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

Marco Furini[1] and Marco Roccetti[2]

[1]Computer Science Department
Piemonte Orientale University
Spalto Marengo 33
15100 Alessandria Italy
email: marco.furini@mfn.unipmn.it

[2]Computer Science Department
Bologna University
Mura Anteo Zamboni 7
40127 Bologna Italy
email: roccetti@cs.unibo.it

TR-INF-2002-09-04-UNIPMN

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, and application dependent, threshold. Recently, it has been proposed a new approach that acts on the video QoS (by dropping frame) in order to provide interactive features to the supported applications. The mechanism has been designed for transmitting videos encoded with intra-frame technique (Motion JPEG). The contribution of this paper is to evaluate the mechanism with MPEG videos. This evaluation is important in order to better understand the behavior of the mechanism, as inter-frame techniques, like MPEG, are more and more used and it is more difficult to drop frames in MPEG streams. 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. In fact, these interactive QoS applications are well supported if the end-to-end delay is not noticeable to the end users. The importance of this constraint is highlighted by several studies [5, 6, 7] that showed how interactive applications are well supported if the end-to-end delay is kept within a threshold along the application lifetime. 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 that go above this bound are noticeable and, hence, interactions are not well supported. This threshold is not fixed [5], but depends on the characteristics of 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).

The end-to-end delay, as pointed out by *Baldi* and *Ofek* [8], is composed by different components: the *processing* delay (the time used to compress/decompress video frames), the *network* delay (the time needed to move data from one end-host to the other end-host) and the *synchronization* delay at the receiver side (the delay introduced to cancel the network jitter).

Among these components the network delay is the most variable. It is essentially composed by two sub-components: *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, 13] a new approach to support interactive video applications over the In-

ternet 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, 13] 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. In this scenario, 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 and, sometimes, annoying. For example, Fig. 1 shows a possible scenario for a video application. A video server transmits a video stream into the network. The stream is delivered to the receiver where it is played out. In this scenario, the network delays the transmission of the video stream as well as the interactive requests of the user at the receiver side. Needless to say, if the network delays these data with a value greater than NIT, the interactive operations are compromised. Conversely, if the user is directly connected to the video server, 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, 13]. All the simulations have been performed using video traces 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 not affecting too much the QoS of the resulting stream.

The remainder of this paper is organized as follows. In section 2 we provide an overview of the mechanism presented in [12, 13]. In section 3 we present results obtained from evaluating our mechanism while transmitting MPEG traces and we propose several dropping algorithms. Conclusions are drawn in section 4.

2 Mechanism Overview

In this section, we present a brief overview of the mechanism proposed to support interactive video applications over IP-Networks [12, 13]. This overview allows the

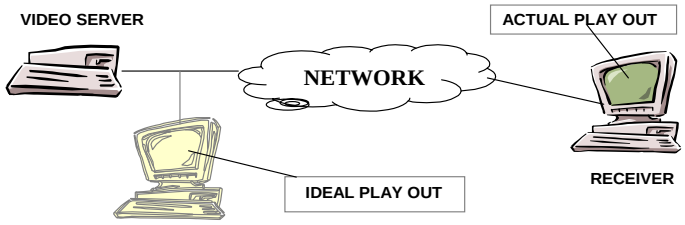

Figure 1: Network scenario while transmitting a video stream between a sender and a receiver. The ideal video play out and the actual video play out are highlighted.

readers to better understand the experimental results that we present in the next section. For the sake of conciseness we don't present a detailed overview, but we refer the readers to [12, 13] for further details.

As pointed out, if the goal is to support interactive applications, the ideal play out happens when the network is not involved; 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 and it uses 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 end-to-end delay is within the NIT), 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 order to better understand the mechanism, we denote with $T_S(i)$ the ideal play out time of the frame i and with $T_R(i)$ actual play out time of the frame i at the receiver side. In the following we present the transmission and playout algorithms and possible problems that may arise in the Internet environment.

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 the first video frame is marked with $T_S(1) = t$.

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

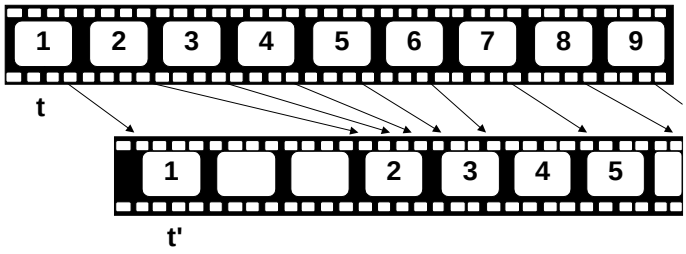

Figure 2: Video stream transmission.

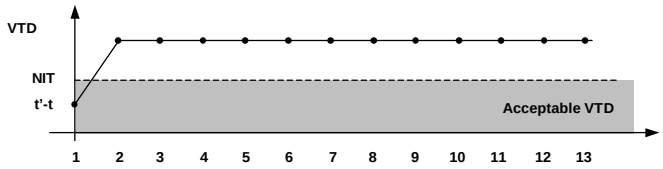

Figure 3: VTD while playing out the video stream.

