Video Quality Assurance in Multi-Source Streaming Techniques

Pablo Rodríguez-Bocca
Instituto de Computación
Facultad de Ingeniería
Universidad de la República
Julio Herrera y Reissig 565,
11300
Montevideo, Uruguay
prbocca@fing.edu.uy

Héctor Cancela
Instituto de Computación
Facultad de Ingeniería
Universidad de la República
Julio Herrera y Reissig 565,
11300
Montevideo, Uruguay
cancela@fing.edu.uy

Gerardo Rubino
Inria/Irisa
Campus universitaire de
Beaulieu
35042 Rennes, France
rubino@irisa.fr

ABSTRACT

In this paper we address the problem of designing a P2P system for distribution real-time video streams through the Internet. The main advantage of this approach is to use the available bandwidth unused by the set of machines connected to the network. The main difficulty is that these machines are typically highly dynamic, they continuously enter and leave the network. To deal with this problem, we explore a multi-source approach where the stream is decomposed into several flows sent by different peers to each client. To evaluate the impact of this approach on quality, we use PSQA, a recently proposed method that allows to obtain a good approximation of the quality as perceived by each client. In particular, we provide a variant of muti-source techniques using some redundancy in the signal, illustrating with real data how the methods allow to compensate efficiently the possible losses of frames due to peers leaving the system.

Keywords

Multi-source streaming, path diversity, quality-of-service, video quality, perceptual quality, PSQA.

1. INTRODUCTION

There is nowadays an increasing growth of multimedia systems present in the Internet. This is a consequence of the development of broadband accesses in residential users, together with the adoption by content providers of new business models. It has been observed that, roughly speaking, the content's volume doubles every year, while the demand is increased by a factor of three These systems have many different architectures, depending on their sizes and on the popularity of their contents. The majority of them have a traditional CDN (Content Delivery Network) structure [3, 23], 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 [17], YouTube [24], Jumptv [9], etc., all working with video content[1].

An increasingly popular alternative 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, which 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.

P2P 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 [2], KaZaA [10], eMule [4], etc.

The 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 dynamic, and thus, that 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.

In this paper, 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. Streaming services in VoD (Video on Demand) has similar characteristics. 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. Our work is then concerned with finding solutions to these problems.

As we said before, P2P systems are scalable, due to the high availability of unused resources of the users. The main

LANC'07 October 10-11, San José, Costa Rica

[1]In general, the technology used is derived from those associated with Web applications, based on the HyperText Transfer Protocol (HTTP) [8, 11, 19].

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 paper we explore one of them: 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 put on the senders that build a set of 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 opportunity 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) 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.

This flexibility must be carefully tuned in order to get a satisfactory quality level at a minimal cost. The usual approach here 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. In this paper we instead address the problem of measuring *perceived* quality by means of the PSQA technology [12, 14, 15]. PSQA is a general procedure that allows the automatic measure of the perceived quality, accurately and in real-time. We apply the technique to the case of multi-source streaming for live video, and improve its efficiency for video analysis by studying the flows at the frame level, instead of the packet level previously considered in the literature.

In order to face the high dynamics of such a system, 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.

The paper focuses then on the analysis of the impact on the perceived quality, as captured by the PSQA metric, of the fact that the stream is received from several nodes decomposed into different flows (explaining the term *multi-sourcing*). 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 extreme cases where the flows are just copies of the original sequence (a very high redundancy level), where the sequence is split into simple disjoint sub-streams (without redundancy at all), and where the sequence is split into redundant sub-streams for a $N + 1$ failure schema (i.e. when is possible to reconstruct completely the stream with only one server failure). After some modeling work needed for the development of a PSQA module able to compute the perceived quality in real-time, we do some experiments in order to explore the consequences of these architecture choices on the quality level, extending the preliminary results presented in a previous paper [18].

The paper is organized as follows. Section 2 introduces multi-source streaming techniques. Different video quality measurements are presented in Section 3, and in particular PSQA. In Section 4 we present a P2P network for live multi-source streaming, and a model of the nodes disconnection from the client's perspective. Section 5 develops models to estimate the streaming performance, in a environment with server nodes fails, needed to the construction of the PSQA measuring module. They calibrate the multi-source streaming on the basis of the perceptual quality. In Section 6 some experimental results are introduced. The main contributions of this work are then summarized in Section 7.

2. MULTI-SOURCE STREAMING

The main architecture we are considering in this paper is the following one. Some server producing a live video stream splits this stream into several 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.

The simplest situation is when there is a single server node which sends all the streaming information to the clients (see Fig. 1).

