Interactive MPEG Video Streaming over IP-Networks: A Performance Report

Marco Furini  
Department of Computer Science  
University of Piemonte Orientale  
Alessandria, Italy  
email: marco.furini@mfn.unipmn.it

Marco Roccetti  
Department of Computer Science  
University of Bologna  
Bologna, Italy  
email: roccetti@cs.unibo.it

ABSTRACT
The popularity of interactive video streaming applications have pushed researchers to propose mechanisms for supporting these applications over the Internet. Several studies showed that interactive operations are well supported if the end-to-end delay, experienced by the application traffic, is kept lower than a pre-defined threshold. Recently, it has been proposed a new approach that acts on the video QoS (by dropping frame) in order to provide interactive features. The mechanism has been tested with Motion JPEG videos. The contribution of this paper is to evaluate the mechanism with MPEG videos, in order to investigate the behavior of the mechanism when the video is encoded with one of the more popular inter-frame techniques. Further, since the mechanism acts on the video QoS, we also present a QoS evaluation of the perceived video play out quality.

KEY WORDS
Multimedia Communications, Quality of Service Issue, Multimedia over IP-Networks, Frame Dropping

1 Introduction
Quality of Service (QoS) applications are becoming an integral part of our communication environment and they are more and more popular also in the Internet environment. Unfortunately, despite their popularity, they reach a QoS that is far from what desired. QoS difficulties are mainly due to the traffic produced by these applications that is time-dependent and may be very bandwidth consuming. The great bandwidth requirements are highlighted by the video streaming applications; in fact, even if the video is compressed (MPEG [1], Motion JPEG [2], H.261 [3]) the resulting stream can require high network capacity compared to the (usually) available in the Internet. In addition, the transmission of this traffic has, at least, two time-constraints: minimal communication delay and network jitter [4]. These time-constraints are very critical to provide in best-effort networks, like the Internet, which cannot guarantee low communication delays and/or low jitter.

A sub-set of these QoS applications, called interactive, enables natural interactions (i.e., more life-like as possible) among end-users, and are more difficult to support than normal QoS applications. Several studies [5, 6, 7] showed that interactive applications are well supported if the end-to-end delay is kept within a threshold along the application lifetime (the end-to-end delay must be not noticeable to the end users). Hence, the threshold represents the limit below which the interactions are well supported; if we denote this threshold with NIT (Natural Interaction Threshold), the end-to-end delays above this bound are noticeable and, hence, interactions are not well supported. This threshold is not fixed [5], but depends on the application and on the level of interactivity requested by the end-users (i.e., the more interactive operations are involved, the lower the threshold value has to be).

Among the components that affect the end-to-end delay [8], the network delay is the most variable. It is essentially composed by: propagation, or transmission, and queuing delay. The propagation delay is known as it depends on the network capacity and on the size of the data to transmit. Conversely, the queuing delay is unknown a-priori and it is also very variable, as data travel from source to destination along a path that is usually shared among traffic generated by other applications. Hence, it may happen that a network resource along the path is busy, causing the data to be delayed until the resource is available.

Further, the network delay varies from time to time, and its variability (the network jitter) may cause QoS problems to the receiver. For instance, if we consider a video streaming application, the network jitter may cause video play out interruptions, as video frames may be not available for play out when needed.

In literature there were studies aimed at ameliorating the network jitter: buffering or smoothing techniques are some of these studies [9, 10]. Although these techniques are very effective, they cannot be used to support interactive QoS applications, as they introduce a start-up delay that increases the overall end-to-end delay, pushing it above the NIT. Since for video application the NIT is usually less than 500ms, it is not possible to use mechanisms, as buffering techniques, that usually introduce a start-up delay of few seconds to ameliorate the network jitter [11].