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. Among the frames present in the local buffer, say k frames, it is selected (for play out) the frame with the lowest timestamp;
4. Once selected, a frame i is removed from the buffer and is played out at time $T_R(i) = T_R(i-1) + \alpha$ 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);

In other words, the last rule says that, if a selected frame i has a timestamp lower than the timestamp of the most recently frame that has been played out (i.e., $T_S(i) < T_S(prev(i))$) then frame i is discarded and a new frame selection must be done. This is done to avoid the play out of a frame i that has been transmitted before the transmission of the frame $prev(i)$, but, due to network problems, arrives later than the play out time of the $prev(i)$ frame.

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.

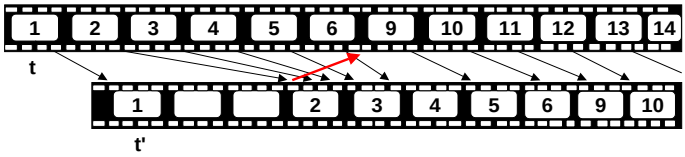

Figure 4: Video stream transmission using the mechanism.

2.2 Video play out problems caused by the network jitter

The algorithms described in the previous section are effective if the underlying network provides guarantees such as low communication delay and jitter. Conversely, possible problems may arise in the Internet: in Fig. 2 a sender, at time t, starts transmitting video frames every α 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, frame 2 is delivered later than expected. For example, if frame 2 arrives between $t' + 2\alpha$ and $t' + 3\alpha$, at time $t' + \alpha$, as well as at time $t' + 2\alpha$, the receiver has no frame to play out. Hence, the video play out is freezed 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 the delay experienced by frame 2 affects the play out time of all the successive frames. In fact, even though all the successive frames are delivered "in-time", their play out is delayed by the network problems experienced while transmitting frame 2.

This situation causes problems if interactive operations are allowed, as the end-to-end delay must stay within the NIT value.

The end-to-end delay experienced by a frame is computed as the time difference between the actual playout time (T_R) and the ideal playout time (T_S). This time difference is called Video Time Difference (VTD). If we consider a frame i, its VTD is denoted with VTD(i), and it is equal to $VTD(i) = T_R(i) - T_S(i)$.

The effects of the network jitter on the VTD are highlighted in Fig 3, where it is depicted the VTD measured for each played frame (with respect to the scenario described in Fig. 2). A hypothetical NIT value is also depicted in order to compare the VTD with the NIT.

Since we supposed that $T_S(1) = t$ and $T_R(1) = t'$, it follows that $VTD(1) = t' - t$. Frame 2 arrives later than expected and it is played out at time $t' + 3\alpha$. Hence $VTD(2) = t' - t + 2\alpha$. If $VTD(2) > NIT$ then the VTD of the successive frames is affected by the network problem experienced while transmitting frame 2. In fact, $VTD(j) \geq NIT$, for each $j \geq 2$.

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

In [12, 13] it is shown that the VTD can be reduced of ρ time units, by dropping a number of frames, say k, that corresponds to ρ time units (i.e. $k \cdot \alpha = \rho$), where $\alpha = 1/\delta$ and δ denotes the number of frames that must be 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. In Fig. 4, we consider again the example depicted in Fig. 2, but now when the receiver finds out that the VTD goes above the NIT (for instance when playing out frame 2), it sends to the sender the value $\rho \equiv VTD(2) - NIT$. When this message arrives at the sender, it is used to compute the number of frames (k) that has to be discarded in order to report the VTD within the NIT. Let us suppose that it is necessary to drop 2 frames; the sender discards (i.e., it does not transmit), frame 7 and frame 8. This means that, just after frame 6, the sender transmits frame 9, frame 10 and so on.

Fig. 5 shows the effects of the mechanism on the VTD. The benefits start when playing out frame 9. In fact, if the mechanism is not used (Fig. 3), frame 9 is played out with VTD(9) greater than NIT, but if the mechanism is used (Fig. 5), VTD(9) is lower than NIT. Moreover, using the mechanism, all the frames transmitted after frame 9 are within the NIT value.

This means that when the mechanism is not used, the considered application does not provide sufficient QoS to the interactive applications, as the VTD is often above the NIT. Conversely, the mechanism is able to report the VTD within the NIT, dropping only some video frames.

The mechanism has been tested with several simulations, and results presented in [12, 13] show that it is effective in supporting interactive QoS applications over the Internet.

3 MPEG video transmission

The mechanism proposed in [12, 13] 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 cam 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, as MPEG, are more and more popular, it is important to evaluate the mechanism with video traces where frames are not independently encoded.

The first contribution of this paper is to analyze the mechanism while transmitting video streams encoded with inter-frame technique. In this section, we present results obtained from several simulations that have been done to test the mechanism when the transmitted videos are encoded with MPEG technique.

