IMPLEMENTATION OF ADAPTIVE STREAMING OF STORED MPEG-4 FGS VIDEO OVER TCP

Philippe de Cuetos‡, *Philippe Guillotel*†, *Keith W. Ross*‡, *Dominique Thoreau*†

Institut EURECOM‡
2229, Route des Cretes
06904 Sophia-Antipolis, France
{decuetos,ross}@eurecom.fr

THOMSON Multimedia R&D France†
Av. de Belle Fontaine, B.P. 19
35511 Cesson Sevigne Cedex, France
{guillotelp, thoreaud}@thmulti.com

ABSTRACT

This paper presents an implementation of an end-to-end application for streaming stored MPEG-4 Fine-Grained Scalable (FGS) videos over the best-effort Internet. Our current implementation runs over TCP but can also be run over TCP-friendly RTP/UDP with an error recovery mechanism. Our scheme adapts the coding rate of the streaming video to the variations of the available bandwidth for the connection, while smoothing changes in image quality between consecutive video scenes. We justify our design choices and present the results of the simulations obtained from our testbed. Our combined use of FGS-encoded video with a simple network-adaptation heuristic gives a system of low complexity, which is suitable for high performing video servers.

1. INTRODUCTION

In today's best-effort Internet, heterogeneity in Internet access and dynamic variability in network conditions have brought the need to design network-adaptive video applications. Recently, bit-plane based FGS-coding has been added to the MPEG-4 standard [3], so that it can be used in such applications. FGS-coding is a new form of video hierarchical encoding. Hierarchical, or scalable, videos are encoded into one Base Layer (BL) and one or several Enhancement Layers (EL). The server can choose the number of EL layers to stream to the client in addition to the BL, as a function of network conditions [5]. However, with FGS coding, any number of bits of the FGS-encoded EL can be suppressed at the server before transmission, and the achieved image quality is directly proportional to the number of bits decoded at the client [6].

In this paper, we study streaming of long videos (from tens of seconds to hours), which are composed of several video scenes. Our current implementation runs over TCP. TCP is still ubiquitous in the Internet, it has been proven to be stable and is available to use. Moreover, although TCP bandwidth is more varying than recent TCP-Friendly UDP protocols such as TFRC [10], bandwidth fluctuations can be accommodated by sufficient playback buffering as shown in [1]. As does Krasic et al. [9], we believe that TCP is a viable choice for quality-adaptive video streaming. Nevertheless, full-reliability of TCP is not necessary for video streaming applications that could benefit from partially-reliable schemes such as selective retransmissions. Therefore, our application has been designed to be potentially run also over any partially reliable TCP-Friendly RTP/UDP connection.

To our knowledge, our system is one of the first implementations of an Internet streaming application which combines the use of FGS-encoded video, a network-adaptive algorithm, and video scene-based segmentation, in order to smooth perceptual changes in image quality, while maintaining high bandwidth efficiency. In [4], Zhang et al. also present a complete end-to-end system that streams FGS-encoded videos, but using a specific smooth TCP-Friendly protocol (MSTFP). The system from Radha et al. [8] does not aim to smooth fluctuations in image quality between consecutive images.

This paper is organized as follows. We first describe the complete architecture of our system. Then, we present separately the MPEG-4 FGS encoder that was used to encode the video sequences, and the streaming algorithm that was implemented to adapt to network bandwidth variations in real time. Finally, we present our tests and results to show the performance of our system.

2. ARCHITECTURE

The end-to-end architecture of our system is depicted in Figure 1. At the server, BL and EL data are stored in separate files. Each data file is associated with a meta-file containing pointers to the individual Access Units (AUs), or frames, as well as their composition and decoding timestamp. Video pumps push the AUs and their associated information to the network module. The server network module encapsulates the AUs into RTP packets in order to stay compatible with implementations over RTP/UDP. (This brings some overhead, although relatively small since AUs do not need to be fragmented into several RTP packets in the case of transmission over TCP.) The RTP payload format we used is the Group Payload (GP) format defined in [2]. Both BL and EL RTP packets are multiplexed over the same TCP connection to the client. Our server streams the video data at the maximum TCP available bandwidth, denoted by $X(t)$ at time t. To reduce server complexity, we require that the server always send both layers of the same frame together.