In [12] a new approach to support interactive video applications over the Internet is presented. The mechanism does not use any start-up delay, and it acts on the video
QoS in order to mask the network jitter. Briefly, in [12] it is pointed out that the source of the video stream can be located in two possible sides: i) at the user side (either the video is generated with a webcam, or it is locally stored) or ii) somewhere in the network. Needless to say, the ideal scenario to perform interactive operations is when the video is locally available (the network is not involved) and hence the end-to-end delay is not noticeable. Conversely, if the network is involved, an excessive delay may be introduced, causing the overall end-to-end delay to be noticeable. In fact, if the network delays these data with a value greater than NIT, the interactive operations are compromised. Conversely, if the network is not involved, the play out can be done without considering the network delay. This last scenario is referred to as the ideal scenario for providing interactions and it is in contrast to the actual scenario. In essence, the goal of the mechanism is to support the video play out in the actual scenario, while attempting to simulate the ideal scenario. This is done by acting on the video QoS (i.e., by dropping frames) at the sender side. The mechanism has been evaluated through several simulations and results obtained show that it is well suited for supporting interactive QoS applications over the IP-Networks [12]. All the simulations have been performed using videos encoded with intra-frame techniques (e.g., Motion JPEG). This means that each frame is independently encoded and can be dropped without particular problems. Conversely, if the video is encoded with inter-frame mechanisms (e.g., MPEG), the discard of a single frame can result in the impossibility of decoding several other frames, as video frames are not independently encoded.

The contribution of this paper is to test the mechanism using several MPEG video traces and to evaluate the QoS (through a cost function) of the resulting streams. In fact, since a discarded MPEG-frame can result in the impossibility of decoding several other frames, and since there may not be much correlation between the number of dropped frames and the perceived playout QoS, there is a need to evaluate the QoS of the resulting stream. We propose several and different dropping algorithms in order to investigate and to find out the algorithm that produces good results while slightly affecting the QoS of the resulting stream.

The remainder of this paper is organized as follows. In section 2 we present an overview of the mechanism presented in [12]. In section 3 we show results obtained from evaluating the mechanism while transmitting MPEG traces. We also propose several dropping algorithms and we present a QoS evaluation of the resulting video streams. Conclusions are drawn in section 4.

2 Mechanism Overview

In this section, we present a brief overview of the mechanism proposed in [12]. Readers can refer to [12] for details.

As already pointed out, if the goal is to support interactive applications, the ideal play out does not involve the network; unfortunately, in the actual play out the network is involved and this can raise QoS difficulties.

The mechanism aims at maintaining the actual play out very close to the ideal play out, by using a timestamp mechanism to measure the time difference between the actual and the ideal play out. If this difference is not noticeable to the users, the interactive application is well supported. By supposing the clocks at the sender and at the receiver side synchronized, the time difference is measured through a metric, called VTD (Video Time Difference), which is periodically measured at the receiver side. In essence, the goal of the mechanism is to maintain the VTD within the NIT value. If the network causes the VTD to go above the acceptable threshold, the mechanism acts on the video QoS (i.e., by dropping some video frames) and reports the VTD within the NIT.

In the following $T_s(i)$ and $T_R(i)$ denotes the ideal and the actual play out time of the frame $i$, respectively.

2.1 Transmission and Play Out Algorithms

2.1.1 Transmission Algorithm

Each transmitted frame is marked with a timestamp, which represents the ideal play out time of the frame, according to the following rules:

1. The timestamp of the first video frame represents the time at which the video frame is transmitted. If we denote this time with $t$, it follows that $T_s(1) = t$.

2. A frame $i$ ($i > 1$) is marked with $T_s(i) = T_s(i - 1) + \alpha$, where $\alpha = 1/\delta$ and $\delta$ is the number of frame that must be displayed every second.

2.1.2 Video Play Out algorithm

The receiver retrieves video frames from the network, temporarily stores them into its local buffer and then plays out these video frames according to the following rules:

1. Video play out starts when the first video frame arrives at the receiver side, say at time $t'$; Hence, $T_R(1) = t$;

2. The receiver plays out the frames at fixed period (i.e., one frame every $\alpha = 1/\delta$ time units);

3. In the buffer, the frame with the lowest timestamp is selected for play out;
4. Once selected, a frame $i$ is removed from the buffer and is played out at time $T_R(i) = T_R(prev(i)) + \alpha$ only if: a) $T_R(i) \geq T_S(i)$ and b) $T_S(i) > T_S(prev(i))$, where $prev(i)$ is the most recently frame that has been played out. If conditions a) and b) are not met, then frame $i$ is discarded and a new frame selection must be done (by applying rule 3);

Based on the previous rules, the algorithm plays out a frame $i$ at time $T_R(i) = T_R(prev(i)) + \alpha$, where $prev(i)$ indicates the frame played out just before frame $i$.