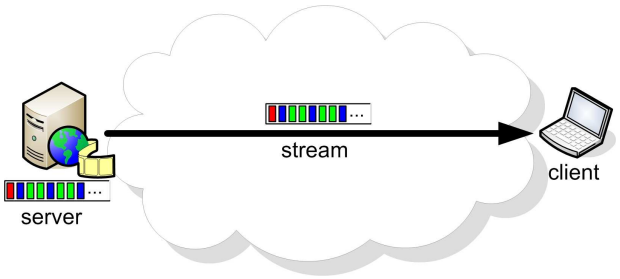

Figure 1: Single source streaming method

Let us consider instead the case where the server will send more than one flow composing the original signal. The quality of service perceived at the client node will be a function of the policy being used to distribute the streaming among the different flows, of the degree of redundancy, and of the loss rates and loss bursts due to transport network conditions or to instabilities at the P2P server nodes. An important complementary 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. That is, this is the full redundant scheme where the client receives many copies of the complete flow (Fig. 2).

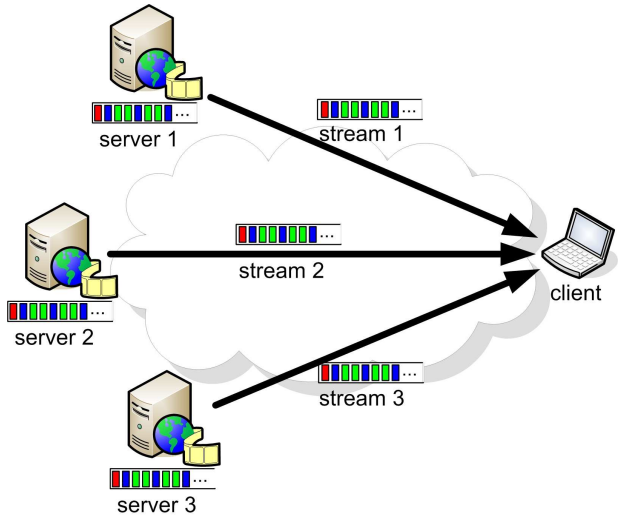

Figure 2: Multi-source streaming copy method

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. Fig. 3 represents this scheme. More precisely, we will consider the case of sending frame 1 in flow or substream 1, frame 2 in flow 2, up to frame K in flow K, then frame $K+1$ in flow 1, etc. (see below).

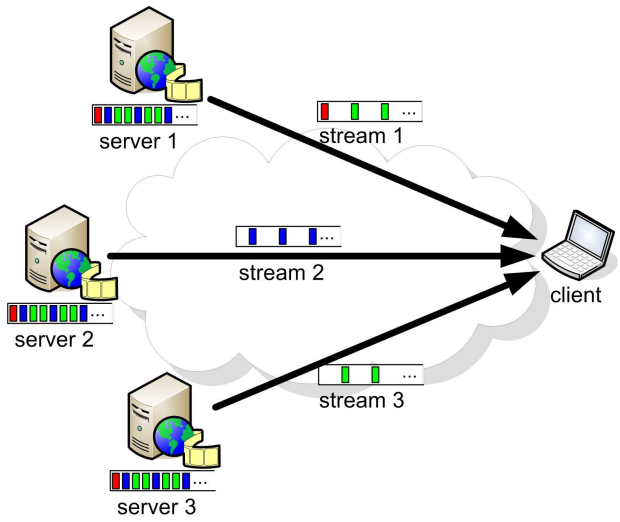

Figure 3: Multi-source streaming split method

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. Fig. 4 represents this scheme.

Of course, we can expect that in a fault context, 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.

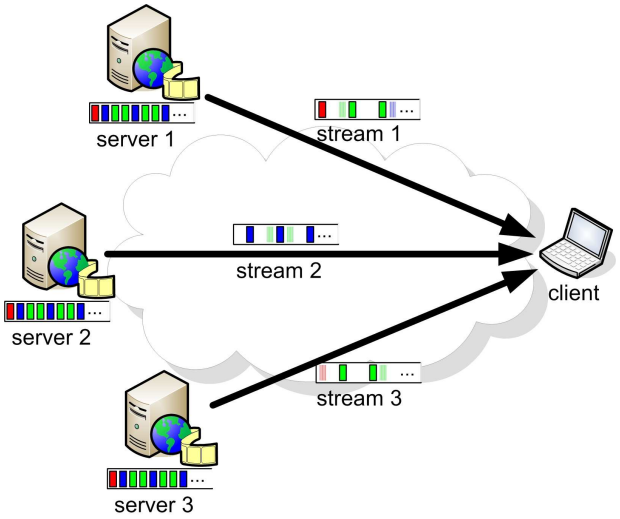

Figure 4: Multi-source streaming redundant method

Although in this work 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 Section 5 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 $MLBS$. The quantitative evaluation of these models not only can give some insights into QoS characteristics of multi-source streaming in a P2P network, but also can serve as bounds for the expected behavior of other policies with an intermediate replication level.

