QoE Monitoring Platform for Video Delivery Networks

Daniel De Vera[2], Pablo Rodríguez-Bocca[1,2], and Gerardo Rubino[2]

[1] Instituto de Computación - Facultad de Ingeniería
Universidad de la República,
Julio Herrera y Reissig 565, 11300,
Montevideo, Uruguay
prbocca@fing.edu.uy
[2] Inria/Irisa
Campus universitaire de Beaulieu
35042 Rennes, France

Abstract. This paper presents a full video delivery network monitoring suite. Our monitoring tool offers a new view of a video delivery network, 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 recently proposed PSQA technology. Moreover, we improve PSQA's efficiency and robustness for video analysis by studying the flows at the frame level, instead of the packet level previously considered in the literature. The developed monitoring suite is a completely free-software application, based on well-known technologies such as the Simple Network Management Protocol or the Round Robin Databases, which can be executed in various operating systems. In this paper we explain the tool implementation and we present some of the first experimental measurements performed with it.

1 Introduction

As a consequence of the opening of video content producers to new business models, the more bandwidth availability on the access network (on the Internet, cellular networks, private IP networks, etc.) and the explosion in the development of new hardware capable of reproducing and receiving video streams, the area of Video Delivery Networks is growing nowadays at increasing speed.

A common challenge of any VDN 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. QoE is basically a subjective measure of end-to-end performance at the services level, from the point of view of the users. As such, it is also an indicator of how well the system meets its targets [1].

To identify factors playing an important role on QoE, some specific quantitative aspects must be considered. 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. The service provider needs 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 [2,3], by large telecommunication infrastructure providers as part of an end-to-end VDN solution [4–6], 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 paper 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 based on [7]. The traditional way to monitor video quality [2,3] 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 in the output of the encoding process and in a simulated client situated in the head-end of the network, where the experts are). 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 paper we instead address the problem of directly measuring *perceived* quality in by means of the Pseudo Subjective Quality Assessment (PSQA) technology [8,9]. PSQA is a general procedure that allows the automatic measure of the perceived quality, accurately and in real-time. Moreover, we extend the technique 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.

The paper is organized as follows. Section 2 introduces the measurement methodology used in our platform. In Section 3, the architecture of the platform is described. In Section 4, we report on some experimental results allowing to illustrate the platform use. The main contributions of this work are then summarized in Section 5.

2 Methodological Considerations

Before describing our methodology, recall that in the most important specifications for video, MPEG-2 and MPEG-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). An I frame codes an image, a P frame codes an image based on a previously coded I or P frame (by coding the differences, based on motion compensation) and a B frame is also a predicted

one based on past as well as future P or I frames. The frame sequence between two consecutive I frames is a group of pictures (GOP) in MPEG-2 and a group 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.

Quality Measurements. Let us consider here the different ways of evaluating the perceived quality in a video delivering system. 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 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). 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 exist standard methods for conducting subjective video quality evaluations, such as the ITU-R BT.500-11 [10]. 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). 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 discret 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.

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. The most commonly used objective measures for video are: Peek signal to noise ratio (PSNR), ITS' Video Quality Metric (VQM) [11], EPFL's Moving Picture Quality Metric (MPQM), Color Moving Picture Quality Metric (CMPQM) [12], and Normalization Video Fidelity Metric (NVFM) [12]. With a few exceptions, objective 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 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.

Pseudo Subjective Quality Assessment (PSQA). In [8] a hybrid approach between subjective and objective evaluation has been proposed. It 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 come from the Random Neural Networks one [13]) in order to capture the relation between the parameters that cause the distortion and the perceived quality.

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. 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 must 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 $\nu()$ from the values of the set of parameters into perceived quality. 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 $\nu()$ to obtain the *instantaneous* perceived quality at t.