MPEG is an inter-frame dependency 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: $IB_1B_2P_1B_3B_4P_2B_5B_6P_3B_7B_8$. Only the I frame can be decoded without using other frames. All the other frames are decoded

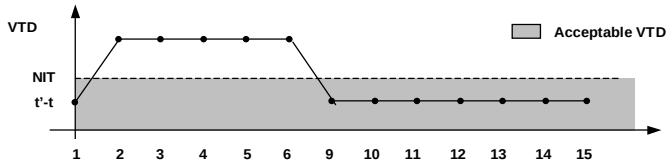

Figure 5: VTD measured while playing out the video stream when using the mechanism.

using others frames. In particular, 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. This means that if a I frame is not present, the entire GOP (plus the two B frame of the previous GOP that depend on the I frame) cannot be decoded and are discarded. Hence, 14 frames are impossible to decode in an I frame is missing. If a P_1 frame is missing, then 11 frames are impossible to be decoded; if a P_2 frame is missing, then it is not possible to decode 8 frames; if a P_3 frame is missing, then 5 frames cannot be decoded. Only a missing B frame does not result in additional frame discard.

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). This discarding algorithm discards the frames when it asked to. No consideration is done on the type of frame.

Drop I frame (DIF). This algorithm discards only frames of type I. Based on the dependency rules, 13 frames depend on the I frame and hence, as a result of discarding an I frame, 14 frames cannot be played out.

Drop P_1 frame (DP1F). This algorithms discards only frames of type P_1. Based on the dependency rules, 10 frames depend on the P_1 frame and hence, as a result of discarding a P_1 frame, 11 frames cannot be played out.

Drop P_2 frame (DP2F). This algorithms discards only frames of type P_2. Based on the dependency rules, 7 frames depend on the P_2 frame and hence, as a result of discarding a P_2 frame, 8 frames cannot be played out.

Drop P_3 frame (DP3F). This algorithms discards only frames of type P_3. Based on the dependency rules, 4 frames depend on the P_3 frame and hence, as a result of discarding a P_3 frame, 5 frames cannot be played out.

Drop B frame (DBF). This algorithms discards only frames of type B and since no frames depends on this type of frame, the discard of a B frame does not produce any domino effect on the stream.

Although the discard of a good selection of frames does not greatly affect the video QoS (see for example, [14] and [15]), the number of discarded frames should be as small as possible.

Unfortunately, while considering MPEG video traces, 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 [16]. One possible approach to accounting for the perceptual playout quality is to use a cost function to

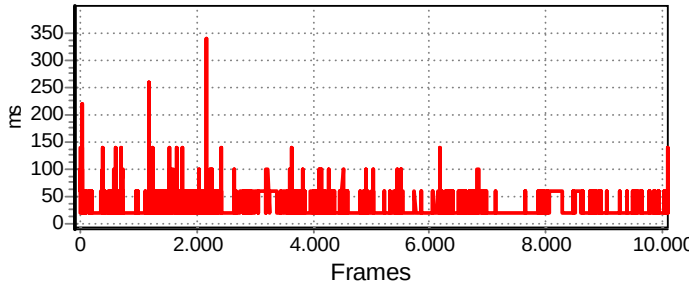

Figure 6: Video Time Difference obtained transmitting a clip of The Simpsons.

measure the perceived video quality. 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 to measure the QoS of the played out video stream, by using a cost function, and, in the following, we present results obtained from analyzing the mechanism over a LAN and over the Internet.

Simulations involve both our department LAN and the Internet and are 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 our department LAN, a 100Mb/s Ethernet network.

In Fig.6 we present the VTD measured while transmitting the first 10.000 frames of the video trace *The Simpsons*. It is possible to note the great variability of the VTD.

In Fig.7 we present the percentage of frames that goes 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 applying 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.

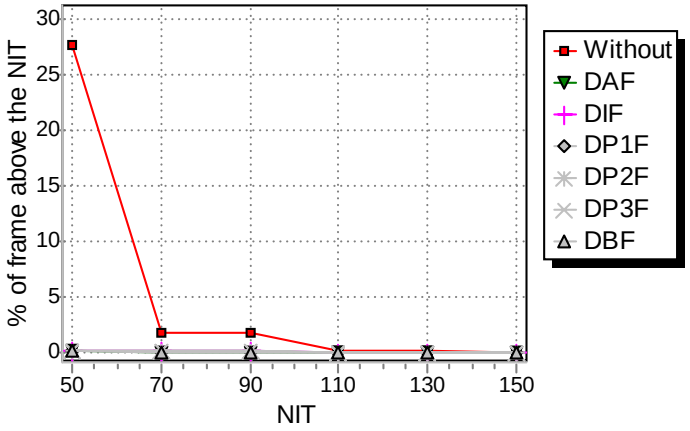

Figure 7: Transmission of *The Simpsons* over a LAN: percentage of frames with a VTD above the NIT.

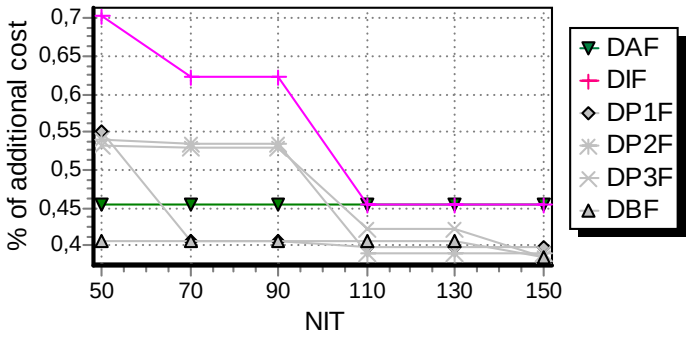

Figure 8: Transmission of *The Simpsons* over a LAN: cost of the dropping algorithms.