3. QUALITY MEASUREMENTS

This section discusses different ways of dealing with the perceived quality in a video delivering system.

3.1 Subjective tests

Perceived video quality is, by definition, a subjective concept. The mechanism used for assessing it is called *subjective testing*. It consists of building a panel with real human subjects, which will evaluate a series of short video sequences according to their own personal notion of 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). 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 streaming in real time, for controlling purposes). There exist standard methods for conducting subjective video quality evaluations, such as the ITU-R BT.500-11 [7]. Some variants included in the standard are: Double Stimulus Impairment Scale (DSIS), Double Stimulus Continuous Quality Scale (DSCQS), Single

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

3.2 Objective tests

Other solutions, called *objective tests*, have been proposed. Objective tests are algorithms and formulas that measure, in a certain way, the quality of a stream. Peek signal to noise ratio (PSNR), ITS' Video Quality Metric (VQM) [1, 22], EPFL's Moving Picture Quality Metric (MPQM), Color Moving Picture Quality Metric (CMPQM) [20, 21], and Normalization Video Fidelity Metric (NVFM) [21]. With some exceptions, the objetive metrics propose different ways of comparing the received sample with the original one, typically by computing a sort of distance between both signals. So, it is not possible to use them in a real-time passive test enviroment, because the received and the original video are needed at the same time in the same place. Besides, these quality metrics often provide assessments that do not correlate well with human perception, and thus their use as a replacement of subjective tests is limited.

3.3 Pseudo Subjective Quality Assessment (PSQA)

The Pseudo Subjective Quality Assessment (PSQA) [13] is a technique allowing to approximate the value obtained from a subjective test but automatically. The idea is to have several distorted samples evaluated subjectively, that is, 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 have been obtained using Random Neural Networks [5]) in order to capture the relation between the parameters that characterize the distortion and the perceived quality. This method produces good evaluations for a wide range variation of all the quality affecting parameters. For instance, in [13], the authors present evaluations with correlation coefficients with the evaluation done by human observers up to 0.97.

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 parameter values. 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. Each human observer evaluates many sequences and each sequence is evaluated by all the observers (as specified by an appropriate subjective test norm). After this subjective evaluation, we must apply 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 we apply the training process, which learns the mapping from the values of the set of parameters into quality. The output of this learning process is then a function able to build a quality value from the values of the parameters (with very low computational effort).

Fig. 5 represents graphically the whole process.

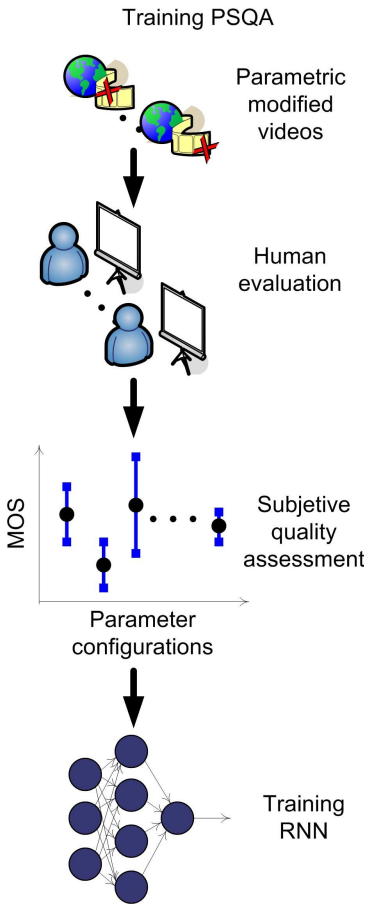

Figure 5: Training PSQA method

After training, PSQA is very easy to use: we measure the values of the chosen parameters at time t and then we use them as the inputs of the function returned by the RNN tool, which gives the *instantaneous* perceived quality at t. See Fig.6 where we represent PSQA in operation.

In this work, we focus on two specific parameters concerning losses, because previous works on PSQA have shown that the loss process is the most important global factor for quality. 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. 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 quality is more directly influenced by loss frames than by loss packets. Packet-level parameters are easier to handle (in the testbed and from the measuring point of view in the network), but frame-level ones provide a more accurate view of perceived quality.