In our work, we focused on two specific parameters concerning losses, because we know from previous work on PSQA that the loss process is the most important global factor for quality. 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. 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. While packet-level parameters are easier to handle (in the testbed and from the measuring point of view in the network), frame-level ones provide a more accurate view of the perceived quality.

An Extensible Measurement Framework. We distinguish two main components within our measuring system: a set of video players and a family of centralized monitoring servers. The video players are an improved version of the VideoLan client software (VLC) [14]. Each VLC client performs the measuring tasks, and makes the measures available to the servers using the Simple Network Management Protocol (SNMP) [7]. A collector server polls the clients to obtain the SNMP MIB values of the measured parameters. Concerning SNMP, our application uses the SNMPv3 [7,15] version and our designed MIB module is compliant with the SMIv2 [16–18].

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 own stream. As mentioned before, the user data defined in MPEG-2 and MPEG-4 allows to insert application's information inside the stream. 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, 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.

In Table 1 we show 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 VCL 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.

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 1. Measurement framework parameters.

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).

3 The Audit Platform

Architecture. Inside the VDN we want to monitor, there are five basic components (Fig. 1): the streaming server, the probes, the data collector server, the PSQA Learning 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's reports. The PSQA Module is where the perceptual quality value is computed. Finally `Webstat` provides a web interface for the probes's reports presentation.

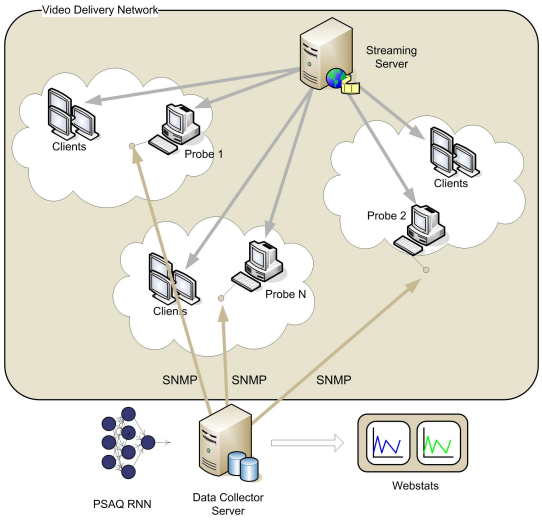


Fig. 1. Global architecture - The streaming server, the probes, the data collector server, the PSQA Learning Tool and the Webstat.

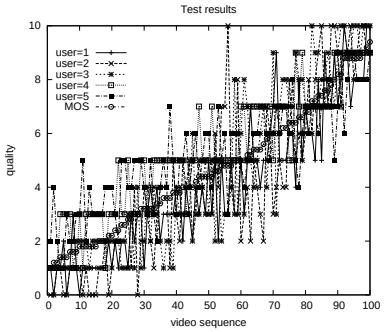
Streaming Server. The function of the streaming server is to provide multimedia content to the VDN. This content can be coded using different video specifications (MPEG-2, MPEG-4...), audio specifications (MP3, FLAC...), container formats (MPEG-TS, MPEG-PS, OGM...) and it can be streamed over different 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.

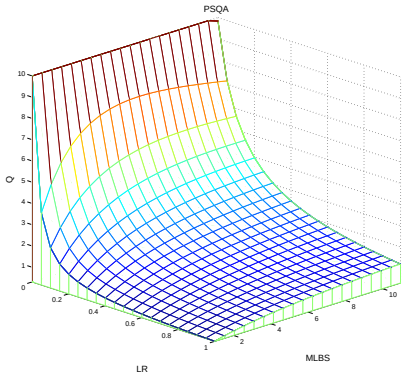
Probes (VLC players). Probes are VLC players modified in order to measure some specific information. They are strategically located inside the VDN. Basically, a probe is a VLC player with supplementary modules for coordination and data calculation, a SNMP module and a Logs module. They allow to capture and parse the user-data, to measure and to calculate generic information (like the start and stop of the stream), to offer realtime reports through SNMP and to manage a set of rotating logs 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.