To maintain the percentage very close to zero, we apply all the discarding algorithms presented in 3 in order to see if some algorithms perform better than others. As shown in Fig.7, all the algorithms produce almost the same results.

To better understand the behavior of the algorithms, we compute the cost and we present them in Fig. 8. Note that the cost is presented normalized with respect to the cost computed to the original stream transmission. In fact, sometimes the network discards frames and this affects the QoS. For this reason, we computed the cost and use it to compare with the cost obtained when the dropping algorithms are applied. It is possible to note that there are small differences among the dropping frame algorithms, and their cost is close to the original one. This is due to the small number of discarder 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. 3 the delay experienced

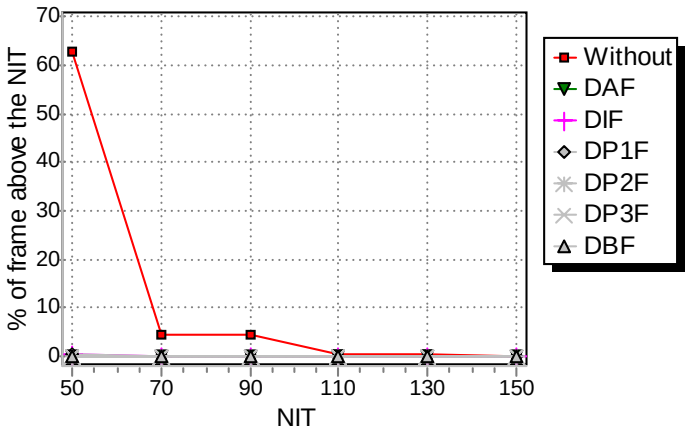

Figure 9: Transmission of *MTV* over a LAN: percentage of frames with a VTD above the NIT.

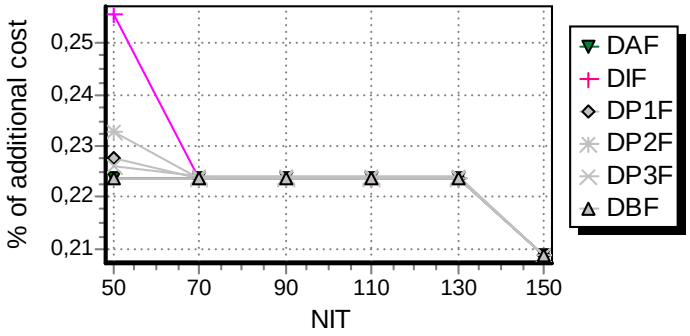

Figure 10: Transmission of *MTV* over a LAN: cost of the dropping frames algorithms.

by frame 2 causes the VTD to go above the NIT for all the successive frames. In Fig. 4 and 5 we showed that by discarding only 2 frames, only few frames have a VTD above the NIT, while all of the other frames are within the NIT limit.

A result very similar has been obtained from transmitting a video traces of *MTV* (Fig. 9). With a 50 ms NIT, more than 60% of the frames goes above the acceptable limit and the percentage decreases while increasing the NIT value. For instance, if the NIT value is equal to 110 ms, the percentage drops to 1% and reaches almost zero percent with a NIT value of 130 ms. Also in this case, the mechanism keeps the percentage very close to zero with all the tested NIT and with all the discarding algorithms. Again, the computed cost (Fig. 10) for the dropping algorithms are very similar to the original cost, and there are noticeable (although small) differences only when the NIT is equal to 50 ms. Also in this case, DIF performs worse than the other algorithm.

Results obtained from transmitting of a video traces of a *NEWS* clip is reported in Fig. 11. With a 70 ms NIT, almost 90% of the frames goes above the acceptable

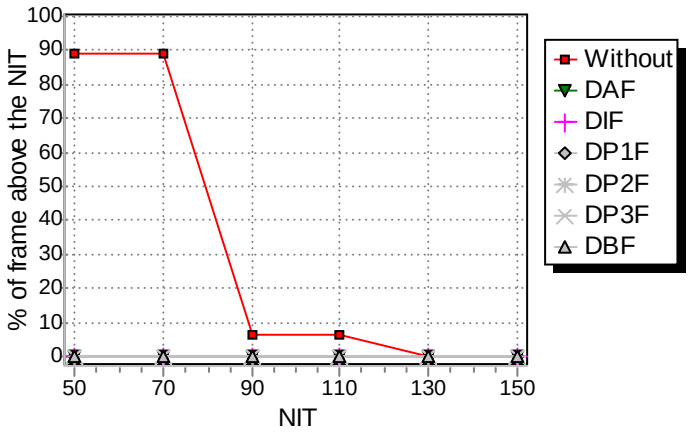

Figure 11: Transmission of *NEWS* over a LAN: percentage of frames with a VTD above the NIT.

limit and the percentage decreases while increasing the NIT value. For instance, if the NIT value is equal to 90 ms, the percentage drops to less than 10% and reaches almost zero percent with a NIT value of 130 ms. Also in this case, the mechanism keeps the percentage very close to zero with all the tested NIT. Here, it is interesting to observe the computed cost (Fig. 12): the additional cost ranges between 3% and 4.3%. This means that the mechanism dropped a number of frames greater than the previous scenarios. Among the dropping algorithms, DIF is the one that performs worse than the other, and DBF is the one that produces the lowest cost.

