Adaptive multimedia streaming using a simulation test bed

S. Linck[a], E. Mory[b], J. Bourgeois, E. Dedu, F. Spies[c]

[a] *Université de Reims Champagne-Ardenne*
7 Bld Jean Delautre - 08000 Charleville-Mézières, FRANCE
Email : sebastien.linck@univ-reims.fr
[b] *Email : emmanuel.mory@orange.com*
[c] *Université de Franche-Comté - FEMTO-ST Institute, UMR CNRS 6174*
1 Cours Leprince-Ringuet - 25200 Montbéliard, FRANCE
Email : {julien.bourgeois, eugen.dedu, francois.spies}@femto-st.fr

Abstract

The major purpose of this paper is to transmit an existing video above a complete simulated video streaming architecture based on Network Simulator (NS2). Thanks to our architecture, the visual quality evaluation of the distributed streaming platform under various conditions is simplified. Indeed, the received video can be easily visualized using a classic video client or be compared using the Peak Signal-Noise Ratio (PSNR) value and the Structural SIMilarity (SSIM) value. In the case study, we compare adaptive video congestion strategies using a transcoder, Datagram Congestion Control Protocol (DCCP) and TCP-Friendly Rate Control (TFRC).

Keywords: network simulator, video quality estimation, transmission protocol, mixer, video streaming, wireless network

1. Introduction

Multimedia streaming over Internet is nowadays widely used. But existing solutions to stream multimedia contents are numerous. Currently, there is no ideal solution. Some of them are standardized by the Internet Engineering Task Force (IETF), such as RTP [1], RTSP[2]. Others are proprietary, such as RealNetwork[1]. Currently, video web broadcasters like youtube.com or dailymotion.com send only fixed video files. Thus, they transmit poor quality and small definition video file in order to prevent throughput decreasing. If the available bandwidth drops below the video bitrate during the transmission, the buffer will be empty and the video will stop. Video web broadcasters offer sometimes better quality of few videos, but it is the user responsability to try it.

At the transport level, several solutions exist too. Some of them do not use any congestion control, such as RTP/UDP, others use control congestion for fixed files (which are the same all the time), such as HTTP/TCP. The main drawbacks of UDP (sending video streams) are not supporting fairness with other flows and not collecting feedback information concerning the packet delivery. In fact, UDP video streams are mainly used on dedicated networks with a good

[1] Home page: `http://www.realnetworks.com/`.

static quality, such as Internet providers networks, but are not used over the whole Internet. At the contrario, TCP drawbacks (sending video streams) are introducing delays in transmission, buffering packets and delivering sequentially packets to the destination by retransmitting lost packets which lead to increase delays. In fact, TCP video streams can be used on the Internet, but with a low video quality.

The goals of this paper are to study the adaptive video streaming in order to transmit higher quality video streams over Internet than currently proposed. Specific congestion control mechanisms are necessary in order to approach the maximum available or dedicated bandwidth. We propose such a mechanism which is included in a mixer component. Thanks to this mechanism, it will be easier to transmit video streams over wireless networks which include in its definition a variable bandwidth. Moreover, we developped a simulation test bed in order to classify video transport solutions and to be able to build objective comparisons based on reproducible situations. These comparisons can be realized on network and video parameters. Because, video evaluators such as PSNR (Peak Signal-to-Noise Ratio) or SSIM (Structural Similarity) [3] can determine the quality of the streamed video in our simulator. Finally, this test bed allows us to practice cross-layers techniques between transport layer and application layer. Simulated transport layers are TCP, UDP and DCCP [4].

The transport protocol DCCP has been evaluated into our test bed, because TCP and UDP limit video stream possibilities due to some drawbacks. DCCP is useful because it offers internal mechanisms which allow to define new congestion control strategies. Indeed, DCCP allows to implement and to compare strategies adapted to the transport of multimedia contents. One of these strategies is the adaptation of a video to the available troughput. This allows the stream to respect real time constraints. This kind of operation can only be realized by a mixer such as defined in [1]. The congestion control used by the video must be TCP-friendly, use the allocated throughput as best it can and have the best visual aspect. All of these components are included in our test bed.

Real time multimedia adds complexity to this, not only because several protocols can be used, but mainly because the data received on the client side needs to be computed according to the video input data. Also, multimedia data is sensitive to wireless specifics. Being able to simulate a video on demand (VoD) architecture, comprising a server, a mixer (see below) and a client opens up real possibilities for the optimization and the comparison of elaborate strategies in a real context of flow competition. Therefore, a flexible test bed is needed which can deal with various transport protocols.

Our contribution is to design and implement a simulation test bed for generic multimedia streaming, with the following main features:

Flexibility: various protocols may be simulated, such as UDP, DCCP, TCP.