PSQA Tool. In this subsection we study how the frame loss process affects the perceptual quality (as measured by the PSQA technique). The first step was to apply the PSQA technique, as explained in Section 2. For this, we chose four MPEG-2 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 [8] for more details). For each of the evaluation points, we used a simple Markov chain (a simplified Gilbert model [8, 9]) 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. Then, 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 norm [10] (see Figure 2(a)). 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 2(b) we can see the obtained 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%, 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



(a) Subjective test results for building the PSQA metric.



(b) The PSQA curve in our setting.

Fig. 2. The subjective test input of our PSQA technique, and the PSQA function (mapping *LR* and *MLBS* into perceived quality) after trained and validated.

MLBS, meaning that, in this losses range, humans prefer sequences where losses are concentrated over those where losses are spread through the flow.

Observe that training is only made once, when building the tool. In operation, the use of PSQA is very simple: the probes sends statistical information about the frame loss rate *LR* and the mean loss burst size *MLBS* in a short period to the data collector, to be evaluated by the PSQA Module. 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.

4 Evaluation and First Results

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). The first scenario consisted of a streaming server who sends traffic 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. In Table 2 we present 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,

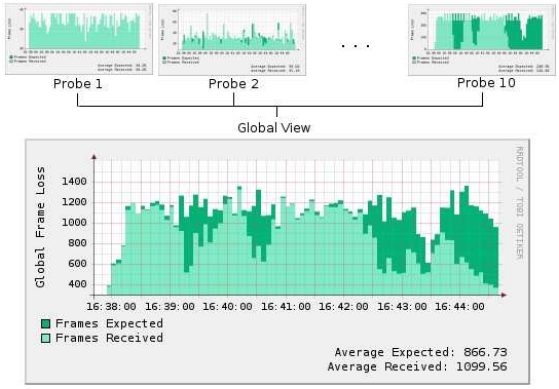
Test Case	Protocol	Video Specification	Video Bitrate (Kbps)	Scenario	Loss Rate
1	UDP	MPEG-2	1024	NoLS	-
2	HTTP	MPEG-2	1024	NoLS	-
3	HTTP	MPEG-4	512	NoLS	-
4	HTTP	MPEG-4	512	SLS	0.50
5	HTTP	MPEG-4	1024	SLS	0.30
6	UDP	MPEG-2	512	NLS	0.30
7	UDP	MPEG-2	1024	NLS	0.30
8	UDP	MPEG-4	1024	NLS	0.30
9	HTTP	MPEG-2	1024	NLS	0.02
10	HTTP	MPEG-4	1024	NLS	0.04

Table 2. 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).

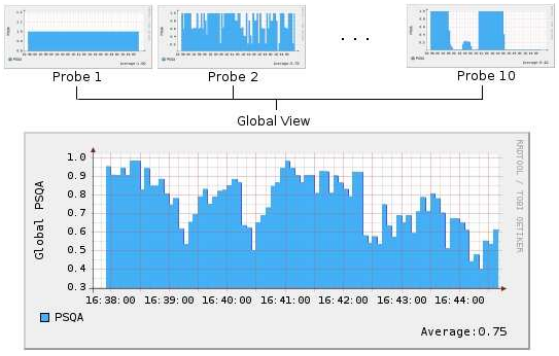
a maximum bandwidth is set in the network. This is in order to make the IP

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 3 and as shown in Fig. 3, the developed application lets us to analyze the results at different levels and providing different views. In Fig. 3(a) we show 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).



(a) Global frame losses.



(b) Global perceptual quality (QoE).

