedia Computing and Networking.*, volume 4673, pages 186–195, December 2001.

[235] E. Nicholas. The economics of networks. In *International Journal of Industrial Organization*, volume 14, pages 673–699, 1996.

[236] E. Nicholas and C. Himmelberg. Critical mass and network evolution in telecommunications. In *Telecommunications Policy Research Conference, Gerard Brock (ed.)*, 1995.

[237] L. Nielsen. A Neural Network Model for Prediction of Sound Quality. Technical report, Technical University of Denmark, 1993.

[238] Octoshape Home page. http://www.octoshape.com/, 2007.

[239] JungHwan Oh and Praveen Sankuratri. Computation of motion activity descriptors in video segments. In *Int. Conf. on Multimedia, Internet and Video Technologies*, ICOMIV 2002, Skiathos, Greece, September 25-28, 2002.

[240] OpenCola home page. http://www.opencola.com/, 2005.

[241] M. Tamer Özsu, Kaladhar Voruganti, and Ronald C. Unrau. An asynchronous avoidance-based cache consistency algorithm for client caching dbmss. In *VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 440–451, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[242] Albitz P. and Liu C. *DNS and BIND, 4th Edition.* O'Reilly, New York, April 2001.

[243] Venkata N. Padmanabhan and Kunwadee Sripanidkulchai. The case for cooperative networking. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 178–190, London, UK, 2002. Springer-Verlag.

[244] Venkata N. Padmanabhan, Helen J. Wang, and Philip A. Chou. Resilient peer-to-peer streaming. In *ICNP '03: Proceedings of the 11th IEEE International Conference on Network Protocols*, page 16, Washington, DC, USA, 2003. IEEE Computer Society.

[245] Venkata N. Padmanabhan, Helen J. Wang, and Philip A. Chou. Supporting heterogeneity and congestion control in peer-to-peer multicast streaming. In *IPTPS '04: Proceedings of the Third International Workshop on Peer-to-Peer Systems*, page 6, San Diego, CA, USA, 2004. International Workshop on Peer-to-Peer Systems.

[246] Venkata N. Padmanabhan, Helen J. Wang, Philip A. Chou, and Kunwadee Sripanidkulchai. Distributing streaming media content using cooperative networking. In *NOSS-DAV '02*, pages 177–186, New York, NY, USA, 2002. ACM Press.

[247] Digital Island Home page. http://www.digitalisland.net, 2007.

[248] Exodus Home page. http://www.exodus.com, 2007.

[249] Onion Tornado-Cache Home page. http://onionnetworks.com/, 2007.

[250] Open Content Network Home page. http://www.open-content.net, 2007.

[251] Gopal Pandurangan, Prabhakar Raghavan, and Eli Upfal. Building low-diameter p2p networks. In *FOCS '01: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, page 492, Washington, DC, USA, 2001. IEEE Computer Society.

[252] Peercast home page. http://www.peercast.org/, 2007.

[253] K. A. Peker, A. A. Alatan, and A. N. Akansu. Low-level motion activity features for semantic characterization of video. *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, 2:801–804 vol.2, 2000.

[254] PENalty method for NONlinear and semidefinite programming (PENNON) Home Page. http://www2.am.uni-erlangen.de/ kocvara/pennon/, 2007.

[255] Dimitrios Pendarakis, Sherlia Shi, Dinesh Verma, and Marcel Waldvogel. ALMI: An application level multicast infrastructure. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS)*, pages 49–60, 2001.

[256] Guillaume Pierre and Maarten van Steen. Globule: A platform for self-replicating Web documents. In *PROMS 2001: Proceedings of the 6th International Conference on Protocols for Multimedia Systems*, pages 1–11, London, UK, 2001. Springer-Verlag.

[257] Guillaume Pierre and Maarten van Steen. Design and implementation of a user-centered content delivery network, June 2003.

[258] Guillaume Pierre, Maarten van Steen, and Andrew S. Tannenbaum. Dynamically selecting optimal distribution strategies for web documents. *IEEE Trans. Comput.*, 51(6):637–651, 2002.

[259] PlanetLab. An open platform for developing, deploying, and accessing planetary-scale services. http://www.planet-lab.org, 2007.

[260] PPLive Home page. http://www.pplive.com, 2007.

[261] PPStream home page. http://www.ppstream.com/, 2007.

[262] Lili Qiu, Venkata N. Padmanabhan, and Geoffrey M. Voelker. On the placement of web server replicas. In *INFOCOM'01. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1587–1596, Anchorage, AK, USA, 2001.

[263] Fourer R., D.M Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming.* Duxbury Press / Brooks/Cole Publishing Company, 2002.