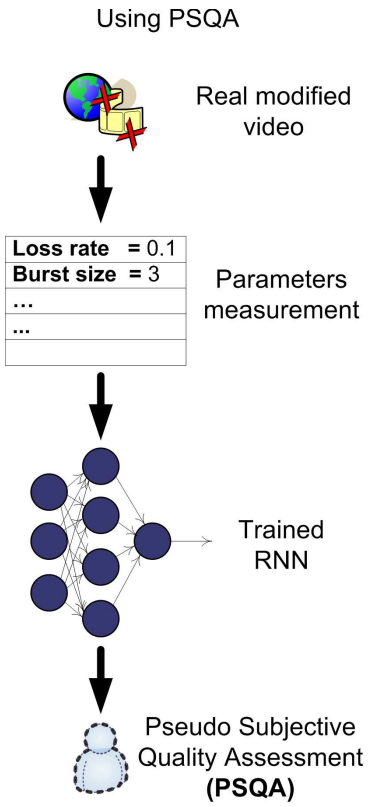

Figure 6: Using PSQA method

4. P2P NETWORK AND MODEL

Peer-to-Peer are virtual networks developed at the application level over the Internet infrastructure. The nodes in the network, called peers, share their resources (bandwidth, processing power, storing capacity) among themselves, basically because they have common interests (through the considered application, for instance a real-time football match video streaming).

Using a P2P infrastructure for real-time video distribution seems a good idea due to the high bandwidth requirements of these applications. However, real-time video streaming (live TV) has strong constraints that imply a series of specific technical problems principally because of the connections/disconnections dynamics of the peers. In a P2P network, the nodes 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 dynamic, and thus, that the network must be robust face to these fluctuations. It has been shown in a previous work [13], that video streaming quality, as perceived by the user, is specially sensitive to frame losses; this means that it is important that the design of the P2P architecture mitigates the impact of losses coming from node disconnections. We think that this is the main challenge in the design of a live-video P2P network: to offer the quality needed by the clients in a highly varying environment, and this is the topic discussed in this work.

To provide good quality levels in a context where this quality depends on the behavior of other clients that are delivering the stream, it is necessary to use some redundancy in the signals. In Section 2 we explained how to achieve this with multi-source streaming techniques. These streaming methods 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 multi-source streaming techniques discussed have different redundant information degree and can be potentially used with any amount of servers).

To decide which client will serve another one, some degree of intelligence and knowledge about the peers and the network state is needed. An important research effort is being done on this hot topic. 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. Thus, 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.

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 the live-video service offered by a medium-sized ISP, which gave us access to this information. We filtered the information of very short lived nodes, as 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 1 based on measured data.

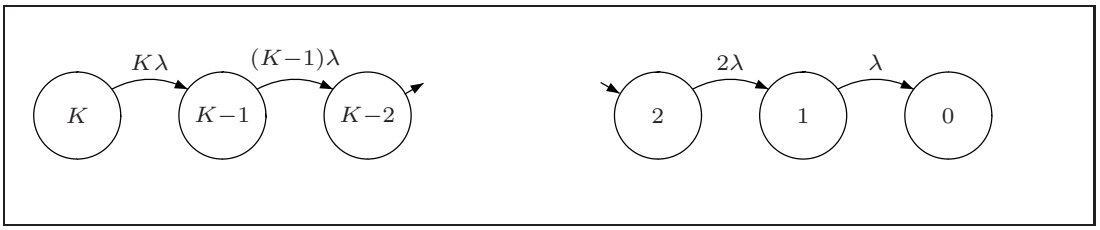

Figure 7: The Markovian model used to represent the evolution of the number of connected servers sending the stream to the same (arbitrary) client.

Table 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 (these are typical values).

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

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.

5. MULTI-SOURCE STREAMING MODELS

In this section we discuss some possible ways of sending the stream using K parallel servers, and the corresponding impact on the quality.

5.1 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 previous section. 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 servers multi-source, and i connected among them, we then have:

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

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

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

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

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

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 help getting a feeling of numerical values, we plot in Table 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,

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

5.3 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 some bandwidth of B Kbps, then we assume that each substream 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.

Table 2: Simple Split method. Global Loss Rate (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

Table 3: Split method with redundancy. Global Loss Rate, 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

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:

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

Some values of $LR_{K,i}^{split-red}$ are given in Table 3.

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

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

6. TESTING AND EXPERIMENTAL RESULTS

In this section we study how the frame loss process affects the quality (as measured by the PSQA technique) for the three multiple server streaming policies ("copy", "simple split" and "redundant split").

