# Experience with Control Mechanisms for Packet Video in the Internet

J-C. Bolot and T. Turletti
INRIA
2004, route des Lucioles
06902 Sophia Antipolis Cedex
France
bolot,turletti@sophia.inria.fr
http://www.inria.fr/rodeo/personnel/{bolot,turletti}

## Abstract

The single class best effort service available in the current Internet does not provide the guarantees, typically expressed in terms of minimum bandwidth and/or maximum delay or loss, associated with real-time applications such as live video. One way to support such applications in best effort networks is to use control mechanisms that adapt the coding, transmission, reception, and decoding processes at the source and at the destination(s) depending on the state of the network. In this paper, we examine and report on our experience over the past several years with such mechanisms for videoconferencing software. We illustrate our points with results obtained with the IVS software developed at INRIA.

We consider in particular rate and error control mechanisms. These mechanisms adapt the bandwidth requirements and the resilience to packet loss of the video stream sent by a source coder. We have found that they do prevent video sources from swamping the resources of the Internet, and that they provide reasonably graceful degradation of image quality during periods of network congestion. However, they raise thorny issues of fairness with non-real time traffic sources (such as TCP sources) and of scalability for the multicast delivery of video in heterogeneous networks. We examine different approaches to tackling these problems.

## 1 Introduction

Video has conventionally been transmitted over telephone and CATV networks, which provide connections with constant or nearly constant capacity channels. However, the rate of a video sequence can vary rapidly with time due to the effects of scene complexity and motion. The problem, then, is to obtain from the variable rate sequence a constant rate stream of data that can be sent into the constant bit rate channel. This is typically done by sending the variable rate stream into a buffer which is drained at a constant rate. The amount of data in the buffer is used as feedback information by a rate control mechanism which ensures that the buffer does not overflow or underflow, and thus that all the video data arrives at the destination. The rate control mechanism avoids buffer overflow or underflow by adjusting the coding process based on the feedback information. Simple control mechanisms in which the coder behavior is fixed for a given buffer occupancy level have been used for more than 25 years [15]. More elaborate mechanisms that rely on models of the coder behavior and on coding rate predictions have been proposed recently [43]. In all cases, the main idea is to decrease the video quality for scenes of higher complexity, and to increase the quality for scenes of lower complexity so as to keep the output rate approximately constant.

Video has more recently been transmitted over networks such as the Internet that provide a single class best effort service. In the paper, we consider the transmission of live video as is done in videoconference applications using IP multicast, UDP, and RTP [10, 24, 35]. From a connection endpoint's point of view, the best effort service amounts in practice to offering a channel with characteristics such as available bandwidth, delay, and loss, that vary over time. Furthermore, these variations are not known in advance since they depend on the behavior of other connections throughout the network. This makes it hard to provide predictable service to applications such as video applications, the quality of which depends on the bandwidth, delay, and loss characteristics of the network.

Two approaches to this problem have been proposed. The first approach is to augment the best effort service provided by the network with other services providing various degrees of performance guarantees. This requires that specific resource allocation and/or reservation mechanisms be implemented in the network. These mechanisms have not yet been widely deployed in the Internet, thus our focus so far has not been on this approach.

The second approach is to adapt application behavior to the service provided by the network, i.e. to the time-varying characteristics of the channels over which the application data packets are sent. This latter approach can be taken in the current Internet. Furthermore, it will remain of interest even in a network with explicit resource allocation and reservation. For example, adaptation can be used to handle uncertainties in cases when it it is hard to make accurate *a priori* reservation requests. Also, resource reservation comes at a price; one way then would be to reserve enough for the more important part of the data flow, and use adaptation for the rest of the data.

The important channel characteristics for live video delivery are the end to end delay, available bandwidth, and loss. Our goal then is to develop control mechanisms to adapt to variations in these parameters.

The efficient control of end to end delay requires more elaborate queueing disciplines in the routers than the single class FIFO discipline used in most of the current Internet. Furthermore, the efficient control of delay jitter with adaptive playout algorithms is a now well understood problem. Therefore, our focus in this paper is on mechanisms to adapt to available bandwidth and loss. There are two such types of mechanisms. Rate control mechanisms attempt to match the bandwidth requirements of a video connection to the bandwidth available over the path of the connection. Thus, they attempt to minimize network congestion and the amount of packet loss. However, they do not prevent packet loss altogether. Error control mechanisms then attempt to minimize the visual impact of loss at the destinations.

Designing efficient control mechanisms for multimedia applications is an important task in the current Internet. A large number of companies have released (with more expected in the future) UDP-based commercial videoconferencing tools for the Internet that do not include any kind of mechanism to adapt for example bandwidth requirements depending on network congestion[1]. This makes it very easy for users to (possibly unknowingly) swamp the resources of the Internet and thus degrade the performance observed by all other connections, in particular TCP (and thus Web) connections.

Rate and error control mechanisms can be either source based or destination based. Source based mechanisms, in which the source collects the feedback information, carries out the control algorithms and takes the control actions, were the first ones used in the Internet. We examine them in Section 2 (rate control) and Section 3 (error control). However, source based mechanisms do not scale well in heterogeneous networks. We review their limitations and describe approaches to overcoming them in Section 4.

Throughout the paper, we illustrate the performance of our schemes with measurements and results obtained with IVS, a software videoconference system for the Internet developed at INRIA [35]. IVS was the main tool used in the MICE European-funded project [2] to hold weekly videoconference meetings, to multicast conferences, etc. At the time, IVS was the only tool on the MBone which included automatic rate and error control mechanisms.

## 2 Rate control mechanisms

Rate control mechanisms have been widely used in non real-time applications in the Internet, the more well-known example being the source-based control mechanism in TCP which adjusts the window size at the source based on estimated network congestion [18]. We mentioned earlier the source-based rate control mechanism used in CATV networks to control the video coding process based on a local buffer occupancy. Source-based mechanisms were also proposed early on for real-time video applications over wide-area networks [19, 39, 20, 3, 7]. These mechanisms control the output rate of an audio or a video coder based on feedback information about the state of the network. They appear somewhat similar to those designed for CATV, except that feedback is local for CATV but network wide for wide-area networks. Thus, it would seem possible to use the same mechanisms designed for CATV networks. However, designing efficient mechanisms for the Internet requires that one solve a problem more difficult that the rate control problem in CATV networks. This is because i) the characteristics of the channel are now time varying, ii) these variations are caused by the presence of other traffic sources with characteristics and requirements different from those of our video source, and iii) the "channel" we consider is in practice a multicast tree, which is made of multiple heterogeneous channels each with its own bandwidth, delay, and loss characteristics.

Points i) and ii) above make it hard to design and analyze mechanisms for our problem using the single-buffer single-channel approach examined for the constant bit rate (e.g. the CATV) control problem. More appropriate models would consider channels with time-varying service rates. Such models have not yet been examined, but it is already clear that they are difficult to solve (although simple models can be solved relatively easily, see e.g. [28]).