The client extracts the AUs and their associated information from the incoming RTP packets and sends them in the corresponding playback buffers. In our implementation we used four playback buffers at the client: two for the BL and EL data

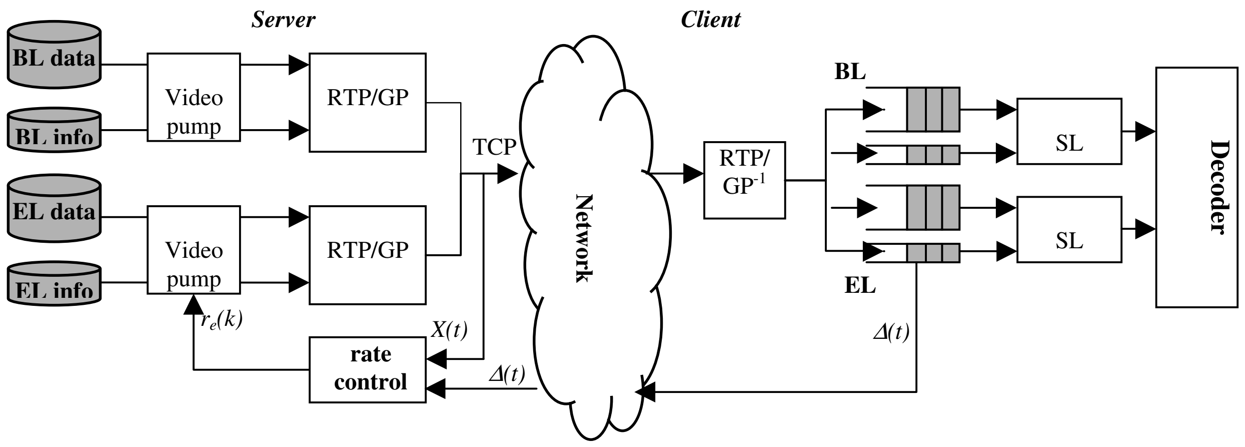

Figure 1: Complete system

and two for the BL and EL meta-information. Because the server streams both layers of the same frame together, BL and EL buffers always contain the same number of frames. We denote by $\Delta(t)$ the contents of the client buffers in seconds-worth of video data at time t. The client sends back periodically to the server the value of $\Delta(t)$. Individual AUs and their meta-information are then given to the decoder as SL packets, according to the MPEG-4 specification [7].

We assume that the stored video has been partitioned into n video sequences, where images within the same sequence are supposed to have similar visual characteristics (e.g. same video shot, same motion, or same image complexity). The BL is VBR-encoded, with coding rate $r_b(t)$, and the FGS-EL CBR-encoded with coding rate r_e. According to the FGS property, the server can cut the EL bitstream anywhere. In our system, the rate control module fixes the coding rate of EL to stream to the client for a given video sequence k, denoted by $r_e(k)$ between 0 and r_e. Then, the EL video pump cuts the EL bitstream according to this rate. The server computes the value of $r_e(k)$ as a function of the observed available bandwidth $X(t)$ and the contents of the client buffers $\Delta(t)$.

3. FGS CODEC

In bit-plane FGS-encoding the EL is obtained by encoding the DCT residue between the BL DCT coefficients and the source video DCT coefficients using a bit-plane approach [6]. DCT residues are encoded bit-plane by bit-plane, from the most significant bit to the least significant bit, and each bit-plane is encoded on a 8x8 block basis. Depending on the BL encoded quality, the residue needs more or less bits to achieve a given image quality. Figure 2 shows the evolution of the video average PSNR for the figure skating sequence when the FGS rate is increased from 0 to 1536 Kbps. We see that here is a clear tradeoff between BL bit-rate and EL quality.

FGS-scalability has two main advantages. First it is very simple to implement at the encoder as well as at the decoder side. Secondly the FGS-EL can be truncated anywhere: the longer the truncated stream, the more bit-planes are decoded. This property allows the EL bit-stream to be cut and packetized during transmission according to the varying available bit-rate for the connection. The FGS enhancement may be spatial (also called SNR scalability) or temporal (the EL increases the frame rate of the BL). In this paper, we only consider SNR scalability.