6.1 Pseudo Subjective Quality with Frame Losses

The first step was to apply the PSQA technique, as explained in Subsection 3.3. For this, we chose four MPEG2 video sequences, of about 10 seconds each, with sizes between 1.5 MB and 2.8 MB. For each sequence, we generated twenty five different evaluation points, where each evaluation point 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 10.0 (the actual process is a little bit more complex but this does not change the essential aspects of the method, see [13] for more details). For each of the evaluation points, we used the simplified Gilbert model 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.

The simplified Gilbert model [13, 16] consists of a 2-state Markov process that controls which frames are lost in the flow (so, with 2 parameters; the original Gilbert model has 3 parameters [6]). Figure 8 illustrates the dynamics of the Markov model; at the left, the semantics of transitions, and at the right, the Markov process (through its transition graph) itself. The two parameters are then

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

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

After generating the one hundred modified video sequences, a group of five experts evaluated the sequences and the MOS for each of the copies was computed, following the ITU-R BT.500-11 [7] norm (see Figure 9). These MOS were scaled into a quality metric in the range $[0, 1]$.

Finally, we employed the MOS value for each of the design points as inputs in order to calibrate a Random Neural Network (RNN). After trained and validated, the RNN provides a function of two variables, LR and $MLBS$, mapping them into perceived quality (on a $[0, 1]$ range). In Figure 10 we can see the obtained function. For completitude, we extrapolate the curve to the borders, but observe that the data are accurate and used on an internal region ([1%, 15%] for LR, and [1, 4] for the $MLBS$).

In particular, we can observe that quality is monotone in the two variables, and particularly increasing with the $MLBS$, meaning that humans prefer sequences where losses are concentrated over those where losses are spread through the flow.

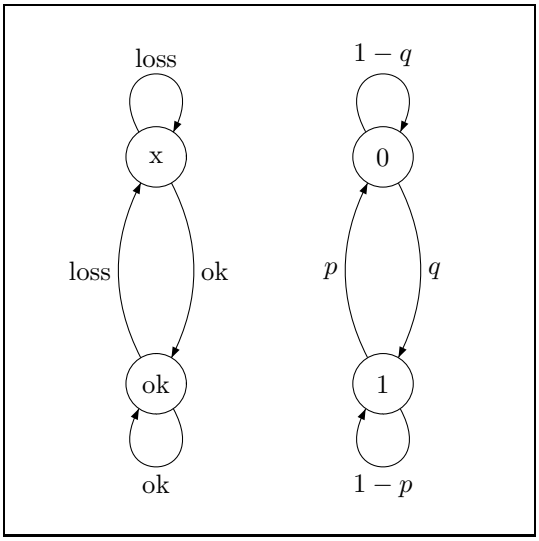

Figure 8: The Gilbert-like model to represent the loss process and the associated 2-states Markov chain. When in state "ok", a transition to state "x" corresponds to a loss, and to the same state "ok" corresponds to a correct transmission. From state "x", the loop corresponds to a loss and the transition to "ok" to a correct transmission.

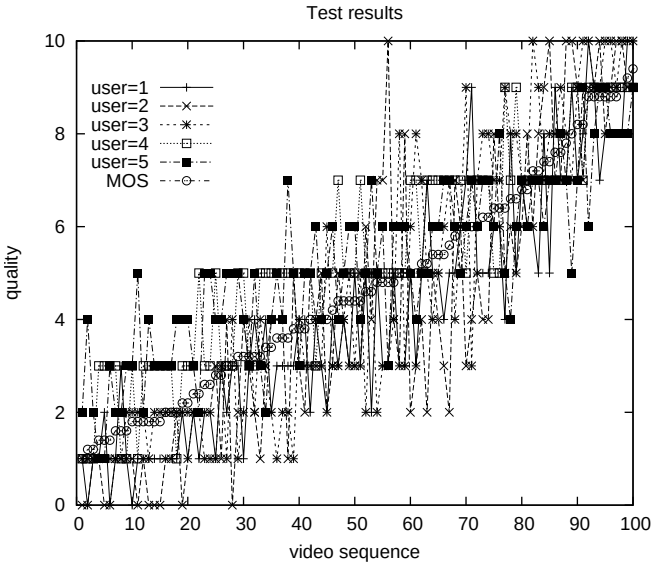

Figure 9: Subjective test results for building the PSQA metric.

