Journal of Mobile Multimedia, Vol. 7, No.1 (2011) 066-088
© Rinton Press

A MORE REALISTIC RTP/RTCP-BASED SIMULATION PLATFORM FOR VIDEO STREAMING QoS EVALUATION

FERNANDO BORONAT, MARIO MONTAGUD, VICENT VIDAL

IGIC Institute, Universitat Polièécnica de València

fboronat@dcom.upv.es; mamontor@posgrado.upv.es; vvidal@dsic.upv.es

Received March 6, 2011

Over the last few years, the demand for real-time multimedia services has been growing progressively so that video streaming applications are expected to be dominant in future communications systems, and most of them using RTP/RTCP protocols. This paper presents an evolved tool-set for video streaming QoS evaluation in simulated environments using such protocols. We have designed a new NS-2 module with a full RTP/RTCP implementation (following strictly the RFC 3550) and we propose to combine it with additional multimedia tools to obtain an advanced simulation framework that allows the measurement of network-level QoS metrics (such as throughput, delay, jitter or loss rate) in simulation time. Besides, as the transmitted video files can be reconstructed and played out at the receiver side, the measurement of the quality of the delivered video streams, by employing the most common objective quality metrics (such as PSNR, SSIM or VQM) or subjective metrics (MOS), is also supported. By using this tool-set, researchers and practitioners can assess their novel designs (such as network protocols, routing strategies or video coding mechanisms) for such applications in heterogeneous scenarios over different network conditions. As RTCP feedback capabilities have been added, source based control techniques (such as rate adaptability or Multiple Description Coding) could be included and tested using this more realistic and powerful simulation platform.

Key words: Multimedia Systems, NS-2, RTP/RTCP, Simulation, Video Quality Measurement

1 Introduction

Recently there has been a growing interest in the development and deployment of real-time multimedia streaming services, such as VoD (Video on Demand) or IPTV (Internet Protocol Television). Although video delivery over packet-switched networks faces many challenges, minimum Quality of Service (QoS) levels (such as accurate synchronization between media streams or end-to-end delay, jitter and packet loss minimization) must be maintained in order to ensure the customer's satisfaction. QoS of video communications is affected by some parameters at both network and application levels. On the one hand, some factors such as delay, jitter, burstiness, packet loss, etc., are introduced at the network level. On the other hand, at the application level, QoS is driven by factors such as frame rate, media codec, quantiser scale, loss recovery techniques, packetisation scheme, etc. Consequently, many researchers have developed new strategies (video coding mechanisms, routing algorithms, buffering policies, network or transport protocols, etc.) in order to enhance the performance of the multimedia data delivery. Test performances of these novel designs can be carried out using different approaches. One widespread alternative is network simulation. With the aid of simulation techniques, the functionality of these novel proposals, which might often be particularly difficult and time/cost expensive to implement in real platforms, can be evaluated effortlessly in heterogeneous network architectures. Simulation allows the networking researchers to learn about a real system by experimenting with a virtual model that characterizes it. Moreover, they can modify, during the simulation process, various attributes of the virtual scenarios (such as network topologies, links' capacity, network load, video codec mechanisms, coding parameters, transmission protocols, etc.) in a controlled way to assess how their proposals would behave under diverse network conditions. Unfortunately, all the network-related or application-related factors involved in video delivery services are often not handled in simulation studies. On the one hand, these factors could not be implemented or supported in the existing simulation tools. On the other hand, not all network researchers are multi-faceted in cross-layer designs (e.g. some might be specialized in multimedia coding but not familiar with underlying networks) and they could not take into account some of these factors, although they could significantly influence the video delivery performance. These reasons frequently result in simplified simulation models that may not match with real experiments. Hence, the development of a realistic, powerful and practical simulation tool-kit (e.g., integrated with existing network simulation tools, such as NS-2 [1], OPNET [2] or OMNeT++ [3]) for the evaluation of the QoS of video streaming would be a valuable contribution for the research community.

1.1. Motivation

Our research group is interested in the use of RTP (Real Time Protocol) and RTCP (Real Time Control Protocol) standard protocols, specified in [4], for streaming services (video, audio or data), their extension with synchronization purposes [5], and their evaluation through simulation techniques. These protocols have become a suitable framework to cope with the temporal and QoS requirements of real-time multimedia applications. On the one hand, the timestamps and sequence numbers mechanisms provided by RTP in each data packet are very useful to reconstruct the original media timing, reorder and detect some possible packet losses at the receiver side. On the other hand, the reporting features (transport and management of feedback control messages between all the participants in an RTP session) provided by RTCP make QoS monitoring (such as round trip time – RTT –, jitter or loss rate) possible and, subsequently, they can be used by service providers in multicast distribution services, such as IPTV, for troubleshooting or fault tolerance management [6]. Nowadays, NS-2 has become a widely adopted simulation tool for networking community as a way of designing and evaluating existing and new proposed protocols, algorithms or designs. Since its inception, NS-2 has been under constant improvement and at present it supports heterogeneous network architectures characterization, such as Mobile IP networks, WLAN, ad-hoc networks, sensor networks, grid architectures, satellite networks, and many others. Additionally, it contains modules for numerous network components such as MAC layer protocols, unicast and multicast routing algorithms, transport layer protocols, traffic source behaviour, queue management mechanisms, modules for statistics measurement, etc. Despite this variety, sometimes networking researchers need to adapt the existing NS-2 modules to their requirements or incorporate new simulation modules which are beyond the scope of the built-in NS-2 code. The simulator is open source; hence, it allows everyone to make changes to the existing code besides adding new protocols and functionalities to it. Therefore, the above reasons made us choose NS-2 as the simulation tool for developing our research goals. As simulations rely on the accuracy of the proposed models, and the NS-2 native implementation for

RTP/RTCP neither defines nor strictly follows many of the attributes specified in [4], we decided to develop a new NS-2 module with a more precise and complete implementation for such protocols [7], by including all the properties and functionalities specified in [4] that originally were not considered or implemented accurately in the legacy code (discussed in Section 3). However, this new RTP/RTCP module could only provide us monitoring of network-level QoS measurements that may be insufficient when application layer video quality is required. Thus, we needed to combine this simulation module with several multimedia tools (such as video encoders/decoders, video players, trace files generators or video quality assessments programs) to obtain a useful tool-set that allows the researchers to assess and analyse their own proposed designs not only by means of classical network-level QoS metrics, but also through the reconstruction of the real transmitted video sequences, which in turn can be played out and compared with the original one, [8]. Then, common (application-layer) video quality metrics, such as the ones detailed in [9], which are those that truthfully reveal the performance of video streaming applications, can be estimated for each received video sequence.

1.2. Related Works

Simulation techniques have been broadly used for communication network performance analysis. Previous video transmission studies, such as the ones in [10], [11] and [12], adopted mathematical traffic generation models, such as CBR or VBR traffic flows characterization, or H.263/H.264/MPEG4 video trace files rather than the real video profile of the specific video sequences under study in their simulation environment. Video traces only provide network-level analysis such as throughput, delay, jitter or loss rate. These metrics can reflect network states but may be insufficient to adequately rate the end user perception quality because it is difficult to correlate network-level metrics into application-layer metrics of a video transmission. As an example, the same value of packet loss rate may have a different impact on the user perception of the received video quality because it depends on the percentage of each type of video frame being lost. Moreover, these kinds of studies do not permit the adjustment of the video coding parameters nor apply alternative coding mechanisms to the same video source. Different multimedia encodings can result in different perceived video quality, although the transmission is carried out with exactly the same set of protocols and the same network conditions. Therefore, the study of any proposed solution by using real video files is essential for associating the simulation results with the user perception on the video quality.