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).

In Fig. 13 we present results obtained from transmitting a 20 minutes video trace of a *NEWS* clip encoded with 24 fps, from Bologna to Trieste (9 hops). A NIT of 70 ms causes more than 70% of the frames to go above the acceptable limit. For a NIT value of 110 ms, the percentage drops to less than 10%, and reaches almost zero percent for a NIT value around 150 ms. Conversely, if the mechanism is used, the percentage is kept very close to zero. The additional cost ranges between 1.7% and 2.3%. Also in this case, DIF performs worse than the other algorithms, and DBF produces the lowest costs with all the tested NIT.

In Fig. 15 we present results obtained from transmitting a video trace of the cartoon *The Simpsons*, from Bologna to Trieste (9 hops). With a NIT of 70 ms, more than 80% of the frames goes above the acceptable limit. With a NIT of 110 ms, the percentage decreases to less than 10%, and reaches almost zero percent for a 170 ms NIT value. The mechanism keeps the percentage very close to zero and the computed costs are presented in Fig. 16. The additional cost is kept within 0.2% and 1.3%. Although for NIT values greater than 130 ms there is no substantial difference among the algorithms,

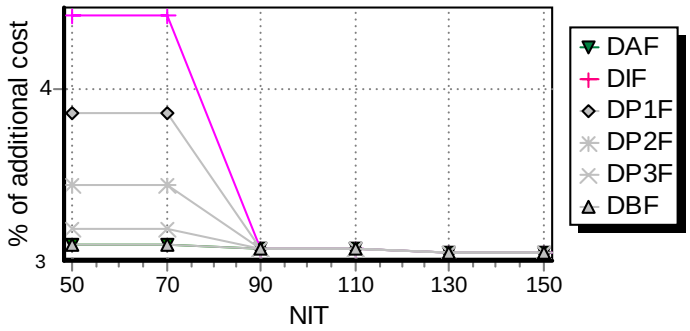

Figure 12: Transmission of *NEWS* over a LAN: cost of the dropping frames algorithms.

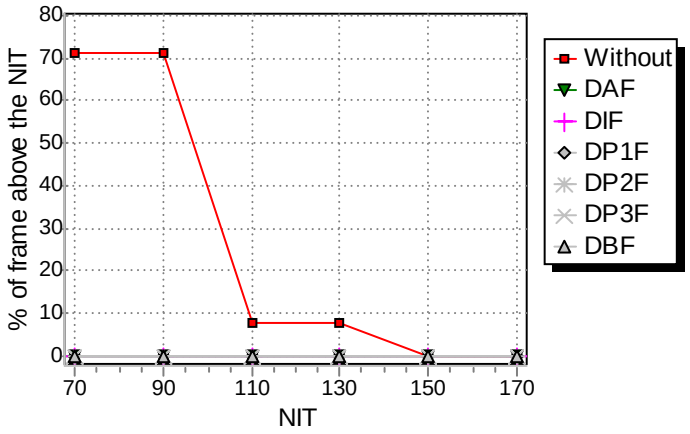

Figure 13: Transmission of *News* over the Internet (9 hops): percentage of frames with a VTD above the NIT.

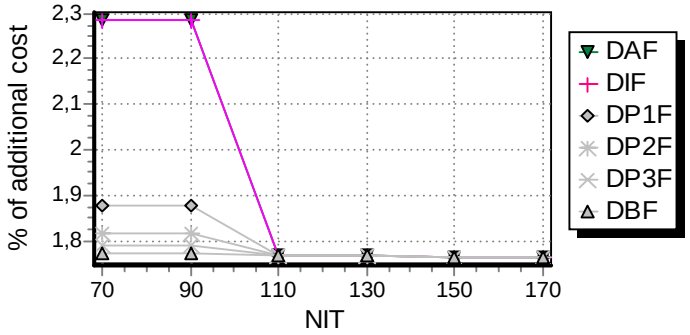

Figure 14: Transmission of *News* over the Internet (9 hops): cost of the dropping frames algorithms.

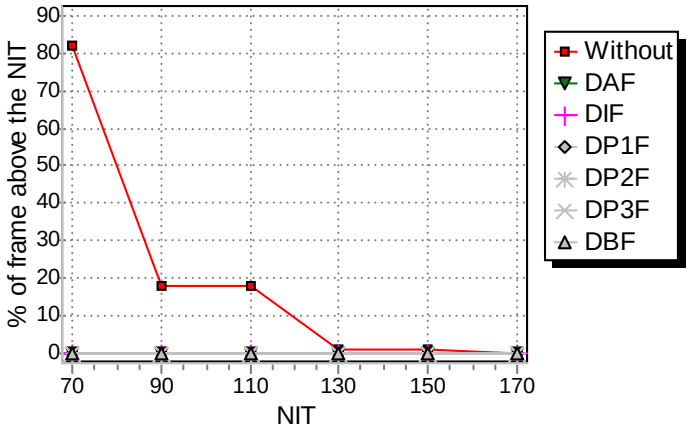

Figure 15: Transmission of *The Simpsons* over the Internet (9 hops): percentage of frames with a VTD above the NIT.