We next examine how source based rate control can be made to work in practice. Specifically, we consider whether video coders can adjust their output rate (i.e.

[1] It is hard to blame the commercial community for this since very few of the tools developed by the research community over the past few years and popular among MBone users use any kind of rate control, even though network congestion and stability are fundamental concerns of the research community. Also, commercial adaptive applications are being introduced, see for example [9].

whether rate control for video applications is possible at all), how the state of the network is characterized and how feedback information is elicited and brought back to the source, and how coders adjust their bandwidth requirements in response to this information.

## 2.1 Rate adjustment

We consider coders with block-based transform coding schemes such as those used in Motion-JPEG, MPEG, and H.261 [16].

It turns out that many parameters of such coders can be adjusted to change the output rate of the coder [35]. For example, decreasing the rate at which video frame are grabbed at the source decreases the output rate of the coder. Increasing the value of the quantizer is equivalent to encoding the frequency coefficients more coarsely, and thus reducing the number of bits used to encode pixels, i.e. the output rate of the coder. Increasing the movement detection threshold decreases number of blocks which are considered to have changed since the last frame. Therefore, the number of bytes required to encode each image, and hence the output rate of the coder, decreases.

In all cases, decreasing the output rate also decreases the quality of the video sent by the coder. The specific requirements of a video application dictate which of the three parameters described above should be modified when adjusting the output rate of a coder. The grabbing rate is modified if the precision of the rendition of individual images is important. The quantizer and the movement detection threshold are modified if the perception of movement is important. These requirements are taken into account in IVS as follows. The user (i.e. the source of video traffic) specifies a maximum output rate, which we denote by $max\_rate$, and a mode. $max\_rate$ is the maximum rate at which the video stream can leave the coder. The mode characterizes which parameters are adjusted in the coder so that the output rate stays below $max\_rate$. In the Privilege Quality mode (PQ mode), only the frame grabbing rate is adjusted. In the Privilege Rate mode (PR mode), only the quantizer and the movement detection threshold are adjusted.

## 2.2 Feedback collection

The coder sends video packets using the Real-Time Transport Protocol (RTP) [31] together with UDP and IP multicast. RTP consists of two parts, a data part and a control part referred to as the RTCP, the Real-Time Control Protocol. Video data sent by the source is carried in RTP data packets. A RTP data packet includes a 12-byte header followed by the payload, i.e. the video data proper (excluding UDP and IP headers). The packetization of video frames is encoding specific. It has been defined for H.261 in [36].

Feedback information is carried in RTCP packets referred to as Receiver Reports (RR). RRs are multicast regularly to the same multicast group as data packets, i.e. they are multicast to the source as well as to all the destinations in the group. The rate at which they are multicast is controlled so that the load created by the control information is a small fraction of that created by data traffic [31].

The RR sent by a destination includes information on the quality of the video observed by this destination, specifically the highest sequence number received, the number of packets lost, the estimated packet interarrival jitter, and timestamps (needed to compute end to end delays). Note that the loss rate of the encoded blocks cannot be computed by the receivers either since only the coder knows which blocks in a frame were actually encoded. However, the loss rate for the blocks can be crudely approximated by the packet loss rate.

## 2.3 Rate control algorithm

The source thus receives RRs from all destinations in the multicast group. It uses the information in the RRs to estimate congestion in the network, and quality of the video at the destinations. The control algorithm then adjusts the maximum output rate $max\_rate$ of the coder using the familiar linear increase / multiplicative decrease algorithm advocated for TCP and other mechanisms because of its performance and fairness properties [8]. Specifically, $max\_rate$ is decreased by a multiplicative factor referred to as $GAIN$ in the case of congestion; it is increased by a fixed amount $INC$ in the absence of congestion. The algorithm also makes sure that the output rate stays within a $[MIN\_RATE, MAX\_RATE]$ interval. In IVS, the values of $MAX\_RATE$ and $MIN\_RATE$ can be set by the user. Thus, the control algorithm is as follows:

if $Congestion$
  $max\_rate = \max(max\_rate/GAIN, MIN\_RATE)$
else if $NoCongestion$
  $max\_rate = \min(max\_rate + INC, MAX\_RATE)$

We make two observations. First, the algorithm can be used either in PQ or PR mode (i.e. rate adjustments translate into changes of the frame rate, or of the quantizer and movement detection threshold). Second, the algorithm differs from that used in TCP in two important ways, namely i) it is a rate control algorithm whereas TCP's is a window control algorithm. and ii) the rate adjustments do not exactly match the window adjustments in TCP. A consequence of i) is that the dynamics of our algorithm will differ from that of a TCP connection. A consequence of ii) is that a video connection and a TCP connection sharing the same resources on the same path will not share the bandwidth equally (this is illustrated e.g. in [34]). The share of the resources grabbed by the video connection depends on the values of the parameters of the algorithm, and thus on the meanings given to $Congestion$ and $NoCongestion$, and on the values of $GAIN$ and $INC$. The question then is how to pick "good" values for these parameters.

Unfortunately, we did not find a good answer to this question, and we ended up picking ad hoc values. The one simple choice we made was to use only loss information in RRs as congestion indicator. We could have used other information such as the delay jitter in RRs or explicit congestion information. We did not use the jitter information because we did not have a good handle on the correlation between jitter and congestion. We did not use explicit information because it is not available in the current Internet. Furthermore, it was not clear how to design control mechanisms using non-loss based feedback information for a network in which essentially (especially at the time) all the controlled traffic was regulated using loss-based feedback information.

In any case, we used $Congestion$ to mean that the loss rate reported by RTCP exceeds some threshold $high\_loss$, and $NoCongestion$ to mean that the loss rate is below another threshold $low\_loss$. The value of $high\_loss$ and $low\_loss$ should be chosen so as to correspond to perceptually important loss threshold values. Unfortunately, such values depend on the application, on the coding scheme used, and on the available network support. In early versions of IVS, we had static values $low\_loss = 2\%$ and $high\_loss = 5\%$.

The problem of designing rate control mechanisms for real-time applications has seen renewed interest recently because of the increase (with more expected) in non rate-controlled UDP traffic in the Internet. One proposed solution is to use so-called TCP friendly mechanisms. The idea is to explicitly estimate the bandwidth available to a TCP connection that would be transferring data between the same source and destination. This estimation is done using analytic models of TCP. For example, the model mentioned in [23] finds that the bandwidth $\lambda_{equ}$ of a TCP connection can be well represented by Equation 1 below

$$\lambda_{equ} = 1.22 * \frac{MTU}{rtt * \sqrt{p}} \tag{1}$$

where $MTU$ is the maximum packet size used on the connection, $rtt$ is the mean round trip time, and $p$ is the mean packet loss rate. The source can find $MTU$, compute $rtt$ (using RTP timestamps as shown in [31]), and it receives $p$ in RTCP RRs. It can then compute $\lambda_{equ}$ and make sure that $max\_rate$ does not exceed that value. This way, we have obtained a TCP-friendly scheme suitable for unicast delivery.