To overcome the above described limitations, a framework for the evaluation of the quality of video transmitted over a real or simulated communication network, which was called Evalvid, was presented in [13]. It supported packet/frame jitter, loss rate, Peak Signal to Noise Ratio (*PSNR*) measurement and a static *PSNR* to Mean Opinion Score (*MOS*) mapping. Evalvid framework can be integrated into several network simulators, such as NS-2 (as in [14], [15], [16], [17], [18], [19], [20] and [21]) or OPNET (as in [22] and [23]). In [14] the original simulated environment in [13], which simply consisted of an error model to symbolize corrupted or missing packets in the real network, was modified by integrating Evalvid into NS-2 through new designed UDP-based agents. Thus, the resulting tool-set from this integration allowed the network researchers to assess their designs in the presence of real video traffic through heterogeneous network scenarios. In [15] some enhanced video transmission interfaces were included, in order to improve the simulation model, and a multiple description coding (MDC) mechanism was introduced and evaluated over a wireless scenario. In [16] an Ad-Hoc Evalvid framework was presented to measure video performance metrics over multi-hop wireless networks by using various routing techniques. Since the previous tool-sets could only evaluate the quality of video, a new tool-kit was developed to evaluate the quality of audio as well [17]. In [18] a solution for rate adaptive MPEG-4 video evaluation (Evalvid-RA) was presented. In [19] the previous work was extended to provide a solution for rate adaptive video transmission in the multicast domain. In [20] a novel NS-2 based simulation mechanism in which the video coder/encoder components, the application layer QoS control module and the simulated network have been fully integrated and work in parallel is presented. In [21] a Shaped VBR (SVBR) mechanism was integrated in the previous rate adaptive framework in order to regulate unpredictable large bursty video traffic by utilizing a leaky bucket algorithm. In [22] Evalvid was integrated into OPNET, and the resultant framework was used to evaluate the performance of several routing techniques in video streaming applications over multi-hop wireless ad hoc networks. In [23] the previous framework was used to measure the performance of a video application over a high data rate multi hop wireless network by using a Directed Diffusion based routing protocol.

Unlike the previous platforms in which network-level performance metrics are evaluated at the end of the simulation process, we have designed our tool-set taking advantage of the RTP/RTCP capabilities, so the RTP timestamps and sequence numbers provide the measurements of QoS network-level metrics during the simulation process, and the RTCP feedback capabilities enable the assessment of novel source or receiver-based proposals. As an example, sources could use the very useful information included in the feedback reports to make possible adaptations required by the multimedia system during the simulation (for example, in order to decrease its transmission rate in congestion cases). This work is an extended version of [8], which was selected for publication in the Special Issue of this Journal. Moreover, it is a result of our effort to improve the functionality, flexibility and performance of the designed simulation tool-set. One evolution regarding the previous tool-set is that the actual version is capable of adopting both raw video YUV video files, such as the ones available in [24], and downloadable video traffic trace files, such as the ones available in [25], as source video streams in each simulation study. Moreover, we have included in the latest version of our simulation platform all the video codec mechanisms that are supported by [13] and [18], such as MPEG4, H.263 or H.264, for a given raw video sequence. The selection of the video codec mechanisms allows a possible comparison between their efficiency and suitability for specific video applications under study. Besides, various coding parameters, such as frame rate, bit rate, quantiser scale or GoP (Group of Picture) size, can be handled for the selected video encoder. The tuning of these parameters can be very useful for adjusting the bit rate according to the available resources (bandwidth, memory, CPU, etc.) or the resolution (video quality) required for specific applications. Another advantage of the new version is that researchers can configure the value of the Maximum Segment Size (MSS) for the RTP data packets in their simulation scripts (in the previous tool-set this size was set by the adopted RTP packetiser tools [26] and it was not configurable). Thus, it includes more flexibility in the configurations of the simulation studies because it specifies the number of RTP packets that are necessary to carry out the encoded video frames. As in the previous work, we also provide the measurement of video quality metrics for the received video sequence. Additionally, in this paper we also include a brief description of the most common and their correlation with the user perception on the reconstructed (possible distorted) video streams in order to facilitate to the users the comprehension and the accuracy of the obtainable results (Section 2). Moreover, as the (objective and subjective) video quality metrics for the received video sequences take a long time to compute, since they have to compare the pixels of each frame of the video sequences, and they have to be performed at the end of the evaluation process, we have also included a decodable frame rate (Q) module at the receiver side, which can predict the possible degradation of the video quality since it is able to detect the frame type (I, P or B frames) and simulation time for each video frame that has not been completely received (or has been received later than its scheduled playout time). This way, for example, it can be corroborated that a significant reduction in the *PSNR* values for a collection of video frames of a specific received video sequence is due to the loss of the intra (I) frame of the GoP that corresponded to these video frames.

In short, this advanced RTP/RTCP based simulation platform includes all the necessary tools to perform simulation studies and assess the QoS indicators that characterize the perceived video quality at the end user and provides flexibility on the selection of specific video codec mechanism and their parameters. This way, researchers can adjust their simulation studies, according to their requirements and assess the benefits and limitations of their proposed designs for specific video delivery services because they can achieve close results to those obtained through real video experimental evaluation processes.

1.3. Organization of the Paper

The remainder of this paper is organized as follows. In Section 2, a brief discussion on some common video quality metrics that can be measured using the presented tool-set is provided. The new developed RTP/RTCP module for NS-2 and the enhancements that have been incorporated are described in Section 3. The structure of the proposed tool-set and the necessary steps to make a new video streaming simulation are depicted in Section 4. In Section 5, the tool-set functionality and capabilities are shown through exemplary simulations. Finally, we present our conclusions, summarize our contributions and suggest some ideas for future work in Section 6.

2 Video Quality Assessment

Though classical network-level metrics, such as throughput, delay, jitter or packet loss rate, can certainly be a sign of the video quality, the user's impression on the delivered video streams is the most important factor to take into account in the design of multimedia systems. Video quality assessment methods can be classified into two categories in respect of the involvement of human interaction during the evaluation process, namely subjective and objective methods. In subjective experiments, a number of "*subjects*" (typically 15-30) are asked to watch a set of video clips and rate their quality level. The test results are usually measured using a *MOS* 5-level scale, that ranges from 5 (excellent) to 1 (bad), as can be seen in Table 1, and the final score is obtained as the average rating of all the individual scores for a given clip. The standard deviation between the individual scores may also be useful to measure the consistency between subjects. Some recommendations about subjective evaluation have been formalized by ITU in [27] and [28]. Though subjective assessments represent the most accurate and reliable method for obtaining quality rankings, the measurement of *MOS* is an expensive and complex process because it requires a large number of viewers, time consumption, controlled evaluation environments and often require special equipment, such as fixed screen size for displaying a video. Thus, they are neither intended nor practical to provide automatic evaluation for in-service monitoring of video quality. To cope with these impairments, objective quality metrics have been designed to characterize the video quality and to emulate the quality impression of the Human Visual System (HVS) and, thus, predict viewer *MOS*. Objective quality metrics do not involve human interaction and they are purely based on mathematical methods and criteria that can be measured automatically by a computer program. Different types of objective metrics exist [9]. For the analysis of decoded video, we can distinguish data metrics, which measure the fidelity of the signal without considering its content, and picture metrics, which consider the visual information that is contained in the video and the relevancy of some features in HVS such as colour perception, contrast sensitivity, pattern masking or some structural features (e.g. contours). Furthermore, metrics can be classified as full-reference (FR), no-reference (NR) and reduced-reference (RF) based on the amount of reference information they require. FR metrics require the entire reference (original) video to be available to perform a frame-by-frame comparison between the reference and the (possibly distorted) test video. NR metrics analyse only the test video, without the need for a explicit reference clip. RR metrics extract a number of features from the reference and/or test video (such as amount of motion or spatial detail), and the comparison is based on those features. In this study, we focus on FR metrics and introduce and give results of some of the most common, such as Mean Squared Error (*MSE*), *PSNR*, Structural Similarity (*SSIM*) or Video Quality Metric (*VQM*).