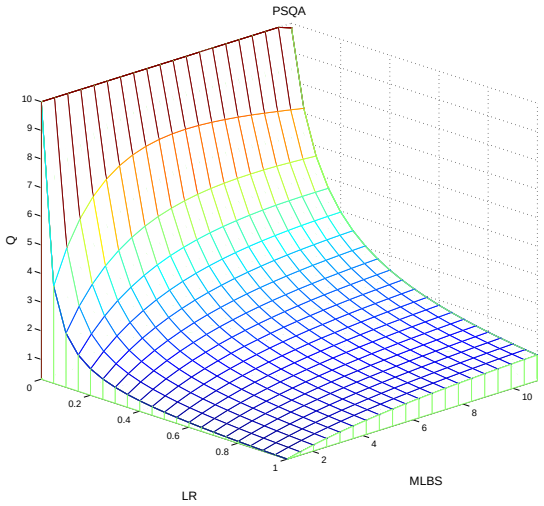

Figure 10: The PSQA curve in our setting

6.2 PSQA Evaluation in our Multi-Source Streaming Techniques

After obtaining a function mapping the two chosen parameters (frame loss rate and mean frame loss burst size) into perceived quality, it only remains to evaluate it on the values generated by the servers' disconnections in each streaming algorithm discussed in Section 5.

6.2.1 *Sending K copies of the stream*

Using the loss model for the "copy" method of Subsection 5.1, we evalute the PSQA measure in the different $LR^{copy}_{K,i}$ possibilities, the result is summarized in Table 4.

Table 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	0.0000	0.0000	0.0000	0.0000	0.0000
1	1.0000	1.0000	1.0000	1.0000	1.0000
2		1.0000	1.0000	1.0000	1.0000
3			1.0000	1.0000	1.0000
4				1.0000	1.0000
5					1.0000

i/K	6	7	8	9	10
0	0.0000	0.0000	0.0000	0.0000	0.0000
1	1.0000	1.0000	1.0000	1.0000	1.0000
2	1.0000	1.0000	1.0000	1.0000	1.0000
3	1.0000	1.0000	1.0000	1.0000	1.0000
4	1.0000	1.0000	1.0000	1.0000	1.0000
5	1.0000	1.0000	1.0000	1.0000	1.0000
6	1.0000	1.0000	1.0000	1.0000	1.0000
7		1.0000	1.0000	1.0000	1.0000
8			1.0000	1.0000	1.0000
9				1.0000	1.0000
10					1.0000

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

Using the loss model for the "split" method of Subsection 5.2, we evalute the PSQA measure in the different $LR_{K,i}^{split}$ possibilities, the result is summarized in Table 5 for the lower quality bound (based in lower bound $MLBS_{K,i}^{split} = 1$) and in Table 6 for the upper quality bound (based in upper bound $MLBS_{K,i}^{split} = K - i$).

Table 5: Simple Split method. Minimal Perceived Quality ($MLBS = 1$) as a function of K (initial number of servers) and i (number of surviving servers).

i/K	1	2	3	4	5
0	0.0000	0.0000	0.0000	0.0000	0.0000
1	1.0000	0.0051	0.0045	0.0045	0.0045
2		1.0000	0.0075	0.0051	0.0045
3			1.0000	0.0099	0.0063
4				1.0000	0.0123
5					1.0000
i/K	6	7	8	9	10
0	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0045	0.0045	0.0045	0.0045	0.0045
2	0.0045	0.0045	0.0045	0.0045	0.0045
3	0.0051	0.0045	0.0045	0.0045	0.0045
4	0.0075	0.0059	0.0051	0.0046	0.0045
5	0.0146	0.0087	0.0067	0.0057	0.0051
6	1.0000	0.0168	0.0099	0.0075	0.0063
7		1.0000	0.0190	0.0111	0.0083
8			1.0000	0.0212	0.0123
9				1.0000	0.0233
10					1.0000

Table 6: Simple Split method. Maximal Perceived Quality ($MLBS = K - i$) as a function of K (initial number of servers) and i (surviving servers).

i/K	1	2	3	4	5
0	0.0000	0.0000	0.0000	0.0000	0.0000
1	1.0000	0.0051	0.0075	0.0097	0.0113
2		1.0000	0.0075	0.0085	0.0097
3			1.0000	0.0099	0.0105
4				1.0000	0.0123
5					1.0000
i/K	6	7	8	9	10
0	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0126	0.0136	0.0145	0.0152	0.0158
2	0.0113	0.0126	0.0136	0.0145	0.0152
3	0.0109	0.0113	0.0126	0.0136	0.0145
4	0.0126	0.0127	0.0127	0.0128	0.0136
5	0.0146	0.0146	0.0144	0.0143	0.0142
6	1.0000	0.0168	0.0165	0.0162	0.0158
7		1.0000	0.0190	0.0185	0.0179
8			1.0000	0.0212	0.0205
9				1.0000	0.0233
10					1.0000

6.2.3 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 5.3, we evalute the PSQA measure in the different $LR_{K,i}^{split-red}$ possibilities, the result is summarized in Table 7 for the lower quality bound (based in lower bound $MLBS_{K,i}^{split-red} = 1$) and in Table 8 for the upper quality bound (based in upper bound $MLBS_{K,i}^{split-red} = K - i$).