This approach is attractive, but it does have limitations. In particular, different versions of TCP include different algorithms for the window-based congestion control scheme, with different throughput characteristics [22]. Thus, it is not clear which version of TCP to be friendly with. Also, analytic models rely on assumptions (such as Bernoulli loss in [23]) and simplifications (such as not modeling timeouts) that do not hold in practice. Thus, the robustness of results such as that in Equation 1 is not clear. Ultimately, the problem of TCP friendliness, and more generally of designing control mechanisms for applications with diverse characteristics, will be solved only with the support of adequate scheduling algorithms in routers.

Let us now get back to the control algorithm of IVS, and consider the case of a multicast connection. We need to examine two problems, namely the feedback explosion problem and the heterogeneity problem.

The amount of feedback information sent by the receivers increases as the number of receivers increases. This generates a feedback explosion problem in the case of large multicast groups. We solved this problem in IVS by using first the probabilistic sampling scheme with expanding scope of [4], then the scaled feedback scheme of RTCP [31].

The heterogeneity problem stems from the fact that loss rates might be widely different on different branches of the multicast tree. With source-based control, the source has to convert the loss rates measures from the different destinations into a "global" loss rate characterizing how well packets arrive at the receivers. One approach would be to take some average of the loss rates measured by each receiver. Another approach would be to take the highest loss rate measured by any receiver. The approach used in IVS is to change the output rate only if greater than a threshold percentage of the receivers are observing sufficiently low or high packet loss.

Specifically, let $N$ denote the number of destinations, $N_c$ the number of destinations for which the observed loss rate exceeds $high\_loss$, and $N_u$ the number of destinations for which the observed loss rate is below $low\_loss$. The control algorithm then periodically adjusts the maximum output rate $max\_rate$ of the coder as follows: $max\_rate$ is decreased by a multiplicative factor referred to as $gain$ if the "global" loss rate is larger than $high\_loss$; it is increased by a fixed amount $inc\_rate$ if the "global" loss rate is lower than $low\_loss$; it is unchanged otherwise. Thus, the control algorithm is similar to that in the unicast case with

$$Congestion \equiv N_c/N \geq 0.1$$
$$NoCongestion \equiv N_u/N \geq 0.1$$

### 2.4 Experimental evaluation

We had the opportunity to use the rate control mechanism over 3 years in the course of weekly MICE/MERCI meeting, monthly MICE seminars, etc. We evaluate its performance and illustrate its limitations with results from experiments carried out over the MBone. Specifically, we examine here two sets of experiments, the first one carried out over part of the MBone inside INRIA (over a test network the bottleneck link of which is a 100 kb/s ISDN line), the second one carried out over the wide area MBone between INRIA, UCL (in the UK), and LBL (in California). In all cases, there is a single video source sending from INRIA.

Figure 1 shows the evolutions of the maximum output rate $max\_rate$ (plain line), the actual output rate (heavy dotted line), and the loss rate at the destination (light dashed line) as a function of time. The measurements were obtained for a unicast connection over the INRIA test network. Parameter values were set to $MIN\_RATE$ = 10 kb/s and $MAX\_RATE$ = 100 kb/s. As expected,

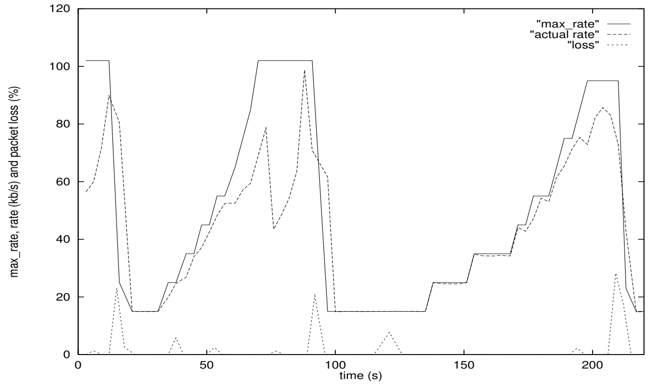


Figure 1: Evolutions of $max\_rate$ (in kb/s) and the loss rate at the receiver (in %).

we observe that $max\_rate$ decreases when losses are observed at the destination. Thus, source bandwidth requirements adapt to perceived network congestion, which prevents sources from swamping Internet resources. If the network is found by the receiver to remain congested for a long time, then $max\_rate$ eventually reaches its minimum value. This is visible for $n \approx$ 4200.

Note that packet losses reported by destinations can be caused either by network congestion or by overload at the destination itself. Indeed, destinations with little CPU power (or in which the load created by applications other than the video application is high) might not have enough CPU cycles left to decode and handle all the video packets they receive. This situation turns out to be quite common in LAN environments with diverse hardware platforms. For example, a videoconference held over INRIA's Ethernet LANs typically experiences very few packet losses in the network proper. Yet the number of losses is extremely large if a fast workstation sends high bit rate video to a slow PC. Thus, it is important to take loss at the destination into account in the feedback information to prevent the source from overwhelming the destination. We illustrate this with an experiment carried out between a SunSparc 20 source and a slower Sun 4/50 destination, both connected to the same Ethernet segment. Figure 2 shows the evolutions of the maximum source output rate (plain line), the actual output rate (dashed line), and the loss rate at the destination (dotted line). We observe in Figure 2

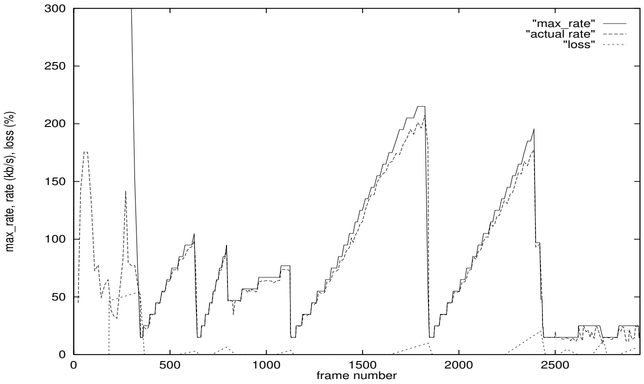


Figure 2: Evolution of $max\_rate$, actual output rate (in kb/s), and loss rate at the destination (in %) vs frame number

that the maximum send rate $max\_rate$ stays equal to its maximum value $MAX\_RATE$ = 300 kb/s for the first 400 frames. The actual send rate varies between 40 and 180 kb/s. The interesting point is that the first 200 frames are received at the destination with no loss. However, the loss rate measured over the next 200 frames exceeds 50%. This is because the slow destination workstation buffers the packets it does not have time to handle. After a while, the buffer fills up and packets are lost. When the source becomes aware of this, it drops its send rate all the way to its minimum value to prevent further losses. The evolutions of $max\_rate$ then display the oscillations typical of linear increase / multiplicative decrease mechanisms with delayed feedback.