Horizontal and vertical modularity: it is easy to plug protocols into the simulation environment, both horizontally, i.e. interpose/modify protocols between client and server (such as DCCP), and vertically, i.e. interpose/modify protocols of OSI levels on a resource (e.g. on the client side).

Based on a well-known simulator: the Network Simulator (NS2 [5]) is frequently used in the research area.

Real application data use: even if the network is simulated, the data are not simulated, but are computed exactly from the input data set; this allows to give more accurate results when

analyzing congestion control protocols for example. Our test bed evaluates the final quality

2.2. Transport level: DCCP congestion controls

The vertical modularity of our test bed allows us to add or change the modules inside the client or the server easily.

Two transport protocols prevail nowadays: TCP and UDP. DCCP [4] is a recently standardized transport protocol sharing characteristics from both of them: It has congestion control mechanisms like TCP, and it is unreliable like UDP.

DCCP separates the transport of packets from the flow congestion control (CC). Each flow can choose the most appropriate CC. Currently, two CC mechanisms are provided: TCP-Like [7] and TFRC [8].

TCP-like resembling TCP uses a congestion window. The window increases when there is no packet loss and decreases by half when there are losses. The congestion window as well as the bandwidth have abrupt changes, which is not appropriate for video streaming.

TFRC uses an equation for the bandwidth. The equation, based on TCP Reno [8], is used regularly, for example once per RTT. It allows moderate bandwidth changes and is more appropriate to video streaming.

3. Related work

3.1. Video streaming using DCCP/TFRC

DCCP adds congestion control to UDP and it has successfully been used in video streaming. [9] presents an implementation of TFRC in Linux, a codec and a video conferencing system with low latency, combining these elements. It divides the video system in three components: the codec, the DCCP module, and the algorithm deciding the video quality to use. The variables used for quality changing are: resolution, JPEG quality and frame rate.

3.2. TFRC in wireless links

TFRC uses a formula based on TCP Reno [8]. Therefore, over wireless links it has the same drawbacks as TCP, the most notable is that a packet loss is considered as a congestion event. Several approaches have been proposed to adapt TFRC to wireless links.

[10] analyses losses in wireless links. The authors propose multiple connections for a video stream and deduce the following rule: "Keep increasing the number of connections until an additional connection results in an increase of end-to-end RTT or packet loss rate". Based on the RTT variation, the number of connections n is increased by α/n or decreased by β, where α and β are constant.

3.3. Video transcoding

Three categories of video transcoding that modify the bitrate of the streamed video have been developed [11]. The first one is referred to as closed loop transcoding or Cascaded Pixel Domain Transcoder (CPDT) [12]. The video is completely decoded, possibly modified and then encoded, this is the solution chosen for our mixer. The second category called Open Loop Transcoding (OLT) [13] does not completely decode the stream but stops at the DCT phase. This solution saves CPU time but the resulting quality is not as good as the CPDT solution. Finally, the third one is an intermediate solution that pushes the decoding process deeper than OLT. This category is named DCT Domain Transcoder (DDT) [14] and an implementation has been realized [15].

It is also possible to transcode by changing the resolution of the video [16, 17, 18] or by modifying the picture frequency [19, 20].

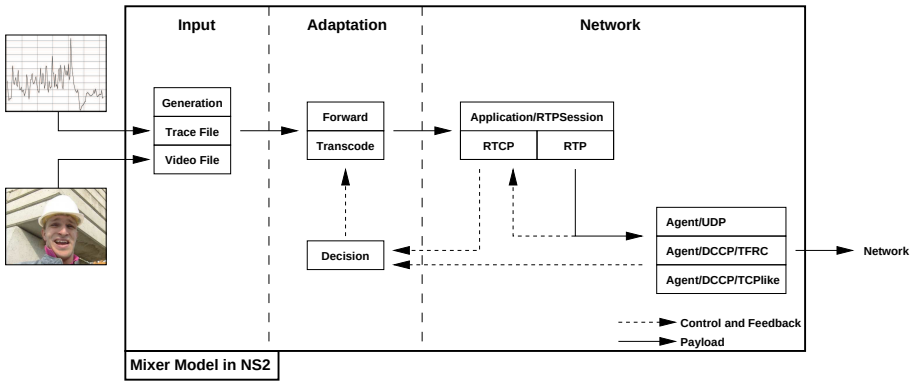

Figure 1: Mixer model in NS2.

4. Simulation testbed

4.1. General architecture

The general architecture of our simulation test bed is composed of two main parts: the mixer and the client.