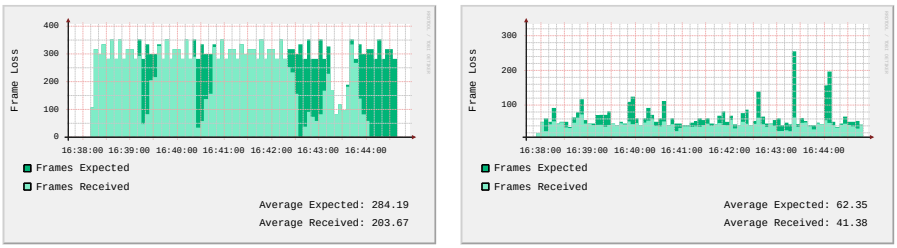
Fig. 3. Typical global view of the entire Video Delivery Network.

In Fig. 3(b) we show the evolution of quality (PSQA) with time, normalized to numbers in the interval $[0, 1]$. In Table 3 we present some values obtained when executing the simulations. As it can be observed, there is no visible relationship

Test Case	Protocol	Specified Loss Rate	Measured Frame Loss Rate	Mean PSQA
1	UDP	-	-	1.00
2	HTTP	-	-	1.00
3	HTTP	-	-	1.00
4	HTTP	0.50 (Frame level)	0.47	0.42
5	HTTP	0.30 (Frame level)	0.29	0.61
6	UDP	0.30 (IP level)	0.09	0.79
7	UDP	0.30 (IP level)	0.19	0.66
8	UDP	0.30 (IP level)	0.33	0.38
9	HTTP	0.02 (IP level)	0.08	0.79
10	HTTP	0.04 (IP level)	0.07	0.89

Table 3. Obtained Results.

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, Fig. 4(a) shows the frame losses measured during a test case where server failures occur and Fig. 4(b) shows the frame losses measured during a test case with network losses.



(a) Server failure example (5th test case). (b) Network loss example (8th test case).

Fig. 4. Typical stream view of the Video Delivery Network.

5 Conclusion

This paper 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 our tool 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 multiples 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.

References

1. DSL Forum Technical Work WT-126: Video services quality of experience (qoe) requirements and mechansims (2007)
2. Evertz Microsystems Ltd. - Manufacturer of High Quality Film and Digital Broadcast Equipment. http://www.evertz.com/ (2007)
3. Tandberg Home page. http://www.tandberg.com/ (2007)
4. Cisco Systems, Inc. http://www.cisco.com/ (2007)
5. Alcatel-Lucent Home. http://www.alcatel-lucent.com/ (2007)
6. Siemens AG. http://www.siemens.com/ (2007)
7. IETF Network Working Group: An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks (RFC 3411) (2002)
8. Mohamed, S., Rubino, G.: A Study of Real–time Packet Video Quality Using Random Neural Networks. IEEE Transactions On Circuits and Systems for Video Technology **12**(12) (2002) 1071 –1083
9. Mohamed, S., Rubino, G., Varela, M.: Performance evaluation of real-time speech through a packet network: a Random Neural Networks-based approach. Performance Evaluation **57**(2) (2004) 141–162
10. ITU-R Recommendation BT.500-11: Methodology for the subjective assessment of the quality of television pictures (2002)
11. Voran, S.: The Development of Objective Video Quality Measures that Emulate Human Perception. In: IEEE GLOBECOM. (1991) 1776–1781
12. van den Branden Lambrecht, C.: Perceptual Models and Architectures for Video Coding Applications. PhD thesis, EPFL, Lausanne, Swiss (1996)
13. Gelenbe, E.: Random Neural Networks with Negative and Positive Signals and Product Form Solution. Neural Computation **1**(4) (1989) 502–511
14. VideoLan home page. http://www.videolan.org (2007)
15. IETF Network Working Group: Management Information Base (MIB) for the Simple Network Management Protocol (SNMP) (RFC 3418) (2002)
16. IETF Network Working Group: Structure of Management Information Version 2 (RFC 2578) (1999)
17. IETF Network Working Group: Textual Conventions for SMIv2 (RFC 2579) (1999)
18. IETF Network Working Group: Conformance Statements for SMIv2 (RFC 2580) (1999)