We have shown above that the rate control scheme prevents video sources from swamping Internet resources since a high loss rate in the network (and hence congestion) will force all sources to reduce their rates. Rate control therefore provides a clear *social benefit*. However, it also provides *individual benefit* when all sources use it. We illustrate this using the results of an experiment carried out at INRIA in which a variable number of video flows share a 100 kb/s serial link. The initial value of $max\_rate$ for all sources is equal to 100 kb/s. Consider first the case when a single source is active. Figure 3 shows the number of frames per second (top graph) and the loss rate measured at the destination for both cases when the source does and does not use rate control. The plain line shows result with rate control, the dashed line without rate control. We observe that without rate control, the frame rate fluctuates around 15 but the loss rate is very high, averaging about $20\%$. With rate control, the loss rate is very low, averaging about $1\%$. However, when losses are detected, the control mechanism makes sure that only enough frames are grabbed and encoded so as not to exceed the value of $max\_rate$ at any given time (recall that all experiments are carried out in PQ mode). Thus, the control mechanism trades off a lower loss rate for a lower frame rate. We note that this is precisely what it was designed for since in PQ mode, the emphasis is placed on the quality of individual frames (which degrades with loss) rather than on the precise rendition of movement (which degrades with lower frame rate). The overall result is that the mechanism provides the destination with a better quality, given the chosen definition of quality. Similar results would be obtained in PR mode.

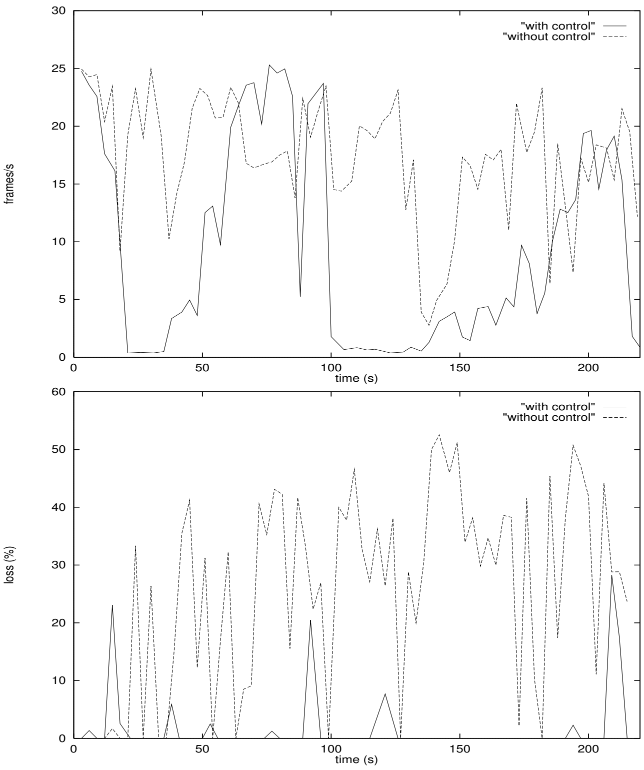


Figure 3: Evolutions of the frame rate (top) and the loss rate (bottom) at the destination with and without rate control – One source

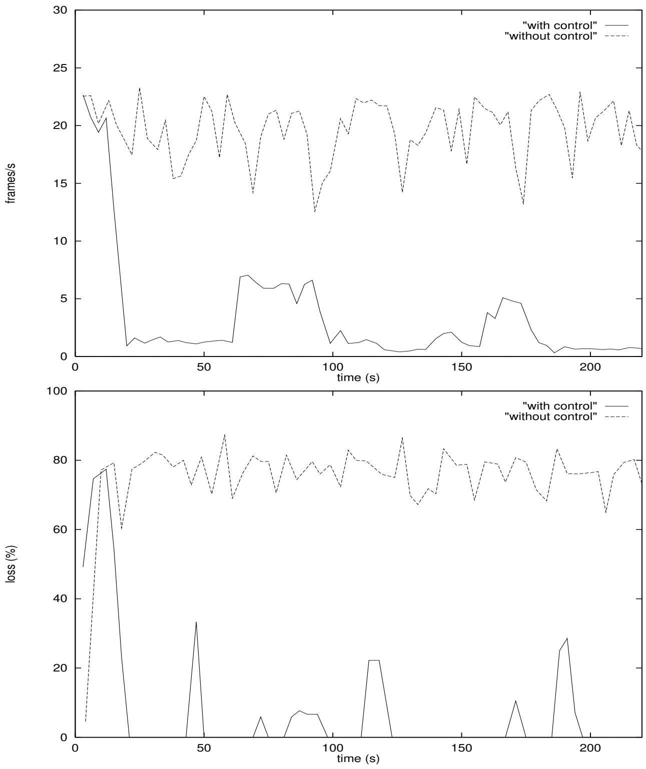


Figure 4: Evolutions of the frame rate (top) and the loss rate (bottom) at the destination with and without rate control – Three sources

The result above is confirmed in the case when three sources share the serial link. Refer to Figure 4. The frame rate without rate control is high and remains close to 20 fps. However, the overall quality is very low because the loss rate exceeding $70\%$ means that the video is made of fast moving images of extremely bad quality. With the rate control mechanism switched on, the loss rate remains low with short lived peaks not exceeding $30\%$ (except for the initial transient). The frame rate is low, but the overall quality is good given the user choice of PQ type control, i.e. given the emphasis on individual picture quality.

We have used the above mechanism to transmit a variety of conferences/events in the MICE project. Our MICE partners have generally found that videoconference quality improved when the rate control scheme was added to IVS. Nevertheless, the scheme has one important limitation illustrated in the figures below, which show measurements carried out during a 4 hour conference between INRIA, UCL, and LBL. Figures 5 shows the evolution of $max\_rate$ and Figure 6 the corresponding loss rate observed at both UCL (plain line) and LBL (dashed line).

The loss rate in the MBone was larger than the higher threshold $high\_loss$ for long periods of time (not an uncommon situation in today's Internet). As a result, the maximum allowed send rate at the source remained stuck at its minimum value of 15 kb/s for long periods of time, which did translate into a conference with overall mediocre quality. The problem of course arises because too many