Figure 1 presents the model of the mixer as it has been modeled into NS2. Three kinds of video data can be used as input of the mixer:

- Generated data: The mixer generates its own data by using various algorithms. The distribution of the various kinds of pictures (I, P and B) is given by Gismo [21].
- Real traces: During a real transmission, the size of packets sent over the network is analyzed and the characteristics of the packets are stored so that they may be used in NS2.
- Real streaming: This mode makes it possible to use real videos in NS2. The packetization is carried out by the mixer. The packets are sent through the different NS2 modules and the video is really transmitted between the server and the final client.

The input is sent to the core of the mixer, that is to say to the adaptation module, which decides either simply to forward the stream or to transcode it in order to fit the available bandwidth better. The packets are sent to the network module which comprises an RTPSession application which manages the RTP and RTCP agents and the transport agent, for example, UDP or DCCP.

Figure 2 presents the model of the client as it has been modeled into NS2. The client receives the packets from the network, and he can reconstruct the video exactly as if he had played it. This video is output to a compressed video file which can be compared with the original video. Finally, the PSNR and SSIM can be calculated to see exactly what the effect of losses or jitter is on the resulting video.

This simulation test bed allows us to test different strategies to stream multimedia contents with various transport protocols. Besides, it also allows us to see the effects of the network problems on the streamed contents clearly. Indeed, during multimedia streaming the lost packets will not have the same effect on the video quality. Some packets will seriously affect the visual quality of the video whereas others will not. This difference depends on various parameters like

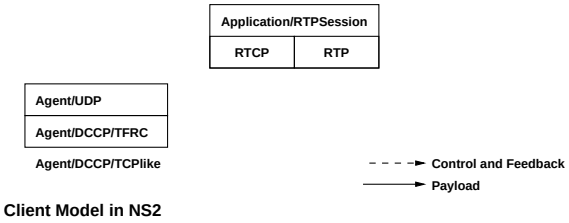

Figure 2: Client model in NS2.

the type of lost packet, the time the packet is lost in the group of pictures (GOP), etc. In fact, if the last P-frame of a GOP is lost, the quality will not be affected because, just after this picture, an intra picture will be decoded. As the last P-frame of a GOP and the first I-frame of the following GOP do not have any time dependance, only one frame will be damaged. Another example is when a packet is lost on a P-frame just before a camera movement. The resulting picture will be damaged but the camera movement will delete this error.

That is why it is necessary to calculate the video quality (with the PSNR and the SSIM) and not just to count the lost packets to evaluate the resulting video quality and this is the aim of our test bed.

4.2. The mixer

4.2.1. Architecture of the mixer

An RTP mixer [1] receives RTP packets from one or more sources, possibly changes the data format, combines the packets in some manner and then forwards a new RTP packet.

This mixer has been developed in the NetMoVie project [6], which is part of a larger project named MoVie. The mixer is not only RTP-compliant but has also extended functionalities like on-the-fly transcoding or adaptability of the contents to the constraints of the network.

It is located the closest to the client in order to react as soon as possible to the variation of the bandwidth. As the clients are connected to a wireless network, the mixer should be ideally located in the bridge between the wired and the wireless network. Putting the mixer near the server is not a good idea, since it deals many clients, and these clients have not necessarily the same bandwidth and CPU processing parameters.

The mixer modifies the transmitted data in order to adapt them to the client or to optimize the available bandwidth. The mixer is also able to deal with several types of coding, like hierarchical video or MPEG coding ones.

In the real environment, the mixer can be interfaced with any server that uses the RTP/RTCP and RTSP standard streaming protocols. The mixer also uses video coding libraries which gather several codecs like FFmpeg and XviD[3]. Tests were carried out by using a *Darwin Streaming Server* video server and clients like Quicktime or Videolan.

Our simulation test bed is used to test the various functionalities of the mixer in a specific environment.

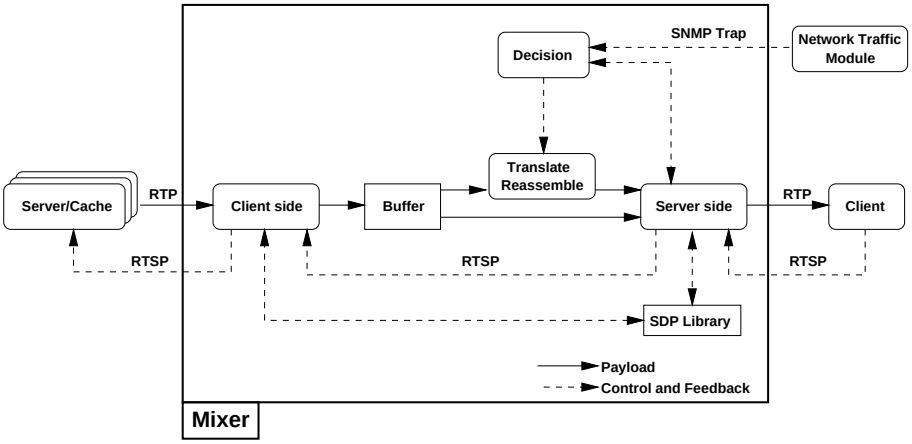

Figure 3: Internal architecture of the NetMoVie mixer.