4. ADAPTIVE STREAMING ALGORITHM

A *transmission policy,* denoted by $(r_e(0), \ldots, r_e(n))$, is a set of successive EL coding rates streamed by the server for all sequences $0,\ldots,n$ of the video. In previous work [1], we derived, under complete knowledge of bandwidth evolution, an optimal transmission policy for a criterion that involves both image quality and quality variability during playback. Based on this ideal optimal policy, we developed a real-time heuristic, for CBR-encoded videos, that was shown to perform almost as well as the ideal optimal policy for a wide range of bandwidth scenarios. In this paper, we adapted our heuristic for the more general case of VBR-encoded Base Layers.

The real-time heuristic is given in Figure 3. The constant C (expressed in seconds-worth of video data) is chosen empirically and allows us to trade off video quality with playout interactivity. The content of the client playback buffers when beginning to stream video sequence number k is denoted by Δ_k. It is estimated at the server from the periodic feedback received from the client. We denote by $X_{avg}(k)$ the average available bandwidth during the streaming of the previous sequences, and by $r_b(k)$ the average BL coding rate of video sequence number k. (This is known in advance at the server since the video is pre-encoded.) In our heuristic, we suppose that the client starts the playout of video data after having received $\Delta_0 = C$ sec. of data. The server decides, for each new video sequence, the coding rate to stream for this sequence, i.e., $r_e(k)$. As shown in Figure 3, it operates according to the value of Δ_k. The heuristic tries to maintain playback buffers with a least C sec. of video, in order to accommodate short-term bandwidth fluctuations and to avoid losses of video data due to buffer starvation. Then, when buffers are sufficiently filled, the coding rate of the EL to stream follows the evolution of the available average bandwidth that remains after streaming the BL data. We include a smoothing factor α, which aims to smooth the variations of $X_{avg}(k)$, so that

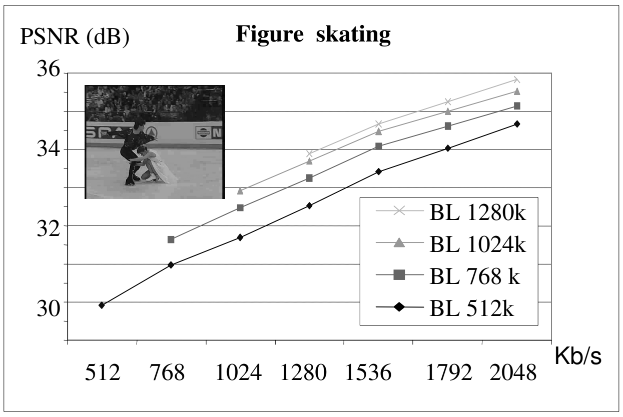

Figure 2: PSNR vs. total bit-rate for different BL rates.

the coding rate of successive video sequences varies slowly (in order to minimize variations in quality between successive video sequences). The value of *α* is chosen empirically to trade off small quality variability (small *α*) with better overall bandwidth utilization (high *α*). When $\Delta_k > 2C$ our heuristic is more aggressive with respect to the available bandwidth, in order to keep the playback buffers small. Indeed, high playback buffers make the streaming end well before the end of the rendering, which does not provide maximum video quality.

5. TESTS AND RESULTS

5.1. LAN simulations

For our experiments, we used a 4-mn video including both high and low motion scenes, and segmented into 54 video sequences of various length (in our tests, each sequence corresponds to a short scene shot). The BL was VBR-encoded, with average encoding rate of 384 Kbps. The EL was coded at 616 Kbps. The server and client were located at each end of a dedicated LAN, and a Linux router was used to limit the end-to-end available bandwidth. Cross traffic was generated from the server to the client in order to make the available bandwidth for the streaming application vary with time, as shown in Figure 4. Figure 4 also shows the choices made by the server for the EL coding rate using our real-time heuristic with *C=5* sec. and *α=0.5*. We see that the variations of the EL coding rate streamed to the client roughly follow the variations of the available bandwidth. (Note that the coding rate of the VBR-encoded BL should be added to the EL coding rate to make the total coding rate of the video.)