The simplest FR metric is the *MSE* between the test and the reference sequence or its derivatives, such as *PSNR*, which mathematically is just a logarithmic representation of *MSE*. *PSNR* is a measure of distortion in transmitted video signal due to encoding (data compression), transmission and decoding processes. This metric compares the maximum possible signal energy to the noise energy, which has shown to give a higher correlation with the subjective quality perception than the conventional Signal to Noise Ratio (*SNR*). Although each pixel is presented by one luminance value (Y) and a set of pixels by two chrominance values (*Cr* and *Cb*), only the luminance component (*Y*) is taken into account in the calculation since it is the most relevant component for the HVS [18]. The methodology for computing the average *PSNR* for a sequence consists of calculating the average *MSE* between the luminance component (*Y*) of all the frames of a source image *S* and a received image *R*, as detailed in equation (1), and after that to derive *PSNR* by setting the *MSE* in relation to the maximum possible pixel value of the luminance (V_{peak}), as detailed in equation (2):

$$MSE(n) = \frac{\sum_{i=1}^{N_{col}} \sum_{j=1}^{N_{row}} [Y_S(n,i,j) - Y_R(n,i,j)]^2}{N_{col} N_{row}} , \qquad (1)$$

$$PSNR(n)_{dB} = 20 \log_{10} \left(\frac{V_{peak}}{\sqrt{MSE(n)}} \right) , \qquad (2)$$

where $V_{peak} = 2^k\text{-}1$ and k = *number of bits per pixel* (typical value k=8, V_{peak}=255), $Y_S(n,i,j)$ represents the original *n-th* sent signal at pixel (i, j), $Y_R(n,i,j)$ is the *n-th* reconstructed signal and $N_{col}N_{row}$ is the picture size (N_{col} and N_{row} are the number of pixels in horizontal –length– and vertical –height– direction, respectively; e.g in QCIF format N_{col}=176 and N_{row}=144). The above equations are simple to understand and to implement, so it contributes to the popularity of these metrics since they are easy and fast to compute. The individual *PSNR* values at the source or receiver are not very significant, but the difference between the quality of the encoded video at the source and the received one can be used as an objective QoS metric to assess objectively the transmission impact on video quality at the application level. Thus, *PSNR* is calculated by comparing the first frame of the streamed video (i.e. processed) with the first frame of the reference video, and then comparing the second frames of the streamed and the reference videos, and so on. This simple calculation method assumes no frames are lost in the streamed video. As *PSNR* is performed frame-by-frame, the total number of video frames in the reconstructed raw video sequence must be the same as the original video. Thus, in our tool-set, if some packets are lost, simple error concealment is employed. That is, the last successfully decoded frame is copied to the consecutive lost frames until a correct decoded frame is found.

Since the quality of the signal can have a very wide dynamic range, the *PSNR* values are normally given in a logarithmic scale. A lower value for *MSE* means less error, and as seen from the inverse relation between the *MSE* and *PSNR*, this translates to a high value of *PSNR* (lower distortion degree). The highest *PSNR* value refers to the case when there are no distortions in any frame of the video. Note that if there is no distortion, the *PSNR* value should be infinite according to the definition (which corresponds to *MSE*=0). But for the sake of calculation and analysis, we use the common approach to define the full score of *PSNR* to be 100 dB [29]. *PNSR* values over 40 dB are often considered undistinguishable frames from the original ones. Typical *PSNR* values range between 20 and 40 dB. Values in the range from 30 to 40 dB usually provide good video quality. Values below 30 dB are not necessarily satisfactory (depending on the specific application), while values below 20 dB normally represent garbled results.

Despite its popularity for the measurement of the (loss of) image quality due to its simplicity and mathematical convenience, the correlation between *PSNR* and subjective human judgement of video quality is not tight enough for most applications (only gives an approximate relationship), simply because it is based on a byte-by-byte comparison of the data without considering what they actually represent. This metric does not take into account the spatial-temporal property of HVS perception nor the content, i.e. the meaning, and thus the visual importance of the packets and bit concerned (the viewer perception depends on the part of the image or video where the distortion occurs). The same number of lost packets can have drastically different effects on the video content, depending on which parts of the bit stream are affected. Therefore, absolute borders between bad/good/very good/excellent may not be specified because the values vary by content. However, previous works, such as [30] and most of [13]-[23], have introduced a direct mapping between the *PSNR* of each single frame to the *MOS* scale, according to Table 1. Based on this frame by frame *MOS* calculation, it could be a possible metric to roughly estimate the user perception on the quality of received videos with relatively low motion (e.g. video from surveillance cameras). A *MOS* value is assigned to each image according to Table 1 which is based on the *PSNR* values that are calculated for every single image of a received video sequence. These values are averaged over all images to produce a single quality indicator for a video transmission as proposed by the methodology described in [27].

Recent research efforts, such as the ones in [29] and [31], have been devoted to handling some aspect such as the spatial and temporal distortion (temporal activity refers to the intensity of movement in the video sequence, whereas spatial activity refers to the variation and amount of details in the spatial domain) or packet loss in the traditional *PSNR* calculation (e.g. by “matching” the correct frame pairs in the streamed video and reference video sequence) to acquire higher (close-to-linear) correlation to subjective video quality (*MOS*), while retaining the low mathematical complexity for its computation.

Table 1 Quality and Impairment Scale and Possible PSNR to MOS Mapping

PSNR (dB)	**MOS**	**Perceived Quality**	**Impairment**
>37	5	Excellent	Imperceptible
31-37	4	Good	Perceptible, not annoying
25-30	3	Fair	Slightly annoying
20-24	2	Poor	Annoying
<20	1	Bad	Very Annoying

Due to the problem with simple data metrics such as PNSR, a lot of works have also been done to develop more advanced quality metrics that specifically account for the effects of distortions and content on perceived quality. *SSIM* index [32] is a FR metric that was designed to improve on the above traditional error-based methods like *PSNR* and *MSE*. *SSIM* is based on the measurement of the the structural distortion or similarity between two images. It computes the mean, variance and covariance of small patches inside a frame and combines the measurements into a distortion map. Motion estimation is used for weighting of the *SSIM* index of each frame in a video. The *SSIM* metric is calculated on various windows of an image. The structural similarity index between two windows of the original signal $x=\{x_i, i=1,2,\ldots,N\}$ and the distorted signal $y=\{y_i, i=1,2,\ldots,N\}$ of common size $N\times N$ can be measured as follows:

$$SSIM(x,y)=\frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu^2{}_x+\mu^2{}_y+c_1)(\sigma^2{}_x+\sigma^2{}_y+c_2)} \qquad (3)$$

In the above equation, μ_x is the average of x, μ_y is the average of y, $\sigma^2{}_x$ is the variance of x, $\sigma^2{}_y$ is the variance of y, σ_{xy} is the covariance of x and y, $c_1=(k_1L)^2$ and $c_2=(k_2L)^2$ are variables to stabilize the division with weak denominator, L is the dynamic range of the pixel-values (typically this is $2^k\text{-}1$ and k = number of bits per pixel), $k_1=0{,}01$ and $k_1=0{,}03$, by default. In order to evaluate the image quality this equation is applied only on luminance component. The resulting *SSIM* index is a decimal value less or equal to 1, and value 1 is only reachable in the case of two identical sets of data. Typically it is calculated on window sizes of 8×8. The window can be displaced pixel-by-pixel on the image but in [32] it is proposed to use only a subgroup of the possible windows to reduce the complexity of the calculation. Recent tests [33] have shown that *SSIM* outperforms *PSNR* metric in most cases since it is closer to the HVS perception.

VQM [34] divides the sequences into spatio-temporal blocks, and a number of features measuring the amount and orientation of activity in each of these blocks are computed from the spatial luminance gradient. This metric uses objective parameters to measure perceptual effects of video impairments such as blurring, jerky/unnatural motion, global noise, block distortion or colour distortion, and combines them into a single metric. The output values range from zero to one, which respectively denote no perceptual impairment and maximum perceived impairment. *VQM* achieves high correlation with subjective video quality assessment and has been adopted by ANSI as an objective video quality standard.

A comparison between the above objective video quality metrics is presented in Table 2. From this table, it can be seen that *SSIM* and *VQM* give more reliable results than *PSNR*. However, the computational complexity of these methods makes them difficult to apply in most scenarios. Nevertheless, *PSNR* metric is still the most common used metric due to their mathematical simplicity and the acceptable performance that it can offer in some applications in which relative quality metrics rather than absolute metrics can be regarded as adequate. Note that the aim of this work is not to provide a detailed description of the existing video quality measurements; however, we think that a brief introduction of them and of their performance may facilitate to the users of our simulation tool-set the comprehension of the obtainable results.