4.2.2. The modules of the mixer

Figure 3 shows the various modules of the mixer:

Client side This module allows the mixer to be connected to the video servers.

Server side This module consists of the implementation of a RTP/RTSP server. The mixer is seen like a server from the clients' point of view.

Buffer The buffer is used to store a certain number of pictures before beginning the streaming to the client. The buffer and the transcoding module are the two most complex elements to manage. Above all, the mixer must be reactive in order to be able to adapt as fast as possible to a change in the streaming conditions. But the buffer should not be too much filled as the streaming is performed in nearly real time.

Transcode/reassemble This module is the element which makes it possible to adjust the quality of the stream according to the various constraints which act on the transmission. One of the major points of this module is the choice of changing the policy of adaptation. A video the compression of which comprises time dependances cannot be decoded directly at any time. Indeed, it is necessary to wait for a new intra picture, or to return to a preceding intra picture and to decode all the pictures until obtaining the current picture. Most of the time, a video is divided into independent blocks, called GOP (Group of Pictures). In order to optimize the buffer space, there will never be more than the current GOP stored in the buffer.

Decision This module takes into account all the parameters which are given to it: information feedback from the server module (RTCP reports, for example) or information on the available bandwidth coming from the network module (see more information at 4.3). Afterwards, it chooses the most appropriate video for the client according to the available bandwidth and the available video quality from the servers.

[3]Home page: `http://www.xvid.org`.

4.2.3. Description of a typical session

1. A client connects to the mixer by the intermediary of the server module and requests the visualization of the video.
2. The request is transmitted to the decision module which will ask the global architecture for the available quality for the required video.
3. The decision module will choose the video which is the most adapted to the client's characteristics and to the constraints of the network.
4. The client module requests the chosen video from the selected server. In the best case, a video the quality of which meets the needs, is directly available in the system and no transcoding or adaptation is necessary. If it is not the case, the decision module chooses the right transcoding method.
5. As soon as the stream is received by the server module, it is sent directly to the client.
6. If a quality change is necessary, the decision module asks the video server if a more adapted video is available. But in order to adapt the quality as soon as possible, it asks the transcoding module to change the quality of the stream while waiting for a new one.
7. As soon as the new video is available, the mixer stops the transcoding and streams the new video.

4.3. RTP and DCCP modifications for video streaming

To calculate an estimate of the current bandwidth, hence doing a congestion control, it is necessary to receive feedback from the client. The classical method is to use RTCP protocol. Unfortunately, the frequency of RTCP prevents an accurate calculation. Another method is to use other protocols, for example *Darwin Streaming Server* uses a protocol called *Reliable UDP* [22], which is based on RTCP/APP message. But the standard RTP/RTCP is not respected because the amount of RTCP packets is greater than what is recommended in the protocol.

To cope with these drawbacks, we decided to use a cross-layer approach, where the network layer provides the network bandwidth information to the mixer. DCCP (*Datagram Congestion Control Protocol*) is a transport protocol which, like UDP, does not guarantee the arrival of data but, like TCP, includes a congestion control. Currently, two congestion controls are available: TCP-Like and TFRC (*TCP-Friendly Rate Control*).

The DCCP congestion control works in the following manner:

- The receiver measures the error rate and returns this information to the sender.
- Thanks to the RTT and the error rate previously acquired, the DCCP/TFRC module on the sender calculates the possible flow rate.
- The mixer asks periodically the DCCP/TFRC module about the estimated available bandwidth. This information will be used to select the rate of the transcoded video, which is detailed below.

The estimated available bandwidth is calculated by TFRC using this TCP-friendly equation:

$$R_{TFRC} \cong \frac{s}{RTT\sqrt{\frac{2bp}{3}} + t_{RTO}(3\sqrt{\frac{3bp}{8}})p(1+32p^2)}$$

where R_{TFRC} represents the flow allowed for transmission in progress, s the size of a packet, p the loss rate, t_{RTO} the timeout, and b the number of packets acknowledged by a single ACK packet.

This information is used as follows: to be sure that our transcoded video is not blocked by the network (estimated banwidth is not always correctly estimated) we choose to take into account only 90% of the estimated bandwidth. This allows to be more often correct and it absorbs most of the peaks in the video bitrate.