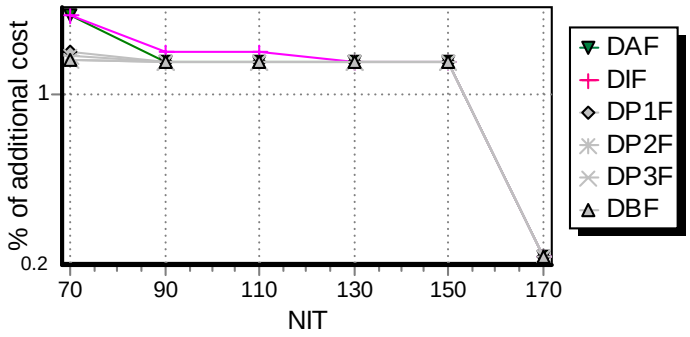

Figure 16: Transmission of *The Simpsons* over the Internet (9 hops): cost of the dropping frames algorithms.

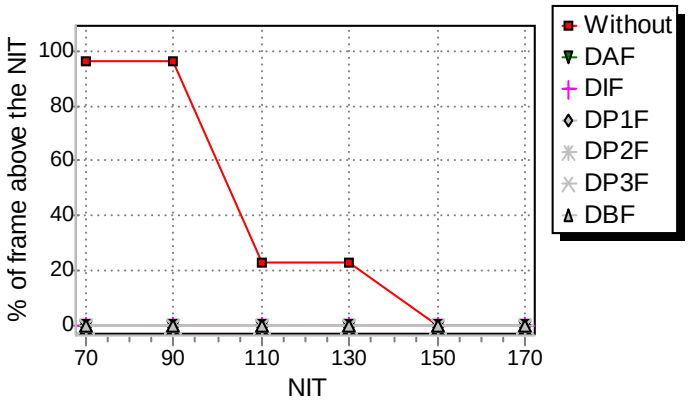

Figure 17: Transmission of *MTV* over the Internet (9 hops): percentage of frames with a VTD above the NIT.

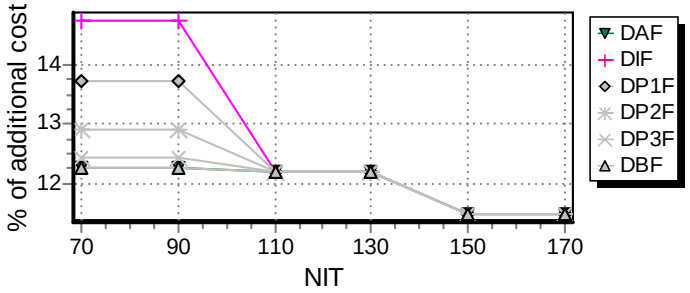

Figure 18: Transmission of *MTV* over the Internet (9 hops): cost of the dropping frames algorithms.

DIF performs worse and DBF perform better than the other algorithms, for NIT values lower than 130 ms.

In Fig. 17 we present results obtained from transmitting a video trace of *MTV* from Bologna to Trieste (9 hops). With NIT values of 70 and 90 ms, almost the entire stream goes above the acceptable limit. The percentage decreases as the NIT increases and reaches acceptable values with NIT starting from 150 ms. Conversely, the dropping algorithms allow the mechanism to keep the percentage close to zero. In Fig. 18 is reported the dropping cost. In this case the dropping algorithms introduce a considerable cost. To keep the percentage, of the frames that go above the NIT, very close to zero, the mechanism has to discard several frames, produce a cost that ranges between 11.5% and 14.5%. Once again, DIF performs worse and DBF performs better than the other algorithms.

In Fig. 19 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. We measure the percentage of frame that goes above the NIT for NIT values between 100 and 600 ms. With a NIT value of 100 ms, more

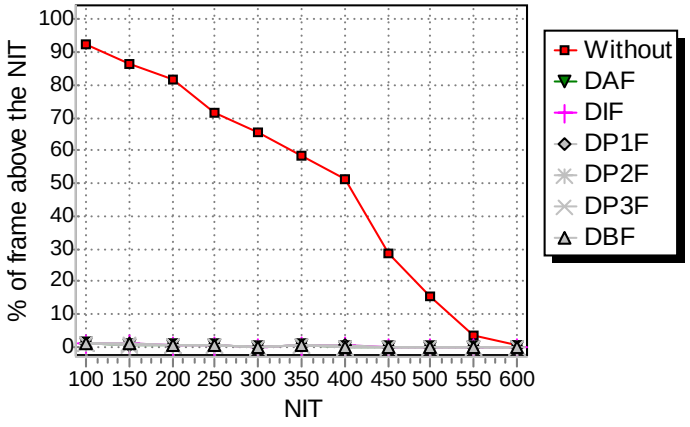

Figure 19: Transmission of *MTV* over the Internet (8 hops): percentage of frames with a VTD above the NIT.