Table 2 Comparison among Video Quality Metrics

Metric	**Mathematical Complexity**	**Correlation with Human Perception**
PSNR	Simple	Poor
SSIM	Complex	Fairly Good
VQM	Very Complex	Good

3. RTP/RTCP in NS-2

Generally, specific protocols are implemented in NS-2 as Agents. These agents represent endpoints where packets are constructed or consumed, and can be used for the implementation of protocols at various layers. The necessary steps to design and implement new NS-2 Agents can be found in [35]. In the original code, RTP and RTCP protocols are implemented as the *RTP Agent* and the *RTCP Agent* classes, respectively. These two classes are implemented in the *rtp.cc* (located in *~ns/apps/rtp.cc*[a] path) and *rtcp.cc* (located in *~ns/tcp/rtcp.cc* path) files, as can be appreciated in Figure 1. Both classes derive from the *Agent* superclass, which C++ implementation is located in *~ns/common/agent.cc* file.

The *RTP Agent* holds all the functionality for sending and receiving data packets, whereas the *RTCP Agent* is responsible for transmission and reception of the control reports. The *RTP Session* class (implemented in *~ns/common/session-rtp.cc*) mainly deals with feedback report building and participant's information tables maintenance through the received packets passed from its agents. This class also defines the procedures for session initialization, report interval calculation, associating new RTP sessions with nodes, managing join and leave processes to multicast groups, stopping RTP flow transmissions, liberating the session resources, etc. It is called by its binding *Session/RTP* OTcl class (implemented in *~ns/tcl/rtp/session-rtp.tcl*). All the C++ files we have cited use *rtp.h* (located in *~ns/apps/rtp.h*) as header file, and they are shown in red boxes in Figure 1.

However, as mentioned in Section 1, the native RTP/RTCP code of NS-2 is quite generic. Many attributes specified in [4] are not included and many others are not implemented accurately: i) it does not define all the RTCP packets, only RTCP SR packets are included, but its format is not complete (does not include payload type field, number of packets/octets sent fields, etc.); ii) since RR messages are not defined, neither QoS metrics (jitter, network delay, RTT, loss rate) monitoring nor reporting are provided; iii) the same packet header data structure is used for both RTP and RTCP packets

[a] *~ns* refers to the directory *ns2.XX* in the simulator, where *XX* denotes the installed version of the simulator.

construction; iv) these packet header fields are specified using incorrect variables' types and sizes; v) there is a bug for multicast transmissions configuration; vi) the code does not support multiple multicast streams on the same node; vii) the RTP Agent is only capable of generating CBR traffic, etc.

Apart from the native implementation, we found two additional implementations of RTP/RTCP protocols including improvements to that native code and new modules for specific purposes ([37] and [38]). On the one hand, in [37] new RTP and RTCP Agents were defined with further functionalities in order to provide loss and jitter control in MPEG-2 traces streaming over wireless environments with QoS (802.11e). In addition, new independent data structures were defined to generate the RTP and RTCP packets. The data structures contained more differentiated fields than the native code but their sizes were not correct and some fields of these packets were not specified in [4]. On the other hand, in [38] new extensions to the legacy RTP/RTCP code in NS-2 were added in order to: i) provide the code with additional features related to QoS measures (loss and jitter control); and ii) employ TCP Friendly bandwidth share behaviour of multimedia data transmission through multicasting. The study of both implementations enabled us to develop a more complete and accurate RTP/RTCP module including the following enhancements: i) definition of all the types of RTCP packets with their exact format (Sender Report or SR, Receiver Reports or RR, Source Description or SDES, Application-defined or APP and BYE packets); ii) network-level metrics (such as end-to-end delay, jitter, RTT, throughput and packet loss) monitoring, processing and registering in simulation time; iii) capability of processing any kind of application traffic pattern supported by NS-2; iv) support for multiple multicast streams on the same node; and v) compatibility with the legacy code. The implementation and functionalities of the new designed code are described in the next sub-section.

3.1. Enhanced RTP/RTCP Module

In contrast to the other two discussed implementations, our new RTP/RTCP module can be included together with the other built-in NS-2 modules without needing to replace the RTP legacy code. As we consider the native NS-2 directory structure a bit confusing, we have collocated our source files in a more coherent way as can be observed in Figure 1 (dotted line blue boxes): the C++ code is located in *~ns/rtp_gs* directory and the OTcl code is located in *~ns/tcl/rtp_gs* directory (*gs* stands for the acronym of authors' research group). We have also included a new header file for each C++ file (Figure 1). In addition, as we did not replace the original RTP/RTCP protocols, we also had to define new packet types for the novel designed RTP/RTCP protocols in order to distinguish them from the native packets in the output trace files generated by the simulator.

3.1.1. RTP Management

In the *rtp_gs* files, we have re-defined the native RTP packet header, with the exact format specified in [4]. Moreover, here we have re-implemented the *RTP Agent* and improved it in order to be capable of generating any kind of traffic pattern passed from the NS-2 Application layer (such as Pareto, CBR, Exponential or Traffic Trace Files generated from real multimedia applications) in contrast to the native *RTP Agent* which is only capable of generating CBR Traffic. Therefore, new RTP packets would be generated based on the reception of new data unit blocks from the application layer (if the data unit block is larger than the Maximum Segment Size, which can be set in each simulation, the *RTP Agent* will fragment the data payload into several packets). Next, the packet headers are filled and the RTP packets are sent to the destination/s in a unicast or multicast way.

When incoming RTP packets are received by a receiver *RTP Agent*, it passes them to the *RTP Session* instance in order to process and register the sender statistics. Each *RTP Session* object keeps a list of the active sources in the session (it is called *allsrcs_*). Thus, the *RTP Session* calls a '*look up*' function for checking if the originator of this packet is an already known source. Otherwise, it will create a new *RTP Source* object in order to store some useful information from this new sender, such as the total number of packets and octets that have been received, the cumulative number of lost packets, the highest sequence number that has been received and the jitter value for that one. For successive incoming RTP packets, the receiver will look up the originator source in its list of sources according to the incoming RTP packets source identifier (if it receives data from more than one source) and will update the aforementioned sender information. Receivers are continuously checking the variations of the inter-arrival time of incoming RTP packets and estimating the delay jitter, as specified in [4]. Let us consider S_i be the RTP Timestamp of *n-th* RTP packet, and R_i be its arrival time, in RTP Timestamp units. Hence, time difference for two consecutive *i-th* and *j-th* packets, $D_{i,j}$, may be expressed as:

$$D_{i,j} = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i) \qquad (4)$$

This delay variation is calculated for each incoming RTP packet, and it is smoothed according to equation (5), which gives only a small weight to the most recent observation in order to avoid temporary fluctuation:

$$J_i = (15/16) \times J_{i-1} + (1/16) \times D_{i-1,i} \qquad (5)$$

As an example, the following piece of C++ code deals with the RTP packets reception and the subsequent measurement of the jitter values:

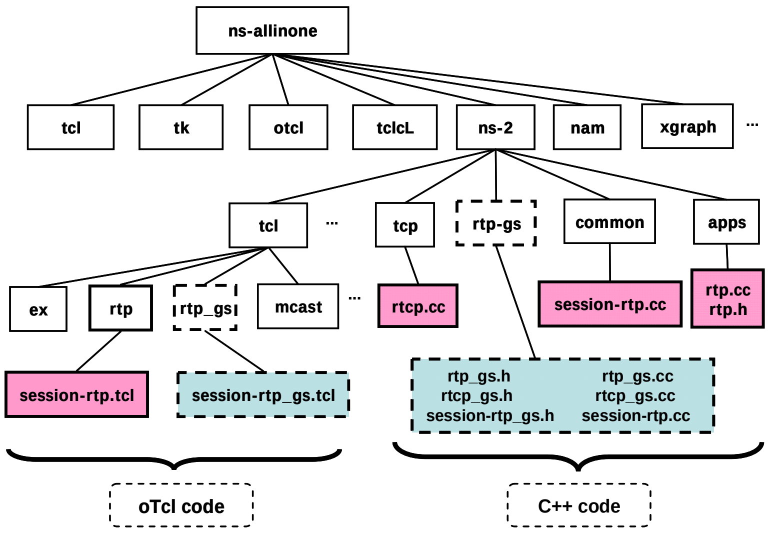

Figure 1 NS-2 Directory Structure.

```
/* jitter calculation according to RFC 3550 */
double last_one_way_delay_ = NOW - rh->timestamp();
double d = last_one_way_delay_ - s->prev_one_way_delay();
/* hold the most recent one_way_delay_ in order to use it for the next jitter
calculation */
s-> prev_one_way_delay(last_one_way_delay_);
if (d < 0) d = -d;
/* hold the jitter for the next calculation */
s->jitter(s->jitter()+(1./16.)*(d-s->jitter()));
```

We provide to the NS-2 users the possibility, from their OTcl script, to specify the name of a new generated output trace file, in the same path the script is executed, in which the above RTP statistics will be recorded for each simulation.