### 2.2 Video play out problems caused by the network jitter

The algorithms described in the previous section may arise possible problems in the Internet, as shown in Fig. 1, where a sender, at time $t$, starts transmitting video frames every $\alpha$ time units. At time $t'$, the receiver plays out frame 1. Frame 2 is supposed to be played out at time $t' + \alpha$, but due to network problems, it is delivered later than expected. In the example, video play out is freezeed up to time $t' + 3\alpha$, when it is resumed playing out frame 2.

In this case, the network jitter compromised the continuity of the video play out and, even though all the successive frames are “in-time”, their play out is delayed by the network problems experienced while transmitting frame 2.

The effects of the network jitter on the VTD are highlighted in Fig. 2, where a hypothetical NIT value is also depicted in order to compare the VTD with the NIT.

Frame 2 is played out, later than expected, at time $t' + 3\alpha$. Hence $VTD(2) = t' - t + 2\alpha$. If $VTD(2) > NIT$ then $VTD(j) \geq NIT$, for each $j \geq 2$.

This situation poses serious problems if the supported application has interactive features, as all the frames, but the first, have a VTD above the NIT. For this reason, the VTD must be reported within the acceptable NIT.

In [12] it is shown that the VTD can be reduced of $\rho$ time units, by dropping a number of frames, say $k$, that corresponds to $\rho$ time units (i.e., $k \cdot \alpha = \rho$), where $\alpha = 1/\delta$ and $\delta$ is the number of frames played every second.

Since the amount of time that exceeds the NIT is known ($VTD - NIT$), the VTD can be reported within the NIT, by discarding a number of frames that corresponds to the time quantity $VTD - NIT$.

The mechanism works as follows. When the receiver finds out that the VTD is above the NIT (for instance when playing out frame 2), it sends to the sender the value $\rho \equiv VTD(2) - NIT$ (Fig. 3). The sender uses this information to compute the number of frames ($k$) that has to be discarded in order to report the VTD within the NIT. In this case, it is necessary to drop 2 frames; the sender discards frame 7 and frame 8 and, just after frame 6, transmits frame 9, frame 10 and so on.

Fig. 4 shows the effects of the mechanism on the VTD. The benefits start when playing out frame 9: if the mechanism is not used, frame 9 is played out with VTD(9) greater than NIT (Fig. 2), but with the mechanism, VTD(9) is lower than NIT (Fig. 4). Moreover, using the mechanism, all the frames transmitted after frame 9 are within the NIT.

The mechanism has been evaluated with several simulations, and results presented in [12] show that it is effective in supporting interactive QoS applications.

### 3 MPEG video transmission

The mechanism proposed in [12] has been designed to transmit video streams encoded with intra-frame techniques (e.g., Motion JPEG). In these streams all the frames are independently encoded/decoded and hence the server can drop any frame without causing problems to other frames. Conversely, in video streams encoded with inter-frame techniques, the frames are not independently encoded/decoded. This means that the discard of a single frame can cause a domino effect on several other frames. Since inter-frame techniques, like MPEG, are more and more popular, it is important to evaluate the mechanism with videos that are not independently encoded.

The first contribution of this paper is to analyze the mechanism while transmitting MPEG video streams.

MPEG is an inter-frame encoding mechanism that yields a smaller average frame size than the Motion JPEG encoding. The difference is that, in MPEG, the frames don’t have the same importance, as some frames depend on other frames. We use MPEG videos organized in Group of Picture (GOP) with a size of 12 frames. The frames may have different importance and are represented by three
type of frames: $I$, $P$, and $B$. Each GOP is composed as: $IB_1BP_1BP_2BP_3BP_4BP_5BP_6BP_7BP_8BP_9BP_{10}$. To decode a $B$ frame, both the previous and future $I$ or $P$ frames are needed. To decode a $P$ frame, the previous $P$ or $I$ frame is needed. Only $I$ frames can be decoded without using other frames.