DCCP/TFRC calculates its bandwidth once per RTT. Transcoding once per RTT is very time consuming for the mixer, and can be quality inefficient. So we propose a staircase transcoding as follows. The wireless network is often unstable at its bandwidth, so we propose two thresholds to avoid transcoding the video too frequently. We define an upper and a lower threshold associated to the current rate. The transcoding operation is performed only when the average bandwidth between the previous and the current bandwidth is outside the interval [lower, upper]. This average prevents too steep bandwidth changing. To avoid transcoding video for nothing (no change of quality) we have threshold on the mixer too. The lower and upper thresholds depend on current rate and a PSNR value. This PSNR value ensures that the new rate is worthwhile (for the video quality) to be used.

It is calculated as follows: PSNR values are empirically associated to bitrates (for the current video for example). Given that small differences in PSNR are not significant to human eyes, a difference of at least 2dB is needed. So, for the current bitrate, the bitrates giving a PSNR difference of at least 2 are considered as lower and upper thresholds.

Finally the transcoding is limited to obvious extrema: a minimal (for example 20kbps) and a maximal (original rate of the video) transcoding rate.

5. Experiments

The purpose of our system relates to the transmission of multimedia data. To this effect, several implementations have been achieved: a real implementation and in a simulator. The actual implementation can highlight current issues as well as in the current distribution architectures. The simulator allows us to explore specific scenarios that would inevitably be difficult to achieve in practice, or to recreate exactly the same scenario with different solutions. Network Simulator 2 is used in our test bed.

This simulator doesn't deal with the application data found in packets. However, when transmitting multimedia data, packet have not the same importance: depending whether a lost packet contained a RTP header or a macro-block of a video, the final result received by the client will be quite different. That is why the possibility of carrying the "real" data of a video during a simulation was developed within NS-2.

5.0.1. Scenario of simulations

The simulation scenario is illustrated in figure 4. In this network, the bandwidth varies over time. The shadowing propagation model is used. A real video encoded at a rate of 1Mbps is used. This video is chosen for its great complexity constant which can reach high speed in terms of compression. The mixer is connected to the base station through a full-duplex 100Mbps network and the mobile client is attached to this base station. The available bandwidth between it and the base station will be affected by its position. This simulation corresponds to a man walking on the street while watching music clip on demand (most watched on YouTube[23]) for example.

1. The mobile moves away linearly from the base station.
2. At $t = 100s$, it stops moving and stays motionless during $50s$.
3. Then it goes back to its initial position and the simulation ends at $t = 250s$.

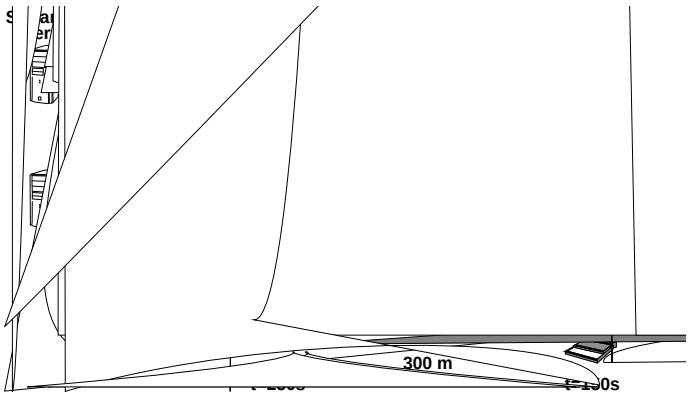

Figure 4: The simulation scenario.

Between the stillness time, the mobile is located at the edge of the covered area of the base station, where most of the retransmissions appear as shown in the figure 5.

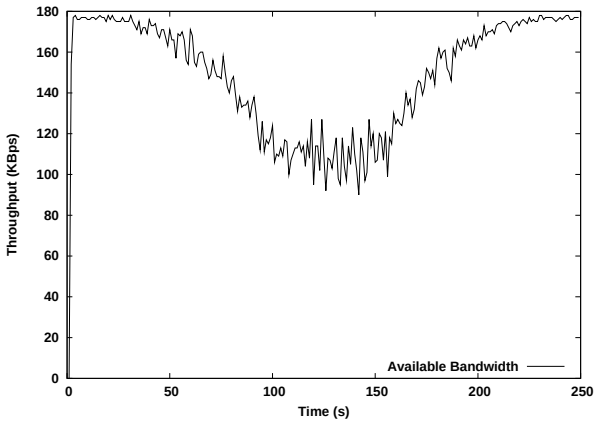

Figure 5: Available bandwidth for our scenario.

We transfer the same video twice with the same movement scenario. Only the transport protocol differs from one simulation to another. We use:

- RTP over UDP, the original video streaming transport protocol,
- RTP over DCCP/TFRC with video adaptation. The mixer adapts the video to the bandwidth.