3.1.2 RTCP Management

In *rtcp_gs* files, we have created a new common header for all the RTCP packets, since, as discussed previously, the native code makes use of the same packet header to generate both RTP and RTCP packets. We have also defined new data structures for each type of RTCP packet, with the exact format specified in [4]. Finally, we have also re-implemented the *RTCP Agent* in order to be capable of managing the sending and receiving processes of the new included and adapted control messages.

The participants of an *RTP Session* will exchange, more or less periodically, RTCP feedback reports to inform about QoS statistics. If the feedback reports from each participant were exchanged at a constant rate, the control traffic would grow linearly with the number of participants. Therefore, the feedback rate must be scaled down by dynamically calculating the RTCP report interval in order to satisfy a trade-off between up-to-date information and the amount of control traffic. Authors in [4] propose that the fraction of the session bandwidth added by RTCP be restricted at 5 %. Due to this, a new *RTCP Timer* has been defined in order to take into consideration the RTCP report interval regulation and a new *build report* function will be called as a result of each timer timeout event. This function calls a new *adapt-timer* OTcl procedure in order to adaptively compute the next report interval, according to the session bandwidth and the number of active participants in the session. On the one hand, in this *build report* function, an RTP sender generates new SR if it has sent RTP data units since the previous sent SR. These reports describe the amount of data (packets and bytes) sent so far, as well as correlating the RTP sampling timestamp (local time) and absolute reference time (e.g. provided by NTP, GPS or other solutions) to allow synchronization between session participants. On the other hand, each participant constructs new RR if it has received new RTP data packets from a source since the previous report. Each RR contains one block for each RTP source in the session, including the instantaneous and cumulative loss rate, jitter and delay information from that source. Every receiver can measure the packet loss rate based on RTP packets sequence numbers. Therefore, the receivers must include in each RR the cumulative number of lost packets since the beginning of the session and the fraction of lost packets between two successive RTCP reports. The following piece of C++ code, which is located in the *build report* function, deals with the fraction lost measurement:

```
/* sp refers to a specific RTP Source */
/* received packets since last report */
int received_ = sp->np() - sp->snp();
if (received_ == 0) continue;
/* ehsr: extended highest sequence number received */
int expected_ = sp->ehsr()-last_ehsr_;
last_ehsr_ = sp->ehsr();
int lost_ = expected_ - received_;
if (lost_ <= 0 || expected_ == 0 )  flost = 0;
else  flost_ = ((double)lost_/(double)expected_);
```

RR may also be used to estimate the RTT between the sender and any receiver. The RR packet includes the LSR (timestamp of Last SR) and the DLSR (Delay since Last SR reception) fields, from which the sender can directly calculate the RTT, as specified in [4]. This process is illustrated in Figure 2.

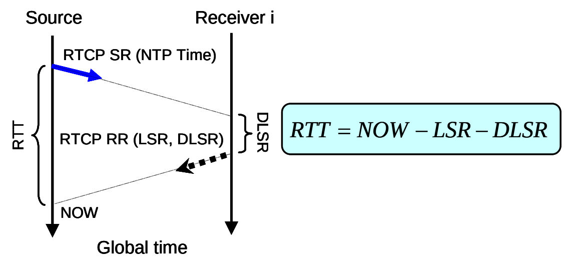

Figure 2 Round-trip delay calculation.

During the streaming session, the sender (receiver) statistics included in SR (RR) messages are registered in the appropriate *RTP Source* (*RTP Receiver*) instances. As in RTP management, we provide to the NS-2 users the possibility, from their OTcl scripts, to specify the name of a new generated output trace file, in the same path the script is executed, in which the report number, the generation timestamp, the reception instant, the RTT value, the fraction lost and the cumulative number of packet lost since the beginning of the session for each RTCP RR received by a specific *RTP Source* will be recorded. Furthermore, users can specify the name of another trace file in which the report number, the sending time, the reception instant and the number of packets/octets sent for each incoming RTCP SR in a specific receiver node will be recorded.

RTCP SDES packets have been also defined. They contain a participant's globally unique identifier, called *Canonical Name* (*cname*), which can be used for resolving conflicts in the SSRC value and to associate a set of concurrent RTP media streams from a given participant (e.g. video sender) in a multimedia session. SDES messages must be included in each compound RTCP packet [4]. If a participant leaves an *RTP Session*, it sends a RTCP BYE message. When active members in the session receive this BYE packet, the statistics of the originator SSRC in this message are removed from their participants' table (senders or receivers) and the number of active participants is updated. Finally, new RTCP APP packets are defined to send useful profile-specific information required by particular applications to the session participants.

4. Tool-Set Structure

Figure 3 illustrates the structure of the designed video delivery simulation tool-set, the interactions between the different multimedia tools and the necessary steps to run a new video streaming simulation. The tool-set has been constructed by combining NS-2, including our new developed RTP/RTCP module, video encoders/decoders (such as *ffmpeg* [38]), new adapted programs for the generation of *Video Traffic Trace Files*, *VLC media player* [39], new adapted programs for the reconstruction of the received video files, and additional programs for video quality measurements, such as a *YUV Viewer* [40] and *VQMT* tool [41].

In order to test a new video streaming application using this tool-set we must follow three differentiated phases, namely pre-processing phase (it is sketched in the upper left corner of Figure 3), simulation phase (lower part of Figure 3) and post-processing phase (upper right corner of Figure 3).

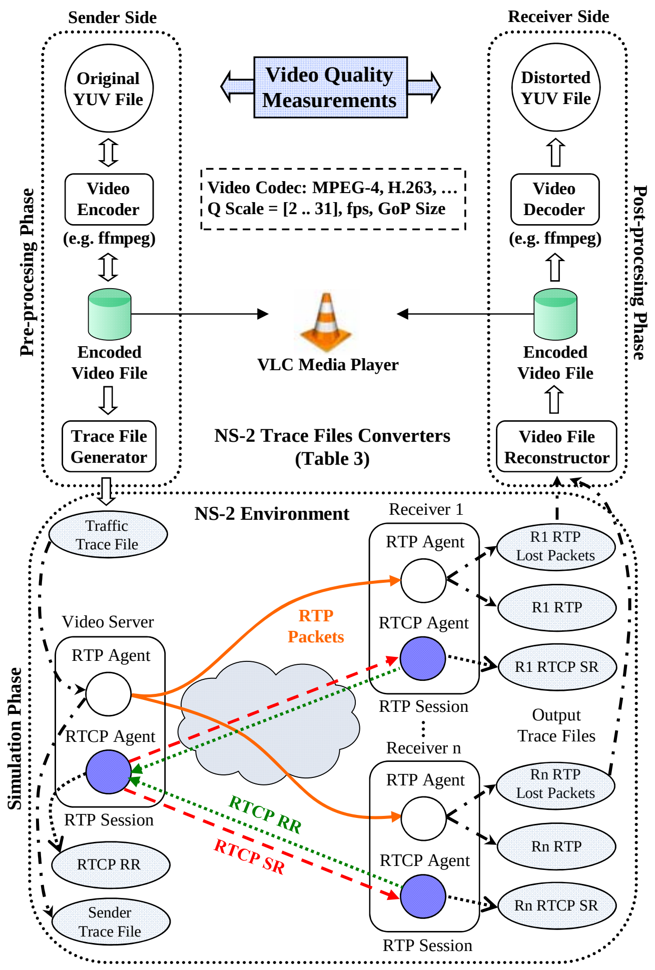

Figure 3 Tool-Set Structure.