These dependency rules have been considered while testing the mechanism and, based on them we propose the following algorithms in order to discard frames at the server side when the client asks to drop frames. 

**Drop any frame (DAF):** Frames are dropped without any consideration about the frame type; 

**Drop I frame (DIF):** Only $I$ frames are considered (14 frames cannot be played out for each discarded $I$ frame); 

**Drop P frame (DP1F):** Only $P_1$ frames are considered (11 frames cannot be played out for each discarded $P_1$ frame); 

**Drop P frame (DP2F):** Only $P_2$ frames are considered (8 frames cannot be played out for each discarded $P_2$ frame); 

**Drop P frame (DP3F):** Only $P_3$ frames are discarded (5 frames cannot be played out for each discarded $P_3$ frame); 

**Drop B frame (DBF):** Only $B$ frames are considered.

The number of dropped frames cannot be considered a good measure of the affected QoS, as there may not be much correlation between dropped frames and perceptual playout quality [13]. One possible approach to accounting for the perceptual playout quality is to use a cost function to measure the perceived video quality [14, 15]. There are many ways to define a cost function, but its definition goes beyond the scope of this paper. For this reason we focus on a cost function introduced in [14, 15], which is used to penalize frame dropping algorithms that drop neighboring frames. Briefly, this cost function takes two aspects into consideration: the length of a sequence of consecutive discarded frames and the distance between two adjacent, but non-consecutive, discarded frames. It assigns a cost $c_j$ to each discarded frame $j$, depending on whether it belongs to a sequence of consecutive discarded frames or not. If frame $j$ belongs to a sequence of consecutive discarded frames, the cost is $l_j$ if the frame $j$ is the $l_j$th consecutively discarded frame in the sequence. Otherwise the cost is given by $1 + 1/\sqrt{d_j}$, where $d_j$ represents the distance from the previous discarded frame. More details about this cost function can be found in [14, 15].

The second contribution of the paper is the evaluation of the video stream QoS, by using a cost function.

In the following, we present results obtained from analyzing the mechanism over a LAN and over the Internet, performed using video delay traces obtained transmitting a set of MPEG video traces (each of 20 minutes long, 320x160 pixels, 12 frames GOP and 24 frames per second). A simulator that uses the collected delay traces to test the mechanism has been developed.

### 3.1 Results: LAN Environment

The first set of experiments has been done over a 100Mb/s Ethernet network.

In Fig.5 we present the percentage of frames with a VTD above the NIT. Since the NIT is application dependent, to cover different situations we vary the NIT value from 50 to 150 ms. As shown, with a NIT of 50 ms, the transmission of the video stream without using the mechanism causes more than 25% of the frames to go above the acceptable limit. This percentage decreases while increasing the NIT value. For instance, with NIT values equal to 70-90 ms, the percentage drops to 2% and reaches almost zero percent with a NIT value of 150 ms.

Conversely, if we use the mechanism and the discarding algorithms presented in section 3, the percentage is kept very close to zero. In this case, all the algorithms produce almost the same results (Fig.5).

To evaluate the QoS of the resulting streams, we compute the costs and we present them in Fig. 6. Note that the cost is presented normalized with respect to the cost computed with the original stream transmission. It is possible to note that there are small differences among the dropping frame algorithms, and their costs are similar to the original one. This is due to the small number of discarding frames. However, it is possible to note that DIF performs worse than the other algorithms, while DBF is the algorithm that produces the lowest additional cost (around 0.4% more than the original cost).

The reason of this great benefit obtained by discarding very few frames is highlighted by the example described in section 2.2. In fact, in Fig. 2 the delay experienced by frame 2 affects the VTD of all the successive frames. In Fig. 3 and 4 we showed that by discarding only 2 frames, only few frames have a VTD above the NIT, while all of

![Figure 5. Transmission of The Simpsons over a LAN: percentage of frames with a VTD above the NIT.](image)

![Figure 6. Transmission of The Simpsons over a LAN: cost of the dropping algorithms.](image)
the other frames are within the NIT limit.