[264] Michael Rabinovich and Oliver Spatschek. *Web caching and replication.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[265] Radio Paradise website. http://www.radioparadise.com/, 2007.

[266] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM Press.

[267] RawFlow home page. http://www.rawflow.com/, 2007.

[268] RealNetworks home page. http://www.realnetworks.com/, 2007.

[269] Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[270] M. G. C. Resende and C. C. Ribeiro. *Greedy Randomized Adaptive Search Procedures.* In F. Glover and G. Kochenberger, editors, Handbook of Methaheuristics, Kluwer Academic Publishers, 2003.

[271] Martin Varela Rico. *Pseudo-subjective Quality Assessment of Multimedia Streams and its Applications in Control.* PhD thesis, INRIA/IRISA, Univ. Rennes I, Rennes, France, nov 2005.

[272] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1), 2002.

[273] Pablo Rodriguez and Ernst W. Biersack. Dynamic parallel access to replicated content in the internet. *IEEE/ACM Trans. Netw.*, 10(4):455–465, 2002.

[274] Pablo Rodríguez-Bocca. "Redes de Contenido: Taxonomía y Modelos de evaluación y diseño de los mecanismos de descubrimiento de contenido.".

[275] Pablo Rodríguez-Bocca. Exploring Quality Issues in Multi-Source Video Streaming. In *2nd Beijing-Hong Kong International Doctoral Forum: Web, Network, and Media Computing (BJ-HK Phd Forum'07). Best Paper Award in the area of Network Multimedia Applications and Systems.*, Hong Kong, SAR, July 3-6 2007.

[276] Pablo Rodríguez-Bocca and Héctor Cancela. Introducción a las Redes de Contenido. In *Jornadas de Informática e Investigación Operativa (JIIO'04)*, Montevideo, Uruguay, November 8–12 2004.

[277] Pablo Rodríguez-Bocca and Héctor Cancela. Redes de contenido: un panorama de sus características y principales aplicaciones. In *33th Argentine Conference on Computer Science and Operational Research, 5th Argentine Symposium on Computing Technology (JAIIO'04)*, Córdoba - Argentina, September 20-24 2004.

[278] Pablo Rodríguez-Bocca and Héctor Cancela. A mathematical programming formulation of optimal cache expiration dates in content networks. In *Proceedings of the 3rd international IFIP/ACM Latin American conference on Networking (LANC'05)*, pages 87–95, New York, NY, USA, October 10-14 2005. ACM Press.

[279] Pablo Rodríguez-Bocca and Hector Cancela. Mecanismos de descubrimiento en las Redes de Contenido. In *XIII Congreso Latino-Iberoamericano de Investigación Operativa (CLAIO'06)*, Universidad de la República, Montevideo, Uruguay, November 27-30 2006.

[280] Pablo Rodríguez-Bocca and Hector Cancela. Modeling cache expiration dates policies in content networks. In *International Network Optimization Conference (INOC'07)*, Spa, Belgium., April 22-25 2007.

[281] Pablo Rodríguez-Bocca, Hector Cancela, and Gerardo Rubino. Modeling Quality of Experience in Multi-source Video Streaming. In *4th Euro-FGI workshop on "New trends in modelling, quantitative methods and measurements" (Euro-FGI WP IA.7.1)*, Ghent, Belgium, May 31 - June 1 2007.

[282] Pablo Rodríguez-Bocca, Hector Cancela, and Gerardo Rubino. Perceptual quality in P2P video streaming policies. In *50th IEEE Global Telecommunications Conference (GLOBECOM'07)*, Washington DC, United States, 26-30 November 2007.

[283] Pablo Rodríguez-Bocca, Hector Cancela, and Gerardo Rubino. Video Quality Assurance in Multi-Source Streaming Techniques. In *IFIP/ACM Latin America Networking Conference (LANC'07). Best paper award by ACM-SIGCOMM*, San José, Costa Rica, October 10-11 2007.

[284] Pablo Rodríguez-Bocca and Gerardo Rubino. A QoE-based approach for optimal design of P2P video delivering systems. Submitted.

[285] Pablo Rodríguez-Bocca, Gerardo Rubino, and Luis Stábile. Multi-Source Video Streaming Suite. In *7th IEEE International Workshop on IP Operations and Management (IPOM'07)*, San José, California, United States, October 31 - November 2 2007.

[286] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–347, 2001.

[287] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[288] Antony Rowstron and Peter Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 188–201, New York, NY, USA, 2001. ACM.

[289] Antony Rowstron and Peter Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 188–201, New York, NY, USA, 2001. ACM Press.