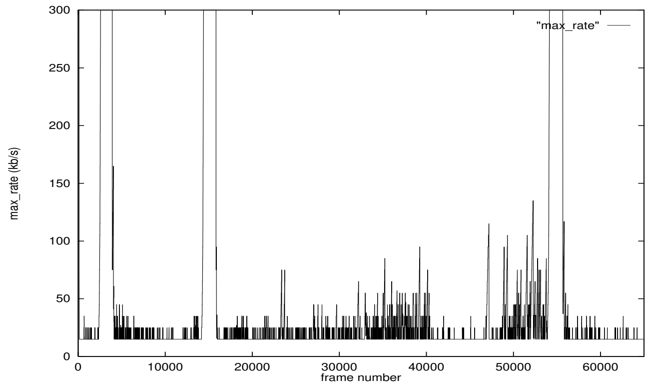


Figure 5: Evolution of $max\_rate$ (in kb/s) vs. frame number

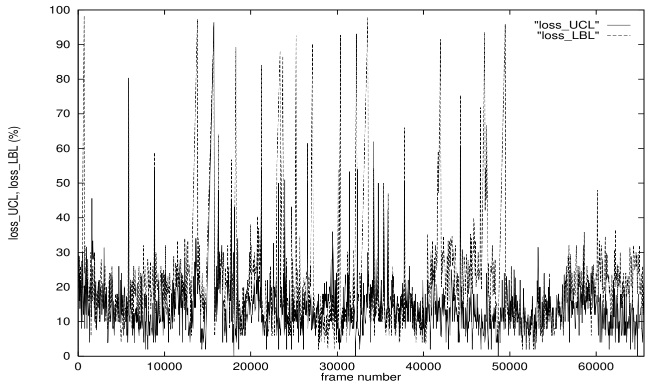


Figure 6: Evolution of the loss rate (in %) vs. frame number

applications compete for the same bandwidth. Furthermore, only making sure that the send rate of the video stream does not go below the minimum rate $min\_rate =$ 15 kb/s does not provide an appropriate minimum quality of service. This is because i) even at the lowest rate, many packets are lost (Figure 6 shows that the loss rate at both destinations is consistently higher than 10%, and ii) the 15 kb/s value might be too small.

One way to solve this problem[2] is to turn off the video, on the ground that there is not enough available bandwidth to provide the desired service and that this bandwidth might be better used by another application (such as audio or shared whiteboard). Clearly, this is not always satisfactory. Interestingly, however, our experience in MICE and MERCI has been that users often do turn off the video in cases like that described above, generally in the hope of having better audio quality as a result. Another way is to increase the value of $min\_rate$ from its current value of 15 kb/s. However, doing this might lead to severe and sustained congestion if video transfers become widespread. Yet another way is to attempt to live with the losses, but to minimize their impact on video quality at the destination. We examine this approach next.

[2] Recall that we do not consider here solutions that involve changing the network architecture or modifying scheduling disciplines in the routers.

# 3 Loss control mechanisms

## 3.1 Loss control using robust coding

The visual impact of lost packets on the quality of the video delivered to a destination depends clearly on the characteristics of the loss process in the network (and maybe in the destination as well), but it also depends in an essential way on the coding scheme used at the source. With compression algorithms such as H.261 or MPEG, the loss of a single packet might degrade video quality over a large number of frames, specifically until the next intracoded frame is received. Intracoded frames are sent at regular intervals. However, the interval between the receipt of two intracoded blocks of image can be very large in congested networks (since intracoded blocks can themselves be lost with non negligible probability) and/or in low bandwidth networks (since the interval between two such blocks being sent by the source is large). Three approaches can be used to tackle this problem. One approach is to reduce the time between intracoded blocks of image, in the extreme case to a single frame. This approach is used for example in Motion-JPEG. However, it clearly has large bandwidth requirements.

Another approach is to intra-code and transmit only those blocks in a frame that change more than some threshold. This approach referred to as conditional replenishment is used in nv [10] and vic [24]. In vic, blocks from the conditional replenishment stage are transformed using a cosine basis (vic refers to this coding algorithm as "intra-H.261"). In nv, they are transformed using a cosine or Haar wavelet basis.

Yet another approach is to use both intra- and interframe coding. The problem however, as we mentioned earlier, is to achieve good resilience to packet loss. This is done by having the rate at which intracoded frames are sent depend on the loss rate observed in the network. We used this approach in IVS [35]. If the number of destinations is small, a NACK based scheme is used. As the number of destinations increases, and/or as network congestion increases, the scheme in IVS becomes closer to that used in conditional replenishment.

Furthermore, in both nv, vic and IVS coders, a background process also scans and transmits at a low rate all the blocks in the image to ensure that all blocks are eventually transmitted.

The approaches above all adjust the mixture of inter- and intracoded frame to minimize the impact of packet loss on image quality. Another approach is to recover at the destination from packet losses using simple loss concealment techniques such as spatial and temporal interpolation. Spatial interpolation reconstructs a missing piece in a frame from its adjacent (presumably non missing) regions in the same frame. For block transform coders such as the H.261 coder used in IVS, its performance is mediocre because several consecutive lines are damaged when a packet is lost. Temporal interpolation replaces the missing region in a frame with the corresponding region in a previous frame. If motion vectors are available, temporal interpolation replaces the missing region with the region specified by the vectors [13]. Spatial and temporal interpolation can be used together, and combined with interleaving to yield good robustness to loss [44].

Yet another approach which applies to both inter- and intracoded frames uses FEC-based (Forward Error Correction) error control mechanisms. In these mechanisms, redundant information is transmitted along with the original information so that (at least some of) the lost original data can be recovered from the redundant information. These mechanisms are attractive because they provide resilience to loss without increasing latency and will little additional bandwidth requirements. FEC schemes for packet audio have been found to considerably decrease the perceived loss rate at destinations and hence to improve audio quality [5, 14]. Thus, it is reasonable to examine similar schemes for video.

## 3.2 Loss control using FEC

The potential of FEC mechanisms to recover from losses depends on the characteristics of the packet loss process in the network. Clearly, FEC mechanisms are more effective when lost packets are dispersed throughout the stream of packets received at a destination. If losses are very bursty (i.e. if the average number of consecutively lost packets is large), then FEC mechanisms have to be completed by other mechanisms such as those implemented in nv, vic, or IVS and described above. However, analytic modeling and measurements of the loss process of periodic flows in the Internet has shown that most loss periods are short (i.e. the number of consecutively lost packets from the periodic flow is small) even when the overall network load is high [42, 6]. This makes FEC schemes attractive for applications that send periodic streams of packets. These include audio applications, but also video applications such as vic and IVS which smooth the outgoing flow of packets.

A large variety of FEC mechanisms have been proposed in the literature. The simpler mechanisms involve exclusive-OR operations, the idea being to send every $k$th packet a redundant packet obtained by exclusive-ORing the other $k$ packets [32]. Such mechanisms can recover from a single loss in a $k$ packet message. However, they increase the send rate of the source by a factor of $1/k$, and they add latency since $k$ packets have to be received before the lost packet can be reconstructed.

We have developed a FEC scheme in which video packet number $n$ includes redundant information about previous packets $\{n-i\}$. We refer to the maximum value of $i$ as the order of the scheme. For example, packet $n$ might include, in addition to the information it would include in the absence of FEC, additional information about packet $n-1$ in a scheme of order 1, or about both packets $n-1$ and $n-2$ in a scheme of order 2, etc. If packet $n$ is lost, the destination waits for packet $n+1$, decodes the redundant information, and displays the reconstructed information. The amount of redundant information (and hence possibly the order of the scheme) is adjusted over time depending on the measured loss in the network. For example, it might change from including no information (i.e. no redundant information is sent by the source) to including information about packet $n-1$ if the loss rate in the network has been found to increase.