Figure 5 depicts the variations of the content of the client playback buffers, i.e., *Δ(t)*. We see that it never empties, i.e. no video data was lost. Finally, Figure 6 shows the PSNR values after decoding video frames 525 to 1029 with our shot-based video segmentation (segm), together with the PSNR obtained when the video segments are of arbitrary length of 200 frames (no segm). We clearly see that without shot-based segmentation the image quality can be abruptly changed within a same scene shot (e.g. around frame number 1000), which may degrade severly the perceived sequence quality.

If $\Delta_k \le C$,

$$r_e(k) = 0$$

Else if $C < \Delta_k \le 2C$,

$$r_e(k) = \alpha \cdot (X_{avg}(k) - r_b(k)) + (1-\alpha) \cdot r_e(k-1)$$

Else if $\Delta k > 2C$,

$$r_e(k) = \alpha \cdot r_e \cdot \frac{\Delta_k}{2C} + (1-\alpha) \cdot r_e(k-1)$$

Figure 3: Real-time heuristic

5.2. Performance of the heuristic

We have run the same experiment with different parameters for our real-time adaptation heuristic. We denote by Δ_{end} the content of the client playback buffers when the server has finished to stream all AUs. Essentially, lower values of Δ_{end} result in higher total bit consumption, or bandwidth efficiency, and then potentially higher overall video quality, since this minimizes the bandwidth left unused at the end of the streaming. We define a metric that can account for the variability in quality between successive video sequences:

$$V = \sum_{k=1}^{n} (PSNR(k) - PSNR(k-1))^2 ,$$

where *PSNR(k)* is the average PSNR of video sequence number *k*. This metric penalizes high differences in quality, which are more visually disturbing than small ones.

Table 1 shows the performance of our real-time heuristic in terms of Δ_{end} and *V*, when *C* is changed. As we can see, a low value of *C* causes higher variations in quality (indeed when *C=1* sec. we observe the loss of several video frames because of playback buffer starvation), whereas a high value for *C* can cause low bandwidth efficiency, since the adaptation is not enough reactive with respect to the variations of the available bandwidth (this also makes the user wait for a longer time before he can start watching the video).

C (sec)	Δ_{end} (sec)	***V***
1	0	696
5	8.8	375
15	45.7	370

Table 1 - variations of *C* for *α = 0.5*

In Table 2 we varied the smoothing parameter α. As we can see, a high value of *α* gives better bandwidth efficiency, at the price of a relatively higher variability (although variability is almost the same here when *α=0.5* or *1.0*).

α (sec)	Δ_{end} (sec)	***V***
0.1	11.5	368
0.5	13.5	375
1.0	8.95	373

Table 2 - variations of *α* for *C = 5* sec.