5.0.2. Video quality comparison

To compare the videos received by the client, we use two different kinds of comparisons:

- The Peak Signal-to-Noise Ratio (PSNR) measure distortion used in compression of pictures that allow to quantify the performance of the codec by measuring the quality of the

compressed picture compared to the original one. The PSNR is expressed in decibels and a picture of good quality gives a PSNR ranging between 30 and 40 dB. This value should not be taken for an objective value of the visual quality of a picture. The PSNR value for a video is calculated frame by frame. First each video (original and received) is converted into an image set (24 frames per second). Secondly the PSNR is computed for each couple of frame.

- Structural SIMilarity [3] (SSIM) is a measure of similarity between two digital pictures, developed to compare the visual quality of a picture compressed compared to the original picture. This measurement fills the gaps of the PSNR by taking account the characteristics of the human visual system more sensitive to the changes of structure of a picture, rather than with a difference pixel with pixel. The SSIM index is a decimal value between 0 whereas 1. 0 would means zero correlation with the original picture, and 1 means the same picture.

To make the various comparisons, we use *MSU Video Quality Measurement Tool*[4] developped by MSU Graphics & Media Lab. The generated videos for all our tests are available on-line at `http://rma.pu-pm.univ-fcomte.fr/ijdmb08/`.

5.1. Results for UDP vs DCCP/TFRC

5.1.1. Throughput

Figure 6 shows the three stages of the movement. During the first $50s$ the throughputs of both DCCP/TFRC and UDP are smaller than the available bandwidth because the video mostly contais still pictures or small movements. Indeed, the bitrate of the video is less than the available bandwidth. After this time interval, the throughput better fits to the available bandwidth and the two curves of the throughputs are quite the same. This result has to be analyzed in conjonction with the packet losses. UDP has indeed no congestion protocol and must have more losses than DCCP/TFRC.

5.1.2. Packet loss

In figure 7, during the $250s$ of RTP/UDP video transmission, 2063 packets are lost. With DCCP/TFRC, this number is reduced to 180 packets only. In the UDP case, most of these losses appear when the available bandwidth is smaller than the video bitrate. This graph shows the efficiency of DCCP/TFRC to evaluate the available bandwidth and to minimize the losses according to this evaluation.

5.1.3. Video Comparison

For each simulated transport protocol, the original video and the received one are compared using the PSNR, figure 8, and SSIM, figure 9.

Figure 8 presents the number of pictures received for each PSNR quality. We can see that the DCCP/TFRC plain curve, has more pictures of good quality (superior to 35dB) than the dotted curve of UDP. This is due to the fact that TFRC better evaluates the bandwidth and that the mixer perform video adaptation. The video reads by the client on its mobile player has a better quality according to the PSNR criteria. The overall ratio of mixer active time can be estimated on figure 8 from the difference on the peak parts of the curves. This value is approximately (350-300)/350= 14% which is quite low according to the benefits in the quality of video stream.

[4] `http://www.compression.ru/video/quality_measure/video_measurement_tool_en.html`

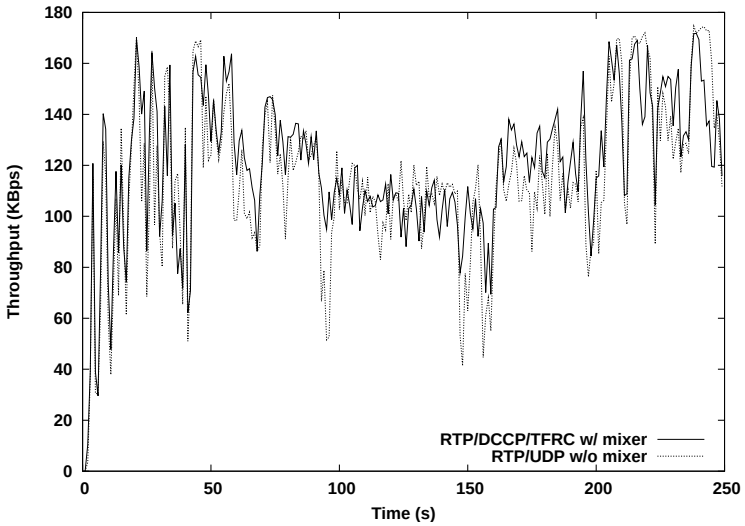

Figure 6: Throughput comparison between UDP and DCCP video streaming.