[290] Roxbeam Nedia Network Corp. home page. http://www.roxbeam.com, 2007.

[291] D. Rubenstein, J. Kurose, and D.F. Towsley. Detecting shared congestion of flows via end-to-end measurement. *IEEE Trans. on Networking*, 3, June 2003.

[292] G. Rubino, P. Tirilly, and M. Varela. Evaluating users' satisfaction in packet networks using Random Neural Networks. In *16th International Conference on Artificial Neural Networks (ICANN'06)*, Athens, Greece, September 2006.

[293] G. Rubino and M. Varela. Evaluating the utility of media–dependent FEC in VoIP flows. In *LNCS 3266: Proceedings of the Fifth International Workshop on Quality of future Internet Services (QofIS'04)*, Barcelona, Spain, September 2004.

[294] G. Rubino, M. Varela, and J-M. Bonnin. Controlling multimedia QoS in the future home network using the PSQA metric. *submitted to IEEE Trans. on Systems Man and Cybernetics part B*, 2005.

[295] Gerardo Rubino and Martín Varela. A New Approach for the Prediction of End-to-End Performance of Multimedia Streams. In *Proceedings of the First International Conference on Quantitative Evaluation of Systems (QEST'04)*, Enschede, The Netherlands, September 2004.

[296] Gerardo Rubino, Martín Varela, and Jean-Marie Bonnin. Wireless VoIP at Home: Are We There Yet? In *Proceedings of the Measurement of Speech and Audio Quality in Networks workshop (MESAQIN'05)*, Prague, Czech Republic, June 2005.

[297] Osokine S. Search optimization in the distributed networks, 2002.

[298] H. Sanneck, G. Carle, and R. Koodli. A Framework Model for Packet Loss Metrics Based on Loss Runlengths. In *Proceedings of the SPIA/ACM SIGMM Multimedia Computing and Networking Conference*, pages 177–187, San Jose, CA, January 2000.

[299] S. Saroiu, P. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems, 2002.

[300] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Multimedia Computing and Networking*, 2002.

[301] Peter Scheuermann, Junho Shim, and Radek Vingralek. Watchman: A data warehouse intelligent cache manager. In *VLDB '96: Proceedings of the 22th International Conference on Very Large Data Bases*, pages 51–62, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[302] Subhabrata Sen and Jia Wang. Analyzing peer-to-peer traffic across large networks. In *IEEE/ACM Transactions on Networking*, volume 12, pages 219– 232, April 2004.

[303] A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, and S. Tewari. Will iptv ride the peer-to-peer stream? *Communications Magazine, IEEE*, 45:86–92, 2007.

[304] S.D. Servetto and K. Nahrstedt. Broadcast-quality video over ip. *IEEE Trans. Multimedia, vol. 3, pp. 162-173.*, Mar 2001.

[305] Seti@home home page. http://setiathome.ssl.berkeley.edu., 2005.

[306] SevernStream home page. http://www.severnstream.com/, 2007.

[307] J. Shim, P. Scheuermann, and R. Vingralek. Proxy cache algorithms: design, implementation, and performance. *Transactions on Knowledge and Data Engineering*, 11(4):549–562, Jul/Aug 1999.

[308] Clay Shirky. What is p2p...and what isn't? Technical report, O'Reilly openP2P Network, 2000.

[309] Siemens AG. http://www.siemens.com/, 2007.

[310] E. Sit. A study of caching in the internet domain name system, 2000.

[311] Swaminathan Sivasubramanian, Michal Szymaniak, Guillaume Pierre, and Maarten van Steen. Replication for web hosting systems. *ACM Computing Surveys*, 36(3):291–334, 2004.

[312] SopCast - Free P2P internet TV. http://www.sopcast.org, 2007.

[313] Squid Web Proxy Cache Project. http://www.squid-cache.org/, 2005.

[314] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. An analysis of live streaming workloads on the internet. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 41–54, New York, NY, USA, 2004. ACM Press.

[315] Stanford Business Software's home page. http://www.sbsi-sol-optimize.com/, 2007.

[316] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM Press.

[317] StreamAudio website. http://www.chaincast.com, 2007.

[318] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1999.

[319] Tandberg Home page. http://www.tandberg.com/, 2007.

[320] S. Tewari and L. Kleinrock. Proportional replication in peer-to-peer network. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12. IEEE, 2006.

[321] S. Tewari and L. Kleinrock. Analytical model for bittorrent-based live video streaming. In *4th IEEE Consumer Communications and Networking Conference (CCNC'07)*, pages 976–980, Las Vegas, NV, January 2007.