Different kinds of redundant information can be used. We examine here a scheme in which the redundant information about packet $n-k$ included in packet $n$ is made up of the macroblocks (MBs) that were sent in packet $n-k$. However, since the redundant MBs correspond "old" frames, it is reasonable to encode them with a lower definition typically obtained with a coarser quantizer. Furthermore, note that only those MBs encoded in packet $n-k$ but not encoded in packets $n-i$ ($i \in [0,k[$) have to be present in the FEC information in packet $n$. Thus, bandwidth requirements of the FEC scheme are low because a MB encoded in packet $n$ will likely be encoded again in packet $n+1$ (temporal redundancy).

Let $Q_n$ denote the quantizer value for packet $n$. Thus, the error control scheme with redundant MBs from packets $n-1$ and $n-2$ would work as follows

- Mark the MBs that have to be encoded in the new frame using the movement detection algorithm
- Identify duplicate MBs in frames $n$, $n-1$, $n-2$ (i.e. MBs already marked in these frames). Keep only the latest DCT coefficients from these MBs.
- Quantize MBs from frame $n-2$ with quantizer value $Q_{n-2}$ (which we take as mentioned above so that $Q_{n-2} > Q_{n-1}$); do Huffman encoding.
- Quantize MBs from frame $n-1$ with quantizer value $Q_{n-1} > Q_n$; do Huffman encoding.
- Quantize MBs from frame $n$ with quantizer value $Q_n$; do Huffman encoding.

Redundant information from flows $n-1$ and $n-2$ is decoded at the receiver only in case of packet loss.

The redundant information can be added in the flow of packets sent by the source using RTP as specified in [30].

### 3.3 Experimental evaluation

For convenience of exposition, we use the notation ($n$, $\{n-i\}$) to indicate that video packet $n$ includes as redundant information MBs from packets in $\{n-i\}$. To each combination of main and redundant information we can associate a cost (we briefly discussed bandwidth and storage cost above) and a perceived loss rate, which is the loss rate after reconstructing the missing packets using the redundant information. Table 1 shows preliminary measures of perceived loss rates for a few combinations measured from INRIA to UCL and LBL. As expected, adding redundant

| Comb # | Combination | Loss (UCL) | Loss (LBL) |
|---|---|---|---|
| 0 | (n) | 9.8 | 19.4 |
| 1 | (n, n-1) | 1.4 | 7.3 |
| 2 | (n, n-2) | 1.3 | 6.0 |
| 3 | (n, n-1, n-2) | 0.4 | 3.9 |
| 4 | (n, n-1, n-2, n-3) | 0.06 | 2.6 |

Table 1: Perceived loss rate for various FEC schemes

information decreases the perceived loss rate. We note that the last two combinations in the table might be overkills for the INRIA-UCL connection (and in general when network load is low and/or losses are rare). Thus, we need a mechanism to adjust the amount of redundancy added at the source based on the loss process in the network as measured at the destination. We use a feedback mechanism similar to that used for rate control in Section 2, except we now use the feedback information to control the amount of redundant information sent by the source. For simplicity, let us consider the simple case of a unicast connection in which the source can choose only one of the five combinations of main and redundant information shown in Table 1. Let $loss$ denote the loss rate reported by the destination. Then the redundancy control mechanism is as follows

if $loss \geq high\_loss$
    $Comb\# = \min(Comb\# + 1, 4)$
else if $loss < low\_loss$
    $Comb\# = \max(Comb\# - 1, 0)$

Of course, this mechanism is used in conjunction with the rate control mechanism described in the previous section.

Figure 7 shows the results of measurements carried out between a source at INRIA and a destination at UCL. Specifically, it shows the evolutions of the loss rate at UCL before reconstructing the lost packets (plain line) and after reconstructing them (dashed line) versus the sequence number. It also shows the combination number (dotted line) used to encode any given packet. For example, combination 3 is used to encode packet 2000, meaning that packet 2000 includes in addition to its main information, MBs from frames 1999 and 1998. In this experiment, we had $high\_loss$ = 5% and $low\_loss$ = 0.2%. Therefore, we

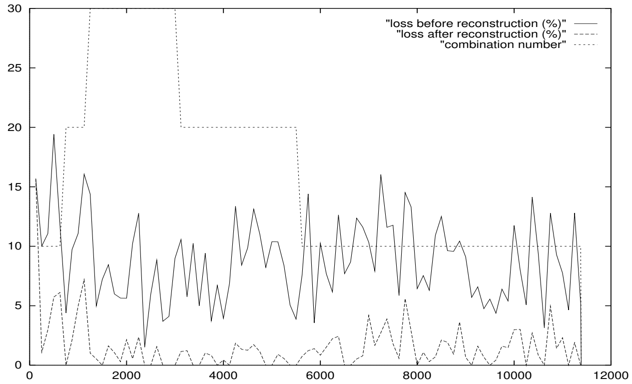


Figure 7: Evolution of the loss rate before and after reconstruction (in %) and of the combination number vs. frame number

expect the loss rate after reconstruction to stay below 5%. This is indeed what we observe in Figure 7.

# 4 Limitations and outlook

The jitter, rate, and error control mechanisms described earlier do prevent video sources from swamping Internet resources and they do help provide controlled degradation of video quality during periods of network congestion. However, they do have limitations as well. We examine some of these limitations next, and we briefly describe how they might be overcome.

The main limitations of our mechanisms are limitations of the rate and error control mechanisms (as well as of the joint rate/error control mechanism), and in fact they are inherent to all source-based control mechanisms for multicast delivery of information in heterogeneous networks. Heterogeneity here means that different branches of a multicast tree might have widely different characteristics such available bandwidth, buffer space, etc. Consider then the case of the rate control scheme. Clearly, the value of the output rate as computed by the control algorithm will impact the level of congestion experienced on the tree. Since the network is heterogeneous, we can expect this level of congestion to be widely different on different branches. With source-based control, the source then chooses its output rate so that a fraction of the branches of the tree are congested. The result is that either low capacity participants of the distribution tree are overwhelmed or high capacity participants receive low quality video.

In practice, it is not clear how to choose an adequate value for this fraction. Ideally, the parts of the multicast tree that experience different levels of congestion should be treated separately. Two possible solutions include using video gateways and using some form of layered coding.

## 4.1 Video gateways

Video gateways take as input a video flow encoded using some scheme, say X, and forward this flow down the multicast tree encoded using some other scheme Y with different (typically lower) bandwidth requirements than X. For example, the RTP-compatible `vgw` video gateway converts video streams to/from Motion-JPEG and H.261 formats [1]. With `vgw`, flows X and Y use different coding schemes. However, it is also possible to use the same coding scheme for X and Y provided different parameters are used. Example parameters for a H.261/MPEG coding scheme include the quantizer value and the movement detection threshold.

The main problem associated with video gateways is that of placing gateways in "appropriate" locations in the network. Some proposed schemes attempt to get around this problem by providing some kind of dynamic gateway placement [29]. Nevertheless, more traditional "static" gateways do provide a solution to the heterogeneity problem, and they have been found useful in a variety of situations that are common in today's MBone [1].