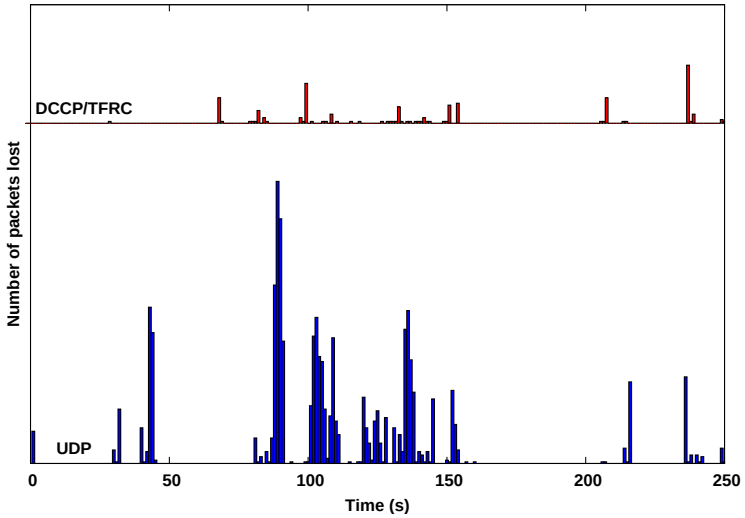

Figure 7: Packet loss comparison between UDP and DCCP/TFRC (linear scale on y axis, UDP(89)=166, TFRC(237)=32).

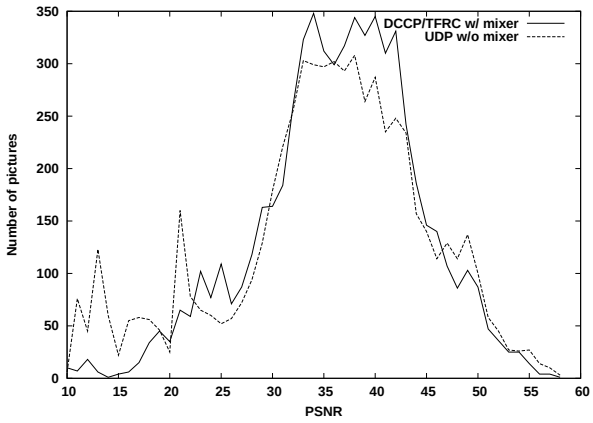

Figure 8: PSNR comparison between UDP and DCCP/TFRC.

Figure 9 presents percentage of pictures found in each quality segment (bad quality, average -, average +, good quality). We can see that the two videos have the same number of good quality pictures, but that DCCP/TFRC has much more average+ pictures than UDP (figure 10). The average SSIM and standard deviation values for UDP are 0.95 and 0.14 respectively which are worst than the average SSIM and standard deviation values or DCCP/TFRC (0.99 and 0.044). This agrees with the results of the PSNR.

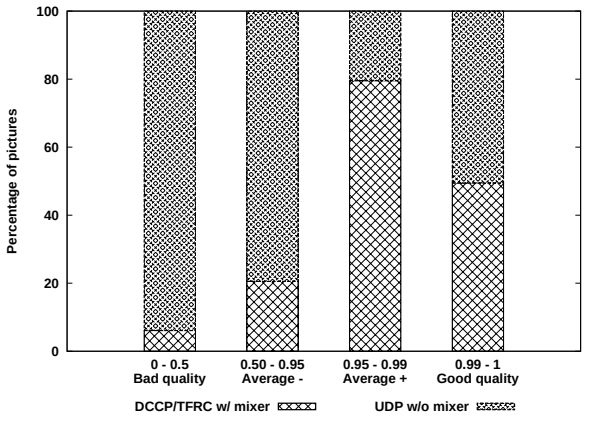

Figure 9: SSIM comparison between UDP and DCCP/TFRC.

6. Conclusion and future work

This paper proposes a complete video streaming architecture which includes a mixer combining a server and a mobile client. This paper demonstrates that this simulation model can help to optimize and to implement a dedicated congestion control protocol.

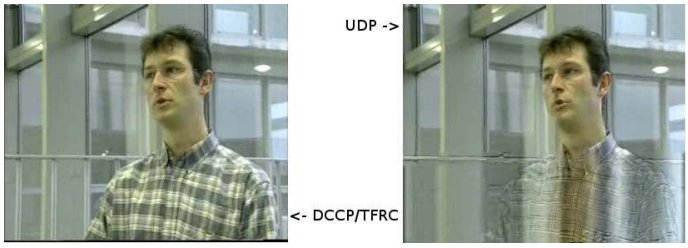

Figure 10: Screenshots of UDP and DCCP/TFRC videos streaming.

The measured improvements of our test bed compared to that of classical video streaming are significant. Two types of results have been presented, for the network (throughput and packet loss) and for the visualization of received video. Our solution gives better results than the classical RTP/UDP solution for two reasons: improvement of video quality measured with PSNR and SSIM functions and the ability to be transmitted over the whole Internet thanks to its congestion control which is TCP-friendly.

Moreover, the use of network simulator allows the realization of successive reproducible scenarios with different data to validate several parts of a development, namely that of the adaptation mixer NetMoVie. Through the use of real video, the content is actually transported, giving content to its importance during the simulated transmission.