4.1. Pre-processing Phase

The first step in order to begin a new video streaming simulation using this tool-set is to generate an encoded video file (e.g. H.263 or MPEG-4) from a given raw file, which is usually stored in YUV format. Otherwise, if we already have an encoded video file at our disposal, we can use a video decoder (e.g. *ffmpeg*) in order to create a reference video to use it for the video quality measurement at the post-processing phase. As an example, we can use *ffmpeg* tool for the creation of the MPEG-4 video clips. By using this video encoder, we can select the quantiser scale (Q), the frame rate and the GoP size for a given video clip. A GoP consists of exactly one Intra frame (I), some related Predictive frames (P) and, optionally, some Bidirectional frames (B) between these I and P frames. I frames are encoded independently of any other type of frame. They have the lowest compression and contain information from encoding a still image. P frames are encoded using predictions from the previous I and P frames. B frames are encoded bi-directionally from the preceding and the following I and P frames. B frames are encoded with the highest compression and require the lowest transmission bandwidth. The encoded video file is then converted, using a Video Trace File Generator, into an NS-2 compatible Traffic Trace File, which will be an abstraction of the real video stream, containing useful information, such as the frame number, frame type, frame size and generation time for each one of them, as shown in Table 3.

Table 3 Traffic Trace File Format

Frame Number	Frame Type	Frame Size (Bytes)	Time (s)
1	I	6488	0,0
2	P	1426	0,04
2	B	625	0,08
…	…	…	…
13	I	6777	0,52

4.2. Simulation Phase

The simulation phase can be initiated once the Video Trace File has been generated. At first, a new *RTP Session* instance have to be declared for each node in the simulated network that we want to take part in the multicast RTP Session as a participant, and new *RTP and RTCP Agents* have to be attached to that *RTP Session*. Logically, before the simulation start-up, every receiver has to be joined to the multicast group in which the video server will transmit the RTP stream. Next, the generated Video Trace File has to be attached to the *RTP Agent* declared for the video server node. This step will characterize the traffic profile of the video stream during the simulation process. For the video sender node, the user can specify the name of an output Sender Trace File (in which the size and sending time of each RTP packet sent by the *RTP Agent* attached to the video server will be recorded). At the receiver side, the incoming RTP packets are passed to the *RTP Session* instance in order to process and register the sender statistics from the originator source, such as the highest sequence number received, receiving time, network delay, the total number of packets and octets that have been received, cumulative number of packets lost and the jitter value for the last one. Furthermore, for each video receiver, those statistics can also be recorded in an output RTP Receiver Trace File (Table 4). Due to the system dynamics, such as network congestion, some RTP packet might be lost and would not arrive to some of the distributed RTP receivers. Thus, a new RTP Lost Trace File can be specified in order to register the sequence number of each expected RTP packet that has not been received. On the other hand, all the useful statistics that are carried out in the received RTCP reports can be recorded in new defined output SR/RR Trace Files (Figure 3).

Table 4 RTP Receiver Trace File Format

Sequence Number	**Size (Bytes)**	**Receiving Time (s)**	**Delay (s)**	**Jitter (s)**	**Lost Packets**
193	1024	0,0561	0,0561	0,0	0
194	1024	0,04755	0,0553	0,0028	0
…		…		…	
212	540	0,971	0,061	0,0031	1

4.3. Post-processing Phase

As discussed previously, we can process the network-level QoS measurements for the video streaming application from the statistics recorded in the output RTP and RTCP SR/RR *Trace Files*. Additionally, when the video transmission is over, the RTP statistics (delay and loss information), recorded in the output RTP trace files, can be used to reconstruct the transmitted video files at the receiver side by using the new adapted *Video File Generator program*. If the output trace files contain information about lost packets, the entire video frames to which these packets belong are considered lost since they could not be correctly decoded. Thus, this program has to insert all missing frames due to drops (or late arrivals) so that sent and received video consist of equal number of video frames. This process is performed by copying the last successfully decoded frame to each frame that has been lost, as a simple error concealment technique. Consequently, the resultant video file would be similar to the video stream received by the simulated client. The reconstructed (possibly distorted) video files can already be played by a media player, such as *VLC*. However, they must be decoded, e.g. using *ffmpeg* tool, to a raw file in order to use it for evaluating the quality of the end-to-end video (e.g., using *VQMT program*). Reconstructed video files at receiver side can also be played using a *YUV viewer program* in order to obtain a subjective assessment, as the one in [5], but it would be more complex and resource consuming.

5. Simulation Example

The unique purpose of this Section is to show the proper functioning of our new developed RTP/RTCP module and the ability provided by our proposed tool-set to achieve both network level measurements and objective video quality metrics. The discussion about the obtained results is not important for us but only the functionality of our tool-set that provide a platform for objective and subjective testing (the distorted video clips can be reconstructed and viewed by users). Subjective assessment is left for further study. The simulation topology and the configuration of its parameters are described in Section 5.1; network-level performance results are presented in Section 5.2; and application-level measurements are shown in Section 5.3.

5.1. Simulation Setup

The simulation scenario consisted of an IP Backbone Network and three distributed Local Area Networks (LAN 1, LAN 2 and LAN 3). All the links in the topology were bidirectional and had 10 Mbps capacity. For the video transmission, we concatenated (using Unix *cat* tool) the popular "*foreman_cif.yuv*" and "*news_cif.yuv*" raw files, both in YUV 4:2:0 format, available in [22]. Each sequence contained 299 frames with Common Intermediate Format (CIF) size (352x288 pixels). We encoded the resultant raw sequence to an MPEG-4 video file using *ffmpeg* tool with a specific rate of 25 frames/s and a fixed GoP size of 12 frames (the quantiser scale was varied in each simulation Q=[2..31]), with pattern IBBPBBPBBPBB(I). Thus, the duration of the resultant media file was 24 seconds. The video traffic was transmitted by an RTP Source (located in LAN 1) to a multicast group with three distributed RTP Receivers (R1, R2 and R3), each one in a different LAN (with different end-to-end delays and different network load between the video server and receivers). The maximum transmission RTP packet size was set to 1024 bytes. As background traffic, we configured different intensive FTP, CBR/UDP and Pareto distribution traffic flows over the network topology in order to force jitter variability and possible packet loss during the simulations. For the multicast video transmission, we configured PIM-DM (*Protocol Independent Multicast – Dense Mode*) protocol.

5.2. *Network-level Measurements*

In order to evaluate the end-to-end delay for the video traffic during simulation time, we ran two different simulations with different network load. First, we ran a single simulation without background traffic. In that case only RTP traffic and feedback RTCP reports were exchanged. Next, we increased the network load by transmitting intensive background traffic in order to assure that the total amount of network traffic (video + background) was near the links' capacity in some instants during the multimedia session.

Figure 4 illustrates the RTT obtained from feedback information carried by each RTCP RR packet sent by one of the receivers (we show the results only for R3 that was in the worst network conditions) during the multimedia session for the two network load simulated cases. For the other receivers the results were similar. As expected, as the network traffic increased, the RTT also increased due to the higher queuing delays in the routers (as the video traffic must contend with the background traffic). Figure 5 shows the jitter values for that receiver when the background traffic was applied (without background traffic the jitter values were negligible). The jitter values could also be measured by the video server from each RTCP RR packet sent by that receiver.