## 4.2 Layered coding

A second solution to the problem of multicast video distribution over heterogeneous networks is to split the video flow generated by the source into multiple flows, each one with different bandwidth requirements than the original flow. The obvious way to do this is to encode the video with a layered or hierarchical coding scheme [12, 33, 26]. In the two layer case, the base flow typically includes the low resolution information, and it can be decoded into a meaningful service. The other flow includes enhancement information. The idea then is to transmit both streams over the non-congested branches of the multicast tree, but to transmit only the base stream over the congested branches. The key observation, made by Deering (and later rediscovered by others), is that this can be achieved even in networks with stateless routers as follows. The source continuously sends all the flows, but it do not exert any kind of rate control on any of them. Receivers then join one or more multicast groups corresponding to different video coding layers depending on the perceived state of congestion in the network. Multicast route pruning is then used so that packets flow over only those links necessary to reach active receivers.

Receiver driven Layered Multicast (RLM) is the first published scheme which described a specific multicast layered transmission scheme and the associated receiver control scheme[25]. RLM uses "probing" experiments similar to those used by TCP to decide when to join and quit layers. Specifically, when a receiver detects congestion, it quits the multicast group corresponding to the highest layer it is receiving at the time (we say that the receiver drops the highest layer); when a receiver detects spare capacity in the network, it joins the multicast group corresponding to the layer next to the highest layer received at the time (we say that the receiver adds the next layer).

The receiver detects network congestion when it observes increasing packet losses. In the absence of loss, the receiver estimates spare capacity, or rather the existence of spare capacity, with so-called join experiments. A join experiment means that a receiver joins the next group and measures the loss rate over an interval referred to as the decision time (to avoid the synchronization of join experiments, experiments are carried out at randomized times). However, the load created by join experiments increases as the size of the multicast group increases. To prevent a load explosion, RLM includes "shared learning", in which a receiver about to start a join experiments multicast its intent to the group. Thus all receivers get to know the result of join experiments carried out by other receivers. To prevent join experiment for different layers to interfere with each others, only receivers that are doing join experiments for layers equal or below a newly advertised experiment can actually conduct their own experiments. RLM with shared learning scales with the size of the group, but it raises a few questions. In particular, it is not clear how effective shared learning is in the absence of knowledge about the structure of the multicast delivery tree. Furthermore, shared learning increases the convergence time to steady state layer subscription especially for receivers with spare capacity (since they will have to wait for slow receivers to reach their steady state before they can join additional upper layers). Also, the times at which layers are added and dropped determine the rate increase and decrease along the branches of the tree. Choosing appropriate values is hard in practice, as the choice has to balance performance and fairness issues.

Regarding performance, we note that receivers should join and quit groups with some care, first because join and quit operations are expensive operations, and second to prevent traffic oscillations. However, finding some kind of "optimal" join/quit interval is an open issue. Actually, it is relatively easy for a receiver to decide to quit a group (i.e. drop a video layer) when the observed loss rate exceeds some threshold. In the absence of active bandwidth probing, it is more difficult to decide when to join a new group.

The fairness issue could be tackled in much the same way as was discussed earlier for the unicast case. Specifically, the idea is to use a RLM-like receiver based rate control scheme, but to replace join experiments by an explicit estimation at each receiver of the bandwidth $\lambda_{equ}$ that would be used by an equivalent TCP connection between the source and that receiver (for example using Equation 1). Assume that each receiver knows the rate $\lambda_i$ of the layer $i$ flow generated by the source over the multicast tree. Then, each receiver executes the following algorithm:

Step 0: Upon joining the group, subscribe to the base layer
Step 1: Compute $\lambda_{equ}$
Step 2: Find $L$, the largest integer such that
$\sum_{i=1}^{L} \lambda_i \leq \lambda_{equ}$. Join the first $L$ groups.

The rate at which data flows between the source and any receiver is equivalent to that of a TCP connection running over the same path. Thus, we have obtained a TCP-friendly scheme suitable for multicast delivery. Furthermore, note that the scheme does not rely on active probing schemes such as join experiments, nor does it require exchange of information between participants as is done with shared learning. However, the limitations and problems associated with TCP friendly schemes, and discussed earlier, remain. Specific mechanisms based on the above scheme are described in [38, 37, 34, 41].

In any case, the use of layered coding schemes allows the development and evaluation of efficient joint source channel coding schemes [11, 21]. The goal then is to make sure that the more important bands are received without error or loss at the destinations. One way to do this in the current single class best effort service Internet is to use FEC schemes to provide unequal error protection proportional to the importance of each band [27, 40]. This amounts in practice to allocating more FEC information to these bands. The problem then is that of a bit allocation problem: how many bits of main and redundant information should be allocated to each band given a model for the network and a distorsion function that takes into account the impact on visual quality of packet loss and of the value of the quantizer used for the band? The design of efficient such algorithms, and more generally of efficient joint source channel coding schemes for the Internet, is an open problem of much interest to the Internet community.

# 5 Conclusion

We have reported on our experience with control mechanisms for packet video in the Internet. Our experiments have shown both the interest as well as the limitations of our scheme. On the minus side, we have seen that it is not completely clear yet how to choose the rate adjustment algorithm, or rather the parameters of the algorithm. Also, source-based control does not scale well to heterogeneous multicast groups (even if group size is small). Finally, it is not clear how to provide some kind of minimum quality to users of video applications in the current Internet.

On the plus side, we have found that control mechanisms provide a reasonably graceful degradation of image quality during periods of network congestion. Furthermore, they are the key to preventing real-time UDP applications from swamping the resources of the Internet, and thus to preventing the so-called congestion collapse which would hurt UDP but also TCP (and thus Web) users. Therefore, it is really unfortunate that so few of the commercial and research tools include such mechanisms. This in fact illustrates what may be their main limitation, namely the lack of a strong incentive to using them in the current single class FIFO Internet.

## Acknowledgements

Much of the work described in the paper was carried out within the MICE and MERCI projects. We would like to thank our colleagues in these projects for many fruitful discussions, with special thanks to Christian Huitema, Mark Handley, Ian Wakeman, and Jon Crowcroft.

# References

[1] E. Amir, S. McCanne, H. Zhang, "An application-level video gateway", *Proc. ACM Multimedia '95*, San Francisco, Nov. 1995.

[2] U. Bilting, A. Sasse, C-D. Schulz, T. Turletti, "Remote seminars through multimedia conferencing: Experiences from the MICE project", *Proc. INET'94*, Prague, June 1994.

[3] J-C. Bolot, T. Turletti, " A rate control for packet video in the Internet", *Proc. IEEE Infocom '94*, Toronto, Canada, pp. 1216-1223.

[4] J-C. Bolot, T. Turletti, I. Wakeman, "Scalable feedback control for multicast video distribution in the Internet", *Proc. ACM Sigcomm'94*, pp. 58-67, Oct. 1994.

[5] J-C. Bolot, A. V. Garcia, "Control mechanisms for packet audio in the Internet", *Proc. IEEE Infocom '96*, San Fransisco, CA, pp. 232-239, Apr. 1996.