Table 7: Redundanct Split. Minimal Perceived Quality ($MLBS = 1$) as a function of K (initial number of servers) and i (surviving servers).

i/K	1	2	3	4	5
0	0.0000	0.0000	0.0000	0.0000	0.0000
1	1.0000	1.0000	0.0075	0.0051	0.0045
2		1.0000	1.0000	0.0146	0.0083
3			1.0000	1.0000	0.0233
4				1.0000	1.0000
5					1.0000
i/K	6	7	8	9	10
0	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0045	0.0045	0.0045	0.0045	0.0045
2	0.0063	0.0053	0.0047	0.0045	0.0045
3	0.0123	0.0087	0.0070	0.0060	0.0054
4	0.0333	0.0168	0.0115	0.0090	0.0075
5	1.0000	0.0442	0.0219	0.0146	0.0111
6	1.0000	1.0000	0.0555	0.0274	0.0179
7		1.0000	1.0000	0.0669	0.0333
8			1.0000	1.0000	0.0781
9				1.0000	1.0000
10					1.0000

Table 8: Redundanct Split method. Maximal Perceived Quality ($MLBS = K - i$) as a function of K (initial number of servers) and i (surviving servers).

i/K	1	2	3	4	5
0	0.0000	0.0000	0.0000	0.0000	0.0000
1	1.0000	1.0000	0.0126	0.0109	0.0113
2		1.0000	1.0000	0.0243	0.0179
3			1.0000	1.0000	0.0388
4				1.0000	1.0000
5					1.0000
i/K	6	7	8	9	10
0	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0126	0.0136	0.0145	0.0152	0.0158
2	0.0158	0.0149	0.0143	0.0145	0.0152
3	0.0263	0.0219	0.0197	0.0183	0.0174
4	0.0554	0.0360	0.0288	0.0250	0.0227
5	1.0000	0.0732	0.0468	0.0365	0.0309
6	1.0000	1.0000	0.0916	0.0585	0.0448
7		1.0000	1.0000	0.1101	0.0709
8			1.0000	1.0000	0.1283
9				1.0000	1.0000
10					1.0000

6.3 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 makes it possible to answer different interesting and relevant questions.

As a first example, we can observe that the worst quality level must occur just before a network re-configuration, that is, at time T if we consider that the last re-configuration happened at time 0. The mean quality (considering the whole client population) is, with our assumptions, given by

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

Table 9 and Figure 11 compare the average video quality for the three policies (the data was computed using the lower bound for the perceived quality, the difference with the upper bound is completely negligible).

It is possible to see that, in the case where there is no redundancy ("simple split"), the subjective quality degenerates rapidly with the growth of servers K. Also it is possible to compare the "copy" and "redundant split" cases, where for the frequency of disconnection of our real scenario it 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.

Table 9: Average Quality as a function of the number of servers K.

K/Method (bandwidth)	"Copy" (KB)	"Split" (B)	"Redundant split" ($2B$)
1	0.9890	0.9890	0.9890
2	0.9999	0.9781	0.9999
3	1.0000	0.9675	0.9996
4	1.0000	0.9570	0.9993
5	1.0000	0.9466	0.9989
6	1.0000	0.9364	0.9983
7	1.0000	0.9264	0.9977
8	1.0000	0.9166	0.9970
9	1.0000	0.9068	0.9963
10	1.0000	0.8973	0.9955

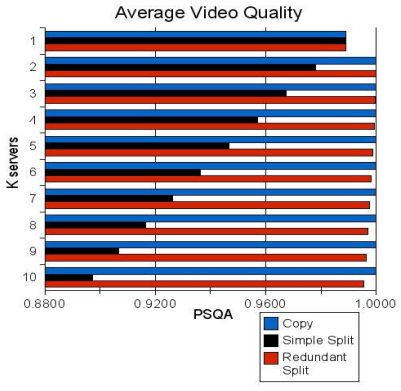

Figure 11: Average Video Quality

Another interesting question that it is possible to answer with our approach is how many servers are needed in each method to ensure that with a given probability (or confidence level), the quality of the transmission will be greater or equal than a pre-defined quality level.