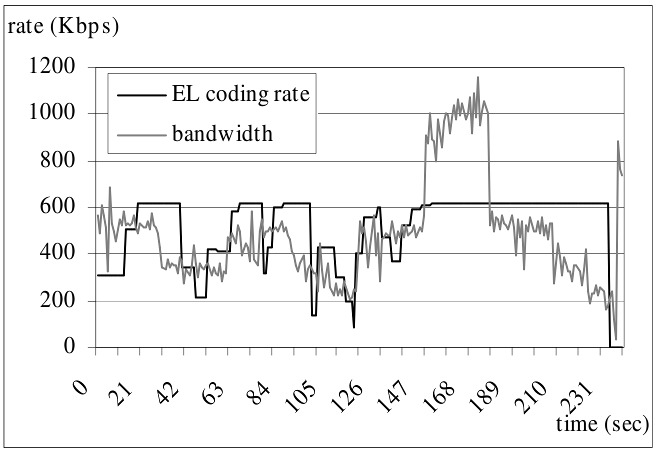

Figure 4: rate adaptation

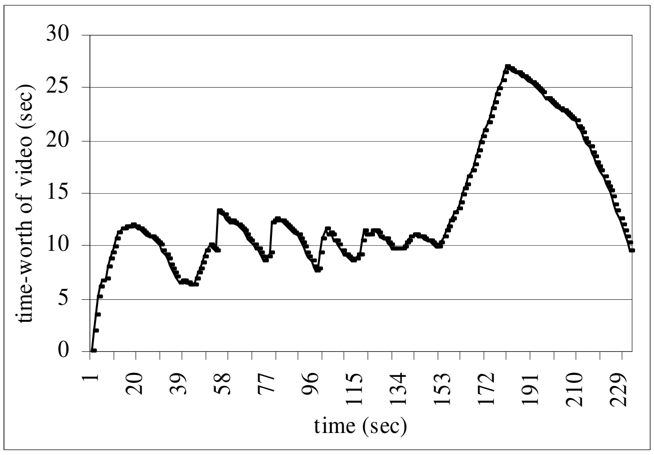

Figure 5: playback buffers content

6. FUTURE RESEARCH

We have presented an implementation of an end-to-end streaming application which uses FGS MPEG-4 videos to adapt in real-time and with low complexity to varying network conditions. Tests in realistic network situations have shown that our system gives good visual performance despite low efficiency of current FGS-encoding schemes. In addition to working on more efficient FGS encoders, we plan to improve the rate adaptation heuristic. In particular, by considering the different rate-distortion characteristics of the successive video sequences, we intend to optimize an actual measure of video quality, instead of bandwidth utilization. Finally, our scheme would also highly benefit from extensive subjective tests on the visual impact of quality variability.

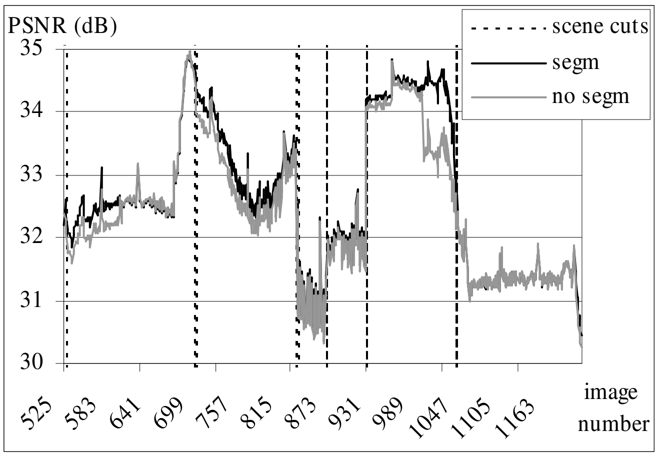

Figure 6: PSNR curves

7. REFERENCES

[1] P. de Cuetos, K. W. Ross, "Adaptive Rate Control for Streaming Stored Fine-Grained Scalable Video", *in Proc. of Nossdav,* Miami, Florida, May 2002.

[2] C. Guillemot et al., "RTP Payload for MPEG-4 with Scalable and Flexible Error Resilience", *Internet draft*

[3] ISO/IEC JTC1/SC29/WG11 Information Technology – Generic Coding of Audio-Visual Objects: Visual ISO/IEC 14496-2 / Amd X, December 1999.

[4] Q. Zhang, W. Zhu and Y-Q. Zhang, "Resource Allocation for Multimedia Streaming over the Internet", *IEEE Trans. on Multimedia*, September 2001.

[5] R. Rejaie, D. Estrin, and M. Handley, "Quality Adaptation for Congestion Controlled Video Playback over the Internet", *in Proc. of ACM SIGCOMM,* Cambridge, September 1999.

[6] W. Li, "Overview of Fine Granularity Scalability in MPEG-4 Video Standard", *IEEE Trans. on Circuits and System for Video Technology*, March 2001.

[7] ISO/IEC JTC1/SC29/WG11 Information Technology – Generic Coding of Audio-Visual Objects: Systems ISO/IEC 14496-1, December 2001

[8] H. Radha, Y. Chen, K. Parthasarathy and R. Cohen, "Scalable Internet Video Using MPEG-4", *Image Communications*, 15, 1999.

[9] C. Krasic, K. Li and J. Walpole, "The Case for Streaming Multimedia with TCP", *in Proc. of IDMS*, Lancaster, UK, September 2001.

[10] S. Floyd, M. Handley, J. Padhye and J. Widmer, "Equation-based Congestion Control for Unicast Applications", *in Proc. of ACM SIGCOMM*, Stockholm, Sweden, August 2001