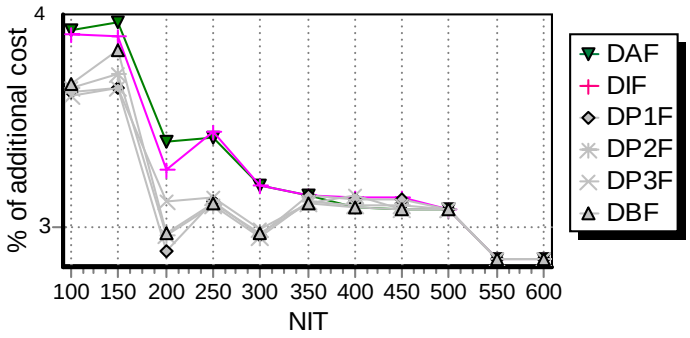

Figure 20: Transmission of *MTV* over the Internet (8 hops): cost of the dropping frames algorithms.

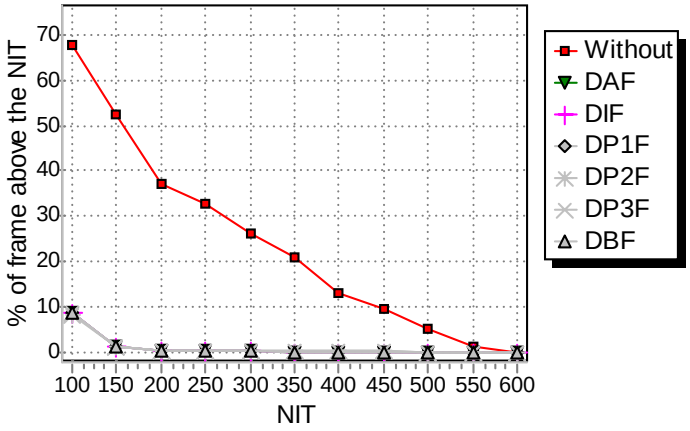

Figure 21: Transmission of *NEWS* over the Internet (8 hops): percentage of frames with a VTD above the NIT.

than 90% of the frames go above the acceptable limit. The percentage decreases as the NIT increases and only for NIT values greater than 500 ms the percentage drops to less than 20%. Conversely, the dropping algorithms allow the mechanism to keep the percentage close to zero. In Fig. 20 is reported the dropping cost. 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.

In Fig. 21 we present results obtained from transmitting a video trace of a *NEWS* clip, from Bologna to Alessandria (8 hops).

We measure the percentage of frame that goes above the NIT for NIT values between 100 and 600 ms. With a NIT value of 100, 70% of the frames go above the acceptable limit and the percentage decreases as the NIT increases and only for NIT values greater than 450 ms the percentage drops to less than 10%. 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. 22 is reported the dropping cost. For a NIT value equal to 100 ms, the cost may increase up to 37% with respect to the original cost. This highlights the great network problems experienced while transmitting the video stream. The cost decreased 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.

In Fig. 23 we present results obtained from transmitting a video trace of *The Simpsons*, from Bologna to Alessandria (8 hops). We measure the percentage of frame that

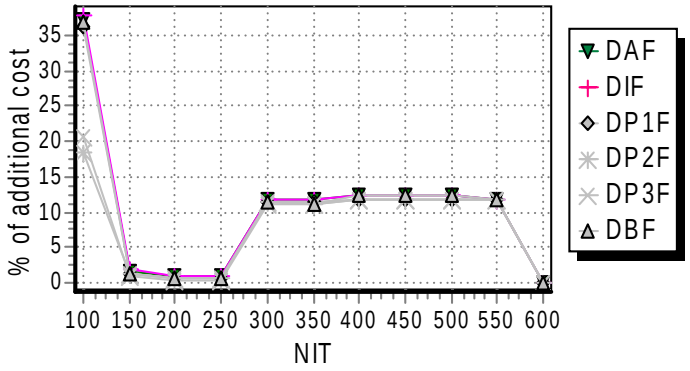

Figure 22: Transmission of *NEWS* over the Internet (8 hops): cost of the dropping frames algorithms.

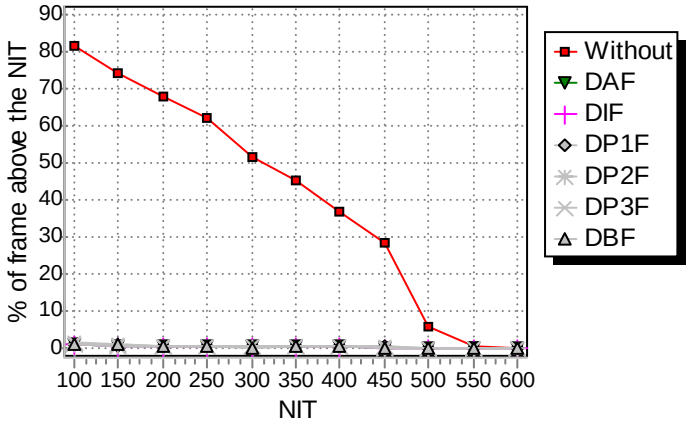

Figure 23: Transmission of *The Simpsons* over the Internet (8 hops): percentage of frames with a VTD above the NIT.