[322] The Freenet Project home page. http://freenetproject.org, 2007.

[323] Julio Orozco Torrentera. *Quality of Service management of multimedia flows over DiffServ IP networks*. PhD thesis, INRIA/IRISA, Univ. Rennes I, Rennes, France, apr 2005.

[324] Duc A. Tran, Kien A. Hua, and Tai T. Do. Zigzag: An efficient peer-to-peer scheme for media streaming. In *INFOCOM*, 2003.

[325] TVAnts home page. http://cache.tvants.com/, 2007.

[326] TVUnetworks home page. http://tvunetworks.com/, 2007.

[327] A. Vahedian, M.R. Frater, and J.F. Arnold. Impact of Audio on Subjective Assessment of Video Quality. In *Proceedings of International Conference on Image Processing*, volume 2, pages 367–370, Kobe, Japan, October 1999.

[328] C.J. van den Branden Lambrecht. Color Moving Picture Quality Metric. In *Proceedings of the IEEE International Conference on Image Processing*, September 1996.

[329] C.J. van den Branden Lambrecht. *Perceptual Models and Architectures for Video Coding Applications*. PhD thesis, EPFL, Lausanne, Swiss, 1996.

[330] N. Vasconcelos and A. Lippman. Towards semantically meaningful feature spaces for the characterization of video content. In *ICIP '97: Proceedings of the 1997 International Conference on Image Processing (ICIP '97) 3-Volume Set-Volume 1*, page 25, Washington, DC, USA, 1997. IEEE Computer Society.

[331] Daniel De Vera, Pablo Rodríguez-Bocca, and Gerardo Rubino. QoE Monitoring Platform for Video Delivery Networks. In *7th IEEE International Workshop on IP Operations and Management (IPOM'07)*, San José, California, United States, October 31 - November 2 2007.

[332] Dinesh C. Verma. *Content Distribution Networks*. John Wiley & Sons, Inc., New York, February 2002.

[333] Video Quality Experts Group (VQEG). http://www.its.bldrdoc.gov/vqeg/, 2007.

[334] VideoLan home page. http://www.videolan.org, 2007.

[335] VitalStream home page. http://www.vitalstream.com, 2007.

[336] A. Vlavianos, M. Iliofotou, and M. Faloutsos. Bitos: Enhancing bittorrent for supporting streaming applications. In *9th IEEE Global Internet Symposium 2006*, April 2006.

[337] Geoffrey M. Voelker, Eric J. Anderson, Tracy Kimbrel, Michael J. Feeley, Jeffrey S. Chase, Anna R. Karlin, and Henry M. Levy. Implementing cooperative prefetching and caching in a globally-managed memory system. *SIGMETRICS Perform. Eval. Rev.*, 26(1):33–43, 1998.

[338] S. Voran. The Development of Objective Video Quality Measures that Emulate Human Perception. In *IEEE GLOBECOM*, pages 1776–1781, 1991.

[339] vTuner website. http://www.vtuner.com/, 2007.

[340] Jia Wang. A survey of web caching schemes for the internet. *SIGCOMM Comput. Commun. Rev.*, 29(5):36–46, 1999.

[341] Yubing Wang. Survey of objective video quality measurements. Technical Report WPI-CS-TR-06-02, EBU Technical Review, February 2006.

[342] Z. Wang, L. Lu, and A. Bovik. Video quality assessment using structural distortion measurement. *Signal Processing: Image Communication, IEEE. Sspecial issue on "Objective video quality metrics"*, 19(2):121–132, February 2004.

[343] Zhou Wang and A. C. Bovik. A universal image quality index. *Signal Processing Letters, IEEE*, 9(3):81–84, March 2002.

[344] A. Watson and M.A. Sasse. Evaluating Audio and Video Quality in Low-Cost Multimedia Conferencing Systems. *ACM Interacting with Computers Journal*, 8(3):255–275, 1996.

[345] A. Watson and M.A. Sasse. Measuring Perceived Quality of Speech and Video in Multimedia Conferencing Applications. In *Proceedings of ACM Multimedia'98*, pages 55–60, Bristol, UK, September 1998.

[346] S. Wee, W. Tan, J. Apostolopoulos, and S. Roy. System design and architecture of a mobile streaming media content delivery network (msdcdn). Technical report, Streaming Media Systems Group, HP-Labs, 2003.

[347] Dan Werthimer, Jeff Cobb, Matt Lebofsky, David Anderson, and Eric Korpela. Seti@homemassively distributed computing for seti. *Comput. Sci. Eng.*, 3(1):78–83, 2001.