The relevance of a mixer adaptation in a chain of transmission has been demonstrated. In the scenarios including an adaptation, video received by client are of better quality than the estimate, either mathematical or visual. The solutions we have selected and developed respect the various standards implemented.

We want to propose new solutions of control congestion combined with RTP mixer in order to adapt video streaming to the constraints of wired and wireless networks. Thanks to our simulation model, it will be faster and easier to propose new efficient strategies for delivering multimedia contents.

References

[1] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A transport protocol for real-time applications, 2001. RFC 1889.

[2] H. Schulzrinne, A. Rao, R. Lanphier, Real Time Streaming Protocol (RTSP), RFC 2326 (Proposed Standard), 1998. URL: `http://www.ietf.org/rfc/rfc2326.txt`.

[3] Z. Wang, A. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment: From error visibility to structural similarity, in: IEEE Trans. Image Processing, volume 13, IEEE Computer and Communication Societies Press, 2004, pp. 600–612.

[4] E. Kohler, M. Handley, S. Floyd, Datagram Congestion Control Protocol (DCCP), RFC 4340 (Proposed Standard), 2006. URL: `http://www.ietf.org/rfc/rfc4340.txt`.

[5] Network Simulator, Network simulator — ns-2, `http://www.isi.edu/nsnam/ns/`, 1989.

[6] J. Bourgeois, E. Mory, F. Spies, Video transmission adaptation on mobile devices, Journal of Systems Architecture 49 (2003) 475–484.

[7] S. Floyd, E. Kohler, Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control, RFC 4341 (Proposed Standard), 2006. URL: `http://www.ietf.org/rfc/rfc4341.txt`.

[8] M. Handley, S. Floyd, J. Padhye, J. Widmer, TCP Friendly Rate Control (TFRC): Protocol Specification, RFC 3448 (Proposed Standard), 2003. URL: `http://www.ietf.org/rfc/rfc3448.txt`.

[9] T. Young, High Quality Video Conferencing, Honours project, University of Waikato, Hamilton, New Zealand, 2003.

[10] M. Chen, A. Zakhor, Rate control for streaming video over wireless, in: INFOCOM, volume 2, IEEE Computer and Communication Societies Press, Hong Kong, 2004, pp. 1181–1190.
[11] Z. Lei, N. Georganas, Rate adaptation transcoding for precoded video streams, in: Proceedings of ACM Multimedia, Juan-les-Pins, France, 2002, pp. 127–136.
[12] J. Youn, M.-T. Sun, J. Xin, Video transcoder architectures for bit rate scaling of H.263 bit streams, in: Proceedings of the seventh ACM international conference on Multimedia, 1999, pp. 243–250.
[13] A. Eleftheriadis, B. Anastassiou, Constrained and general dynamic rate shaping of compressed digital video, in: International Conference on Image Processing (ICIP), 1995, pp. 396–399.
[14] Q.-F. Zhu, L. Kerofsky, M. B. Garrison, Low-delay, low-complexity rate reduction and continuous presence for multipoint videoconferencing, IEEE Transactions on Circuits and Systems for Video Technology 9 (1999) 666–676.
[15] S. Roy, B. Shen, Implementation of an algorithm for fast down-scale transcoding of compressed video on the itanium, in: Proceeding of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video, 2001, pp. 119–126.
[16] J. Xin, C.-W. L. M.-T. Sun, Digital video transcoding, in: Proceedings of the IEEE Advances in Video Coding and Delivery, vol. 93, 2005, pp. 84–97.
[17] N. Bjork, C. Christopoulos, Transcoder architecture for video coding, IEEE Trans. Consum. Electron 44 (1998) 88–98.
[18] T. Shanableh, M. Ghanbari, Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats, IEEE Transactions on Multimedia 2 (2000).
[19] J. Youn, M.-T. Sun, C.-W. Lin, Motion vector refinement for high-performance transcoding, IEEE Transactions on Multimedia 1 (1999) 30–40.
[20] M.-J. Chen, M.-C. Chu, C.-W. Pan, Efficient motion-estimation algorithm for reduced frame-rate video transcoder, IEEE Transactions on Circuits and Systems for Video Technology 12 (2002) 269 – 275.
[21] S. Jin, A. Bestavros, Gismo: a generator of internet streaming media objects and workloads, ACM SIGMETRICS Perform. Eval. Rev. 29 (2001) 2–10.
[22] C. Partridge, R. Hinden, Version 2 of the Reliable Data Protocol (RDP), RFC 1151 (Experimental), 1990. URL: `http://www.ietf.org/rfc/rfc1151.txt`.
[23] B. Klasen, Efficient content distribution in social-aware hybrid networks, Journal of Computational Science 4 (2013) 209 – 218. PEDISWESA 2011 and Sc. computing for Cog. Sciences.