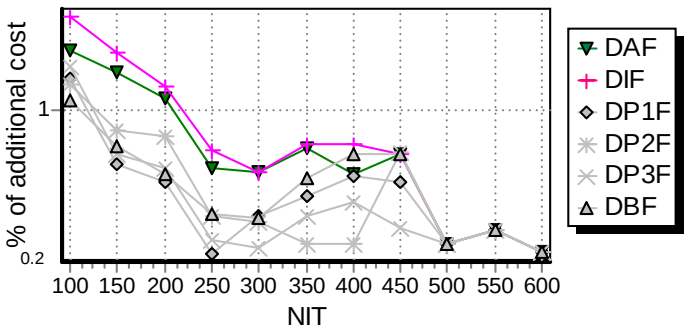

Figure 24: Transmission of *The Simpsons* over the Internet (8 hops): cost of the dropping frames algorithms.

goes above the NIT, for NIT values between 100 and 600 ms. With a NIT value of 100, more than 80% of the frames go above the acceptable limit and the percentage decreases as the NIT increases and only 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. In Fig. 24 is reported the dropping cost. Here, the number of dropped frames is very small and hence, the additional cost is kept within the 1.3%.

Despite the network problems, all the performed experiments confirmed the benefits of the mechanism, as the VTD is maintained within the acceptable limit. In all the performed experiments, we noticed that there is no substantial difference among the dropping algorithms. This showed the effectiveness of the mechanism. However, to better understand the effects of the mechanism on the video QoS, the computed cost allows us to choice one of the dropping frame algorithms. In most of the performed experiments, 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 (no domino effect produced while discarding a B frame).

4 Conclusions

In this paper we evaluated a recently proposed mechanism for supporting interactive video streaming applications. The mechanism was presented in [12, 13] and uses a new approach to support interactive video applications over the Internet. Roughly, it acts on the video QoS (i.e., it drops video frames) to mask the network jitter.

In [12, 13] the mechanism evaluation has been done considering only video stream transmission with videos encoded with Motion JPEG. Since Motion JPEG is an intra-frame technique, it is possible to discard a frame without causing problems to the decoding of other frames. Conversely, with inter-frame encoding mechanisms (e.g., MPEG), video frames are encoded/decoded using information present in other frames. Without these information, it is not possible to decode the considered frame. Hence,

the discard of a frame may produce a domino effect, increasing the number of discarded frames.

The contribution of this paper was to test the mechanism proposed in [12, 13] with MPEG video traces. Several experiments have been conducted, and results showed that the benefits 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 or Motion JPEG video applications over the Internet, and its benefits can ameliorate the system reaction to the end-user commands (pause, fast forward, rewind).

References

[1] ISO/IEC, Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5 Mbit/s, *International Organization for Standardization*, 1993.

[2] B. Furht, A survey of multimedia compression techniques and standards. Part I. JPEG standards, *Real Time Imaging*, 1, 1995, 49-67.

[3] ITU-T, Recommendation H.261. September 1994.

[4] D. Sitaram, A. Dan, Multimedia Servers: Application, Environments, and Design, *Morgan Kaufmann Publishers*, 2000.

[5] T. Kurita, S. Iai, N. Kitawaki, Effects of transmission delay in audiovisual communication, *Electronics and Communications in Japan*, 77(3), 1995, 63-74.

[6] M. Roccetti, V. Ghini, G. Pau, P. Salomoni, M. Bonfigli, Design and Experimental Evaluation of an Adaptive Playout Delay Control Mechanism for Packetized Audio for Use over the Internet, *Multimedia Tools and Applications*, 14(1), 2001, 23-53.

[7] M. Furini, M. Roccetti, Adaptive Video Transmission over the Internet: an Experimental Study, *Proc. of 2000 European Simulation Symposium*, Hamburg, Germany, 2000, 159-163.

[8] M. Baldi, Y. Ofek, End-to-End Delay Analysis of Videoconferencing over packet-Switched Networks, *IEEE Trans. on Networking*, 8(4), 2000, 479-492.

[9] W. Feng, S. Sechrest, Optimal buffering for the delivery of compressed prerecorde video, *Proc. of the IASTED Intern. Conference on Networks*, 1995.

[10] J. Salehi, Z. Zhang, J. Kurose, D. Towsley, Supporting Stored Video: Reducing Rate Variability and End-to-End Resources Requirements through Optimal Smoothing, *IEEE Trans. on Networking*, 1998.

[11] WEBTV NETWORKS INC., Web TV technical specifications, www.webtv.net/HTML/home.specs.hmtl

[12] M. Furini, M. Roccetti, Design and Analysis of a Mechanism for supporting Interactive Video Streaming Applications over the Internet, Technical Report, DISTA Department, University of Piemonte Orientale, www.di.unipmn.it, 2002.

[13] M. Furini, M. Roccetti, Design and Analysis of a Mechanism for supporting Interactive Video Streaming Applications over the Internet, Proceedings of the IASTED/IMSA Conference, Kauai, August 2002.

[14] Z. Zhang, S. Nelakuditi, R. Aggarwal, R. Tsang, Efficient Selective Frame Discard Algorithms for Stored Video Delivery across Resource Constrained Networks, *Proc. IEEE INFOCOM'99*, NYC, 1999.

[15] M. Furini, D. Towsley, Real-Time traffic transmission over the Internet, *IEEE Transaction on Multimedia*, 3(1), 2001, 33-40.

[16] Wijesekera, D., Srivastava, J. : *Quality of Service (QoS) Metrics for Continuous Media*, Multimedia Tools and Applications, 2(3), 1996, 127-166.