[348] Duane Wessels. Intelligent caching for World-Wide Web objects. In *Proceedings of the INET '95 conference*, Honolulu, Hawaï, 1995.

[349] Wayne Wolf. Key frame selection by motion analysis. In *ICASSP '96: Proceedings of the Acoustics, Speech, and Signal Processing, 1996. on Conference Proceedings., 1996 IEEE International Conference*, pages 1228–1231, Washington, DC, USA, 1996. IEEE Computer Society.

[350] Alec Wolman, M. Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry M. Levy. On the scale and performance of cooperative web proxy caching. In *SOSP '99: Proceedings of the seventeenth ACM symposium on Operating systems principles*, pages 16–31, New York, NY, USA, 1999. ACM.

[351] Chi-Jen Wu, Cheng-Ying Li, and Jan-Ming Ho. Improving the download time of bittorrent-like systems. In *IEEE International Conference on Communications 2007 (ICC 2007)*, Glasgow, Scotland, June 2007.

[352] Gang Wu and Tzi cker Chiueh. How efficient is bittorrent? In *Thirteenth Annual Multimedia Computing and Networking (MMCN'06)*, San Jose, CA., Jaunary 2006.

[353] Susu Xie, Gabriel Y. Keung, and Bo Li. A measurement of a large-scale peer-to-peer live video streaming system. In *ICPPW '07: Proceedings of the 2007 International Conference on Parallel Processing Workshops (ICPPW 2007)*, page 57, Washington, DC, USA, 2007. IEEE Computer Society.

[354] Xvid Codec home page. http://www.xvid.org/, 2007.

[355] Xing Y. Caching on the Changing Web.

[356] YahoOhome page. http://www.yahoo.com/, 2005.

[357] M. Yajnik, S. Moon, J.F. Kurose, and D.F. Towsley. Measurement and Modeling of the Temporal Dependence in Packet Loss. In *Proccedings of IEEE INFOCOM '99*, pages 345–352, 1999.

[358] Beverly Yang and Hector Garcia-Molina. Comparing hybrid peer-to-peer systems. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 561–570, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[359] Beverly Yang and Hector Garcia-Molina. Designing a super-peer network. In *19th International Conference on Data Engineering (ICDE'03)*, pages 49–60, Bangalore, India, March 2003.

[360] youTube website. http://www.youtube.com/, 2007.

[361] D. Zeinalipour-Yatzi and T. Folias. A quantitative analysis of the gnutella network traffic. Technical report, Department of Computer Science, University of California, Riverside, 2002.

[362] X. Zhang, J. Liu, and B. Li. On large scale peer-to-peer live video distribution: Coolstreaming and its preliminary experimental results. In *in IEEE International Workshop on Multimedia Signal Processing (MMSP'05)*, Washington, DC, USA, 2005. IEEE Computer Society.

[363] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing Peter Yum. Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming, 2005.

[364] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.

[365] Ziena Optimization Inc. KNitro Home Page. http://www.ziena.com/knitro.html, 2007.

List of Figures

Abstract

The use of Peer-to-Peer (P2P) networking is a scalable way to offer video services in the Internet. This document focuses on the definition, development and evaluation of a P2P architecture to deliver live video. The global design is driven by the Quality of Experience (QoE), instead of the traditional Quality of Service, through its main component in this area, the video quality perceived by final users. We extend the recently proposed Pseudo–Subjective Quality Assessment(PSQA) methodology to measure, in real time and automatically, the perceptual video quality in our context. Two main research lines are developed. First, we propose a multi-source streaming technique with extremely low signaling overhead (opposite to current existing systems). We describe a design method, based on PSQA, which allows a fine control of the way the video signal is split into substreams and of the amount of redundancy added, as a function of observed dynamics of network users. This way it is possible to improve the robustness of the system as much as desired, while controlling the impact of the communication overhead. Second, we present a tree-based structured overlay system to control the topology of the network. The selection of which peer will serve each other peer is important for the robustness of the network, especially when the peers are heterogeneous in their capability and connection time. Our design maximizes the global expected quality (evaluated using PSQA), selecting a connection topology which enhances this robustness property.

We also study how to extend the network with two complementary services: Video on Demand (VoD) and MyTV. The challenge here is how to make video libraries efficiently searchable, because of the high dynamics in the contents. We present a caching search strategy, for these services, that maximizes the total number of correct answers to video queries, given a particular contents' dynamics and bandwidth constraints.

Our global design focuses on a real scenario, where the test cases and the configuration parameters come from real data of an existing video delivery reference service. Our full-working prototype is a completely free-software application, based on well-known technologies which can be executed on various operating systems.