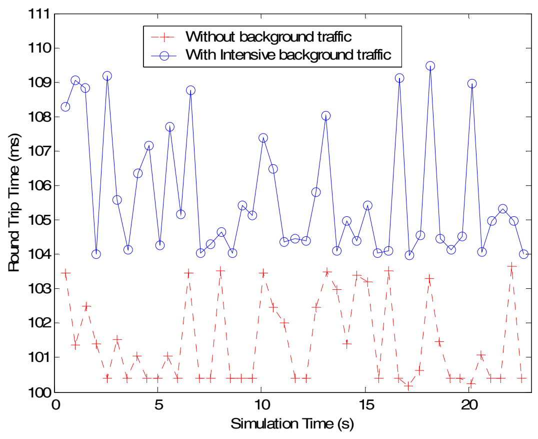

Figure 4 RTT under different Network Load.

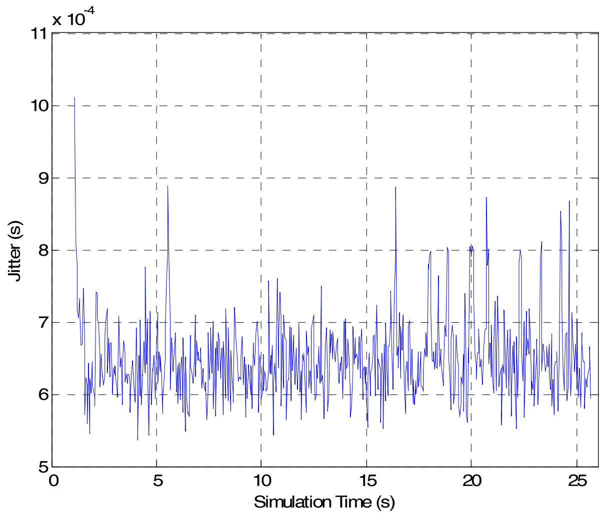

Figure 5 Jitter Measurements.

Figure 6 illustrates, also for R3, the average throughput of the received video stream regarding all the possible quantiser scales (Q=[2..31]) for the encoded video clip. As expected, the higher the Q was the lower the throughput was, because a smaller Q means a lower compression rate for a specific video clip.

Additionally, we configured a Packet Error Rate (PER) module in order to force losses in a specific link of the network topology and, thus, simulate several network congestion cases. The PER was varied in each simulation in the range from 0 to 0.1 (0-10%), with 0.01 (1%) intervals, following a random uniform error distribution model. Figure 7 illustrates the fraction of packets lost carried in each RTCP RR sent by R3 (i.e., the number of packets lost over the number of packets expected in the time interval between two successive RR) regarding different values of PER.

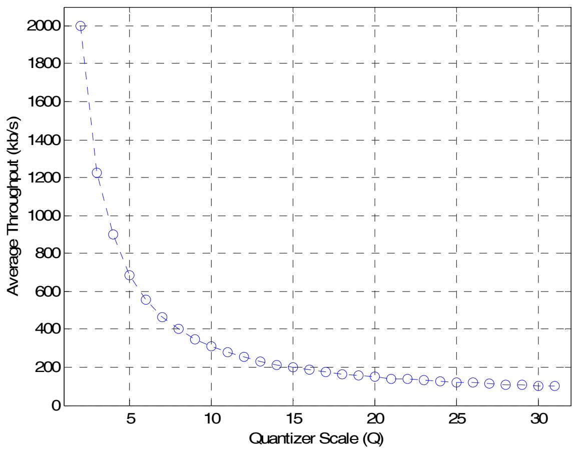

Figure 6 Video Bit Rate vs Compression Rate (Q).

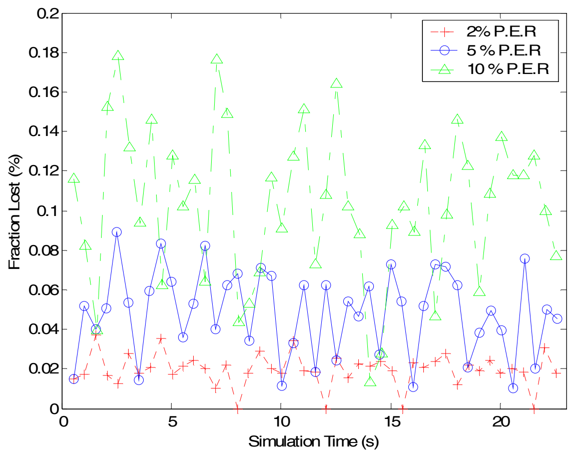

Figure 7 Fraction of Lost Packets.

5.3. Application-level Measurements

Once the different simulations were finished, we used *VQMT tool* to obtain the results shown in Table 5, which presents the average values of some of the common objective metrics used for video quality assessment, such as *PSNR*, *SSIM* and *VQM*. Although, these metrics were introduced in Section 2, the interpretation of each one of them is summarized in Table 6 to facilitate the understanding of the obtained results. The values in Table 5 prove that the perceived quality of the received video stream, for each one of the measured video quality metrics, became gradually worse with the increasing of PER (i.e. higher percentage of lost frames) during the multimedia session, as expected.

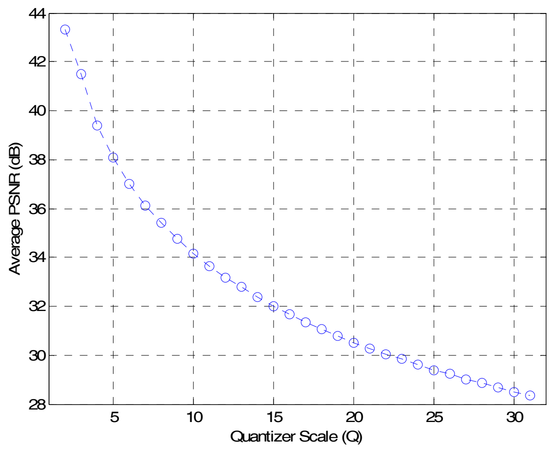

Figure 8 PSNR vs Compression Rate (Q).

Additionally, we also measured the effect of the video encoding rate on the perceived video quality. Figure 8 represents the mean *PSNR* values between the original raw video file (no compression) and the reconstructed ones at the R3 receiver side for all the possible values of quantiser scales (Q). As expected, the average *PSNR* value, and thus the video quality, was significantly reduced when a higher compression rate was employed in the video transmission, mainly due to the loss of information and the introduction of some distortions in the compression (encoding) process. Thus, a reduced value of Q in the video encoding process gives a good video quality (Figure 8) but, in turn, it consumes a high bit rate in the video transmission process (Figure 6), and vice versa. Notice that with the obtained results we only want to prove the ability of our tool-set to get them. Thus, that is the reason why we do not discuss them in detail.

Table 5 Video Quality Measurements

P.E.R (%)	**PSNR(dB)**	**SSIM**	**VQM**	**Blocking Metric**
0	52,57	1	0,06	14,29
2	29,62	0,78	4,59	14,47
4	24,76	0,73	6,07	15,26
6	21,8	0,679	8,59	18,24
8	20,1	0,65	8,74	19,12
10	18,74	0,6	11,59	23,97

Table 6 Video Quality Metrics Interpretation

Metric	**Interpretation**
PSNR	Infinite for equal frames, higher values are better
SSIM	1 for equal frames, higher values are better
VQM	0 for equal frames, lower values are better
Blocking Metric	Lower value corresponds to lower blocking.

6. Conclusions and Future Work

In this work, we have presented an evolved version of an RTP/RTCP-based tool-set for video streaming evaluation in simulated environments. We have combined a new designed NS-2 module with the most complete implementation of RTP/RTCP protocols with several multimedia tools, such as *VLC,* media encoders/decoders (e.g. *ffmpeg*), a new adapted *Video Trace File Generator*, a new program for the received video file reconstruction, a *YUV Viewer* and *MVQT application*, to obtain a simulation platform that allows both network-level and application-level measurements that can reflect the performance of video streaming applications. We have shown the usefulness of the presented tool-set by giving some results of a simple evaluation of a multicast video transmission under different network conditions. As RTCP feedback capabilities have been included, researchers interested in multimedia systems could test novel source or receiver-based proposals with slight modifications to our tool-set. For imminent future work, we plan to implement and evaluate (by extending this tool-set) our source-based multimedia group or inter-destination synchronization proposal [5], which is based on the extension of new RTCP reports and aims to synchronize the playout of a group of distributed video receivers. The source code, installation guides, the necessary tools of the proposed platform, a complete guide to configure a new multimedia streaming simulation and several simulation examples will soon be available at author's web page: *http://personales.gan.upv.es/~fboronat*.