It is possible to see for example that if we want to ensure a perfect quality transmission with a given probability of 0.999 (an alternative equivalent interpretation is that we want to ensure that at least 99.9% of the users will perceive perfect quality), we need to chose a method and a K such that:

$$\Pr(Q_K > Q_{min}) = \sum_{i=1/Q_{K,i} > Q_{min})}^{N} p_{K,i}(T) = 0.999.$$

The result is shown in the Table 10. It is possible to see for example that if we want to ensure a perfect quality with an availability of 0.999 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. From this table, we can see that it is impossible to attain this goal with the "simple split" method.

Table 10: Probability of perfect quality as a function of the number of servers K.

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

7. CONCLUSION

This paper proposes some models that may be useful 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 come from different servers and travel independently through the network.

The main focus is on how to ensure a high QoS for the users, by decomposing the original signal into different flows and eventually adding some redundancy. The QoS perceived by the end users is captured using the PSQA technique, which consists in training a neural network with scores given by a panel of users for a sample of transmission conditions, and afterwards employing this neural network to automatically compute an estimation of the perceived quality.

The paper focuses on three policies; 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. The third technique has much less transmission overhead than the first one, as the total bandwidth consumed is twice the original one (while in the first case, we use K times as much bandwidth).

To evaluate the different options, we develop analytical models for computing the Loss Rate and Mean Loss Burst Size 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.

These models were used to experimentally evaluate some configurations computed from realistic parameters (from information collected by a medium-sized ISP provider about the behavior of the users of its streaming video service).

In this particular scenarios, we evaluated the resulting perceived quality associated with the architectures mentioned before. Among the main conclusions, we can see that (i) in the simple split policy (with no redundancy), when the number of servers grows, the subjective quality degrades very quickly; (ii) 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; (iii) the copy policy has always the best perceived quality levels, but at the expense of much transmission overhead; (iv) using the information computed by the models, it is possible to take into account the different tradeoffs and for example to define the optimal number of servers to be used to ensure with a certain confidence a given QoS level.

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 Qos 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 support the designers' decision making, it can be useful to develop more realistic models, perhaps including other aspects as costs or bandwidth limitations at the networks components.

8. REFERENCES

[1] 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.

[2] Bittorrent home page. http://www.bittorrent.org, 2007.

[3] C. Cranor, M. Green, C. Kalmanek, D. Shur, S. Sibal, and J. V. der Merwe. Enhanced streaming services in a content distribution network. *IEEE Internet Computing*, (2):66–75, 2001.

[4] eMule home page. http://www.emule-project.net, 2007.

[5] E. Gelenbe. Random Neural Networks with Negative and Positive Signals and Product Form Solution. *Neural Computation*, 1(4):502–511, 1989.

[6] E. Gilbert. Capacity of a burst–loss channel. *Bell Systems Technical Journal*, 5(39), Sept. 1960.

[7] ITU-R Recommendation BT.500-11. Methodology for the subjective assessment of the quality of television pictures, June 2002.

[8] K. L. Johnson, J. F. Carr, M. S. Day, and K. M. F. The measured performance of content distribution networks. In *Proceedings of the 5th International Web Caching and Content Delivery Workshop*, 2000.

[9] jumpTV Home page. http://www.jumptv.com/, 2007.

[10] KaZaA home page. http://www.kazaa.com, 2007.

[11] I. Lazar and W. Terrill. Exploring content delivering networking. Technical report, IT Professional, aug 2001.

[12] 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, Jan. 2001.

[13] 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, Dec. 2002.

[14] 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.

[15] 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.

[16] 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.

[17] msnTV website. http://www.msntv.com, 2007.

[18] P. Rodrguez-Bocca, H. Cancela, and G. Rubino. Perceptual quality in P2P video streaming policies. In *IEEE Globecom 2007 Performance Modeling, QoS and Reliability Symposium*, Washington DC, USA, 26-30 November 2007.

[19] S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. van Steen. Replication for web hosting systems. *ACM Computing Surveys*, 36(3):291–334, 2004.

[20] C. van den Branden Lambrecht. Color Moving Picture Quality Metric. In *Proceedings of the IEEE International Conference on Image Processing*, Sept. 1996.

[21] C. van den Branden Lambrecht. *Perceptual Models and Architectures for Video Coding Applications*. PhD thesis, EPFL, Lausanne, Swiss, 1996.

[22] S. Voran. The Development of Objective Video Quality Measures that Emulate Human Perception. In *IEEE GLOBECOM*, pages 1776–1781, 1991.

[23] 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.

[24] youTube website. http://www.youtube.com/, 2007.