Similar results have been obtained with MTV and a NEWS clip. Readers can refer to [16] for further details.

### 3.2 Results: The Internet Environment

To test our mechanism over the Internet, we evaluated it in two more scenarios: one from Bologna to Trieste (9 hops) and the other from Bologna to Alessandria (8 hops).

Fig. 7 shows results obtained from transmitting a NEWS clip from Bologna to Trieste (9 hops). With a NIT of 70 ms, more than 70% of the frames have a VTD above the NIT. For a NIT value of 110 ms, the percentage drops to less than 10%, and reaches almost zero for a NIT value around 150 ms. Conversely, if the mechanism is used, the percentage is kept very close to zero and the additional cost ranges between 1.7% and 2.3%. Also in this case, DIF performs worse than the other algorithms, and DBF has the lowest costs with all the tested NIT (see [16] for details).

Similar results have been obtained from transmitting MTV (readers can refer to [16] for details).

In Fig. 8 we present results obtained from transmitting a video trace of MTV from Bologna to Alessandria (8 hops). In this case, the network is much more slower than in the previous experiments. With a NIT value of 100 ms, more than 90% of the frames have a VTD above NIT. Conversely, the dropping algorithms allow the mechanism to keep the percentage close to zero. In this case the dropping algorithms introduce a cost that ranges between 2.8% and 4%. For NIT values greater than 250 ms, DIF performs worse and DBF performs better than the other algorithms. For NIT values lower than 250, DAF is the one that produces the highest cost, while it is difficult to point out an algorithm that performs better than the others (Fig. 9).

In Fig. 10 we present results obtained from transmitting a video trace of a NEWS clip, from Bologna to Alessandria (8 hops). With a NIT value of 100 ms, 70% of the frames have a VTD above the NIT. Here, it is interesting to note that, due to the network problems, the mechanism is not able to maintain the percentage close to zero when the NIT is 100 ms, but 9% of the frames goes above the NIT limit. However, the benefits are still considerable with respect to the 70% of frames that goes above the NIT if the mechanism is not used.

In Fig. 11 is possible to note that for a NIT of 100 ms, the cost increases up to 37% with respect to the original
cost. This highlights the great network problems experienced while transmitting the video stream. The cost decreases to 2-3% for NIT values between 150 and 250 ms, while, for greater values, the additional cost is maintained around 12%. For a NIT of 600 ms, the mechanism discards a very few number of frames, and the cost is very close to the one produced by the original stream.

Results obtained from transmitting a video trace of The Simpsons, from Bologna to Alessandria (8 hops) are reported in [16]. With a NIT value of 100 ms, more than 80% of the frames go above the acceptable limit and for NIT values greater than 500 ms the percentage drops to less than 10%. The benefits of the mechanism are remarkable, as it allows the percentage to stay near the zero percent and the additional cost is kept within 1.3%.

Despite the network problems, all the performed experiments confirm the benefits of the mechanism, as the VTD is maintained within the acceptable limit. We also notice that there is no substantial difference among the dropping algorithms. However, if we consider the computed cost DIF is the algorithm that produces the higher cost, while DBF is the best among the presented algorithms. Since these algorithms are both effective in maintaining the VTD within the NIT, the DBF algorithm is worth using as it is also very simple to use.

4 Conclusions

In this paper we evaluated a recently proposed mechanism for supporting interactive video streaming applications over IP-Networks [12]. The evaluation has been done using MPEG video traces and several experiments have been conducted. Results showed that the benefits introduced by the mechanism are remarkable also when transmitting MPEG video streams.

Another contribution of the paper is the evaluation of the dropping mechanism. In fact, there may not be much correlation between dropped frames and perceptual quality of playout. For this reason we proposed several dropping algorithms and we evaluated them using a cost function that measures the perceived video quality.

In conclusion, the mechanism is well suited for supporting interactive MPEG video applications over the Internet, and its benefits can ameliorate the system reaction to the end-user commands (pause, fast forward, rewind).

References