Acknowledgements

This work has been financed, partially, by Polytechnics University of Valencia (UPV), under its R&D Support Program in PAID-06-08-002-585 Project and in PAID-01-10, and by Generalitat Valenciana, under its R&D Support Program in GV 2010/009 Project.

References

1. NS-2 Simulator Home Page, http://www.isi.edu/nsnam/ns
2. OPNET Simulator Home Page, http://www.opnet.com
3. OMNET Simulator Home Page, http://www.omnetpp.org
4. Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V., RTP: a transport protocol for real-time applications, RFC-3550, July 2003.
5. Boronat, F., Guerri, JC. and Lloret, J., An RTP/RTCP based approach for multimedia group and inter-stream synchronization. Multimedia Tools and Applications Journal, Vol. 40 (2), 285-319, 2008.
6. Begen, A., Perkins, C. and Ött, J., On the use of RTP for Monitoring and Fault Isolation in IPTV. IEEE Network, Volume 24 Issue 2, March/April 2010.
7. Montagud, M. and Boronat, F., A new network simulator 2 (NS-2) module based on RTP/RTCP protocols to achieve multimedia group synchronization. In SIMUTOOLS 2010: 3rd International ICST Conference on Simulation Tools and Techniques, Torremolinos, Malaga, Spain, March 2010.
8. Boronat, F., Montagud, M. and Vidal, V., Enhanced RTP-based Tool-Set for Video Streaming Simulation using NS-2. In MoMM 2010: The 8th International Conference on Advances on Mobile Computing and Multimedia, Paris, November 2010.
9. Winkler, F. and Mohandas, P., The Evolution of Video Quality Measurement: From PSNR to Hybrid Metrics. IEEE Transactions on Broadcasting, Vol. 54, No. 3, pp. 660-668, September 2008.
10. Golaup, A. and Aghvami, H., A multimedia traffic modelling framework for simulation-based performance evaluation studies. Computer Networks, 50, 12 (Aug. 2006), 2071-2087.
11. Seeling, P., Reisslein, M. and Kulapala, B., Network Performance Evaluation Using Frame Sizes and Quality Traces of Single-Layer anf Two-Layer Video: A Tutorial. IEEE Communications Surveys and Tutorials, Vol. 6, No. 2, pp 58-78, Third Quarter 2004.
12. Fitzek, F.H.P. and Reissling, M., MPEG-4 and H.263 Video Traces for Network Performance Evaluation. IEEE Network, Vol. 15, No. 6, pp. 40-54, December 2001.
13. Klaue, J., Rathke, B. and Wolisz, A., EvalVid – A Framework for video transmission and quality evaluation. In Proceedings of the 13th International Conference on Modelling, Techniques and Tools for Computer Performance Evaluation, Urbana, Illinois, 2003.
14. Kao, K., Ke C. and Shieh, C., An advanced simulation tool-set for video transmission performance evaluation. In WNS2'06: the 2006 Workshop on Ns-2: the IP Network Simulator, Pisa, Italy, October 10 - 10, 2006, vol. 202. ACM, New York, NY, 10.
15. Ke, C.H., Shieh, C.K., Hwang, W.S. and Ziviani, A., An Evaluation Framework for More Realistic Simulations of MPEG Video Transmission. Journal of Information Science and Engineering, vol. 24, n 2, pp. 425-440, 2008.
16. Abdel-Hady M. and Ward, R., A Framework for Evaluating Video Transmission over Wireless Ad Hoc Networks. In PACRIM 2007: IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp.78-81. 2007.

17. Yu, C-Y, Ke, C-H, Chen, R-S, Shieh, C-K, Munir, B. and Chilamkurti, N. K., MyEvalvid_RTP: A evaluation framework for more realistic simulations of multimedia transmissions. International Journal of Software Engineering and Its Applications Vol. 2, No. 2, April, 2008.
18. Lie, A. and Klaue, J., Evalvid-RA: Trace Driven Simulation of Rate Adaptive MPEG-4 VBR Video. Multimedia Systems, Springer Berlin / Heidelberg, Volume 14, Number 1 / June, pp. 33-50, 2008.
19. Bouras, C., Gkamas, A. and Kioumourtzis, G., Evaluation of Single Rate Multicast Congestion Control Schemes for MPEG-4 Video Transmission. In NGI'09: 5th Euro-NGI conference on Next Generation Internet networks, Aveiro, Portugal, July.
20. Yan Z., Liu G., Su R., Zhang Q., Chen X., and Chen L., A Simulation Mechanism for Video Delivery Researches. In ChinaCom 2009: the Fourth International Conference on Communications and Networking in China, China, August 2009.
21. Suki, A., Hassan, S., Ghazali, O. and Awang, S., Evalvid-RASV: Shaped VBR Rate Adaption Stored Video System. In ICETC 2010: the 2nd International Conference on Educational Technology and Computer, Shanghai, China, June 2010.
22. Klein, A. and Klaue, J., Performance Evaluation Framework for Video Applications in Mobile Networks. In IEEE MESH 2009, Athens, Greece, June 2009.
23. Klein, A. and Klaue, J., Performance Study of a Video Application over Multi Hop Wireless Networks with Statistic-Based Routing. In IFIP Networking, 2009.
24. YUV Video Files: http://www.tkn.tu-berlin.de/research/evalvid
25. Video Traces Library: http://www.tkn.tu-berlin.de/research/trace/trace.html
26. RTPTools, http://www.cs.columbia.edu/irt/software/rtptools
27. ITU-R Recommendation BT.500-11, Methodology for the subjective assessment of the quality of television pictures. International Telecommunication Union, Geneva, Switzerland, 2002.
28. ITU-T Recommendation P.910, Subjective video quality assessment methods for multimedia applications. International Telecommunication Union, Geneva, Switzerland, 2008.
29. Chan, A., Zeng, K., Mohapatra, P., Lee, S. and Banerjee, S., Metrics for Evaluating Video Streaming Quality in Lossy IEEE 802.11 Wireless Networks. In IEEE Infocom 2010.
30. Ohm, J.R., Multimedia Communication Technology, Springer, Heidelberg (2004).
31. Korhonen J. and You J., Improving objective video quality assessment with content analysis. In the Fifth International Workshop on Video Processing and Quality Metric for Consumer Electronics, Melbourne, Australia, January 2010.
32. Wang, Z., Lu, L. and Bovik, A. C., Video quality assessment based on structural distortion measurements. Signal Processing: Image Communication, vol. 19, no. 2, pp 121-132, February 2004.
33. Kotevski, Z. and Mitrevski, P., Experimental Comparison of PSNR and SSIM Metrics for Video Quality Estimation. in ICT Innovations 2009.
34. Pinson and, M.H. and Wolf, S., A new standardized method for objectively measuring video quality. IEEE Transactions on Broadcasting, vol. 50, no. 3, pp. 312-322, September 2004.
35. Issariyakul, T. and Hossain, E., Introduction to Network Simulator 2 NS-2. Springer, ISBN: 978-0-387-71759-3, 2009.
36. Carrascal, V., Grupo de Servicios Telemáticos, Universidad Politécnica de Cataluña, España. http://sertel.upc.es/_vcarrascal/ns2
37. Bouras, C., Gkamas A. and Kioumoutzis, G., Extending the Funtionality of RTP/RTCP Implementation in Network Simulator (NS-2) to support TCP friendly congestion control. In SIMUTools 2008: the 1st International ICST Conference on Simulation Tools and Techniques, Marseille, France March 3-7.
38. VLC Media Player, http://www.videolan.org/vlc
39. FFMPEG program, http://sourceforge.net/projects/ffmpeg/

40. YUV viewer, http://www.brothersoft.com/elecard-yuv-viewer-download-142207.html
41. MSU Video Quality Measurement Tool. Graphics&Media Lab, Computer Science (VMK), Moscow State University, http://compression.ru/video/quality_measure/video_measurement_tool_en.html