[6] J-C. Bolot, A. V. Garcia, "The case for FEC-based error control for packet audio in the Internet", to appear in *ACM Multimedia Systems*.

[7] I. Busse, B. Deffner, H. Schulzrinne, "Dynamic QoS control of multimedia applications based on RTP", *Computer Communications*, vol. 19, no. 1, pp. 49-58, Jan. 1996.

[8] D-M. Chiu, R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks", *Comp. Nets. and ISDN Systs.*, vol. 17, pp. 1-14, June 1989.

[9] L. Cline et al., "DirectShow support for adaptivity in networked multimedia applications", Intel technical report, 1997.

[10] R. Frederick, “Experiences with real-time software video compression”, *Sixth International Workshop on Packet Video*, Portland, Oregon, Sept. 26-27, 1994, pp. F1.1-F1.4.

[11] M.W. Garrett, M. Vetterli, “Joint source/channel coding of statistically multiplexed real time services on packet networks, *ACM/IEEE Trans. Networking*, vol. 1, no. 1, pp. 71-80, Feb. 1993.

[12] H. Gharavi, “Multilayer subband-based video coding”, *IEEE Trans. on Communications*, vol. 39, no 9, pp. 1288-1291, Sept. 1991.

[13] H. Ghanbari, V. Seferidis, “Cell loss concealment in ATM video codecs”, *IEEE Trans. Circuits Systs. Video Tech.*, vol. 3, no. 3, pp. 238-247, June 1993.

[14] V. Hardman, A. Sasse, M. Handley, A. Watson, “Reliable audio for use over the Internet”, *Proc. INET ’95*, Honolulu, HI, pp. 171-178, June 1995.

[15] B. Haskell, “Buffer and channel sharing by several interframe picturephone coders”, *Bell Syst. Tech. J.*, vol. 51, Jan. 1972.

[16] “Video codec for audiovisual services at $p \times 64$ kb/s”, *ITU-T Recommendation H.261*, 1993.

[17] D. Hoffman, M. Speer, G. Fernando, “Network support for dynamically scaled multimedia data”, *Proc. NOSSDAV ’93*, Lancaster, UK, pp. 269-278, Oct. 1993.

[18] V. Jacobson, “Congestion avoidance and control”, *Proc. ACM Sigcomm ‘88*, Stanford, CA, pp. 314-329, Aug. 1988.

[19] K. Jeffay et al., “Adaptive, best-effort delivery of digital audio and video across packet-switched networks”, *Proc. NOSSDAV ’92*, San Diego, CA, Nov. 1992.

[20] H. Kanakia, P. Mishra, A. Reibman, “An adaptive congestion control scheme for real-time packet video transport”, *Proc. ACM Sigcomm ’93*, San Fransisco, CA, pp. 20-31, Sept. 1993.

[21] G. Karlsson, “Layered error-control coding for layered multicast”, SICS Technical report, 1997.

[22] T. V. Lakshman, U. Madhow, “Performance analysis of window-based flow control using TCP/IP”, *Proc. HPN ’94*, Grenoble, France, pp. 133-147, June 1994.

[23] J. Mahdavi, S. Floyd, “TCP-friendly unicast rate-based flow control”, Technical note sent to the end2end-interest mailing list, January 8, 1997.

[24] S. McCanne, V. Jacobson, “vic: a flexible framework for packet video”, *Proc. ACM Multimedia ’95*, San Francisco, CA, Nov. 1995.

[25] S. McCanne, V. Jacobson, M. Vetterli, “Receiver-driven layered multicast”, Proc. ACM Sigcomm ’96, Stanford University, CA, pp. 117-130, Sept. 1996.

[26] S. McCanne, M. Vetterli, “Low complexity video coding for receiver driven layered multicast”, *IEEE JSAC*, vol. 16, no. 6, pp. 983-1001, Aug. 1997.

[27] J. W. Modestino et al., “Combined source-channel coding of images using the block cosine transform”, *IEEE Trans. Comm.*, vol. 29, pp. 1261-1274, Sept. 1981.

[28] A. Ortega, M. Khansari, “Rate control for video coding over variable bit rate channels with application to wireless transmission”, *Proc. ICIP ’95*, Washington DC, Oct. 1995.

[29] J. Pasquale et al., “Filter propagation in dissemination trees: Trading off bandwidth and processing in continuous media networks”, *Proc. NOSSDAV ’93*, Lancaster, UK, pp. 269-278, Oct. 1993.

[30] C. Perkins et al., “Payload Format Issues for Redundant Encodings in RTP”, internet-draft, July 1996.

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

[32] N. Shacham, P. McKenney, “Packet recovery in high-speed networks using coding and buffer management”, *Proc. IEEE Infocom ’90*, San Fransisco, CA, pp. 124-131, May 1990.

[33] N. Shacham, “Multipoint communication by hierarchically encoded data”, *Proc. IEEE Infocom ’92*, Florence, Italy, pp. 2107-2114, May 1992.

[34] D. Sisalem, H. Schulzrinne, “End to end quality of service control using adaptive applications”, *Proc. 5th Intl. Wksp on QoS (IWQPS’97)*, New York, NY, May 1997.

[35] T. Turletti, C. Huitema, “IVS Videoconferencing in the Internet”, *IEEE/ACM Trans. Networking*, Vol. 4, No 3, June 1996, pp.340-351.

[36] T. Turletti, C. Huitema, “RTP Payload Format for H.261 Video Streams”, RFC 2032, Oct. 1996.

[37] T. Turletti, S. Fosse Parisis, J.C., “Experiments with a layered transmission scheme over the Internet”, INRIA report RR3296, 1997.

[38] L. Vicisano, L. Rizzo, J. Crowcroft, “TCP-like congestion control for layered multicast data transfer”, UCL research note RN/97/75, 1997.

[39] I. Wakeman, “Packetized video - Options for interaction between the user, the network, and the codec”, *The Computer Journal*, vol. 36, no. 1, 1993.

[40] P. H. Westerink et al., “Adaptive channel error protection of subband encoded images”, *IEEE Trans. Comm.*, vol. 41, no. 3, pp. 454-459, March 1993.

[41] L. Wu, R. Sharma, B. Smith, “Thin streams: an architecture for multicasting layered video”, *Proc. NOSSDAV 97*, St. Louis, MO, May 1997.

[42] M. Yajnik, J. Kurose, D. Towsley, “Packet loss correlation in the MBone multicast network,” *IEEE Global Internet Conf.*, London, UK, Nov. 1996.

[43] J. Zdepsky et al., “Statistically based buffer control policies for constant rate transmission of compressed digital video”, *IEEE Trans. Comm.*, vol. 39, pp. 947-957, June 1991.

[44] Q-F. Zhu et al., “Coding and cell loss recovery in DCT-based packet video”, *IEEE Trans. Circuits Systs. Video Tech.*, vol. 3, no. 3, pp. 248-258, June 1993.