A rate control mechanism for packet video in the Internet

Jean-Chrysostome Bolot Thierry Turletti
INRIA
B. P. 93
06902 Sophia-Antipolis Cedex
France
{bolot, turletti}@sophia.inria.fr

Abstract

Datagram networks such as the Internet do not provide guaranteed resources such as bandwidth or guaranteed performance measures such as maximum delay. One way to support packet video in these networks is to use feedback mechanisms that adapt the output rate of video coders based on the state of the network. In this paper, we present one such mechanism. We describe the feedback information, and how it is used by the coder control algorithm. We also examine how the need to operate in a multicast environment impacts the design of the control mechanism.

Our mechanism has been implemented in the H.261 video coder of IVS. IVS is a videoconference system for the Internet developed at INRIA. Experiments indicate that the control mechanism is well suited to the Internet environment. In particular, it makes it possible to establish and maintain quality videoconferences even across congested connections in the Internet. Furthermore, it prevents video sources from swamping the resources of the Internet, which could lead to unacceptable service to all users of the network.

1 Introduction

Data networks such as the Internet use datagram switching as a means of dynamically allocating network resources on a demand basis. Datagram switching facilitates the interconnection of networks with different architectures and provides flexible resource allocation and good reliability against node and link failure. However, datagram switching coupled with the FCFS discipline typically used in current switches make it essentially impossible to provide the guarantees, generally expressed in terms of minimum bandwidth or maximum delay, required by the so-called real-time applications such as videoconferences [16]. Two distinct approaches have emerged recently to tackle this problem.

One approach is to extend current protocols and switch scheduling disciplines to provide the desired performance guarantees. This approach requires that admission control, policing, reservation, and/or sophisticated scheduling mechanisms be implemented in the network. The design, analysis, and evaluation of such mechanisms is an active research area (e.g. [3, 16, 18]). These mechanisms will eventually be implemented in the Internet, but they are not expected to be available in the very near future. An example videoconference system based on this approach is described in [5]. Another approach is to control the rate at which packets can be sent over a connection, the objective being to limit this rate to the capacity of the connection. We observed earlier that this capacity changes with time. Therefore, the control mechanism must be somehow informed of such changes. One way is for sources of packets to receive feedback about the state of the network and to control the rate at which packets are sent into the network accordingly.

We propose to use this second approach to control sources of real-time traffic. In this paper, we consider specifically sources of video traffic, i.e. video coders. The goal then is to use feedback information about the state of the network to adapt the output rate of the coder.

This approach has two important advantages. The first advantage is that it is in fact surprisingly easy to control the output rate of a software video coder by adjusting parameters inside the coder. Specific ways to do this will be described in Section 2.

The second advantage of this approach is that it does not require special support from the network such as admission control, resource allocation, etc. Therefore, it can easily be implemented in the current Internet. This is very important because the increased computing power of workstations and the availability of multicast transmission [6] as well as audio and video applications such as VAT [12], NV [8], NEVOT [23], and IVS has led to a huge increase of real-time traffic in the Internet. For example, IETF meetings, conferences (e.g. the 4th European Networking Conference), and seminars (e.g the Xerox PARC Thursday seminars) are now regularly broadcast. The uncontrolled transmission of audio and video streams could easily swamp the resources of the Internet and lead to unacceptable service for all users of the network. Including control mechanisms in real-time applications helps prevent such situations from happening.

Feedback control mechanisms are already used in the Internet to control sources of non real-time traffic. The best example is TCP, where the feedback information is packet losses detected by timeouts or multiple acknowledgements at the source, and the control scheme is Van Jacobson's dynamic window scheme [11]. An important goal of TCP and other mechanisms for data traffic is to maximize throughput and minimize packet delay or loss, or equivalently to optimize some function of throughput and delay such as power [15]. However, we note that this goal does not necessarily translate into optimizing the quality of a video transmission. Therefore, we do not expect that the mechanisms proposed for the control of data traffic will be suitable for video traffic.

The idea of controlling sources of video traffic to adapt to the capacity available in a network is not new. Video has conventionally been transmitted over specific networks which provide connections with constant or nearly constant capacity channels (e.g. telephone or CATV networks). However, the rate of a video sequence can vary rapidly with time due to the effects of scene complexity and motion. The problem, therefore, is to obtain from the variable rate sequence a constant rate stream of data that can be sent into the network. This is typically done as shown in Figure 1. The variable rate video stream is sent into a buffer which is drained at a constant rate. The amount of data in the buffer is used as a feedback information by a controller which adapts the parameters of the coder, and hence the output rate of the coder, in order to prevent buffer overflow or underflow (e.g. [2]).

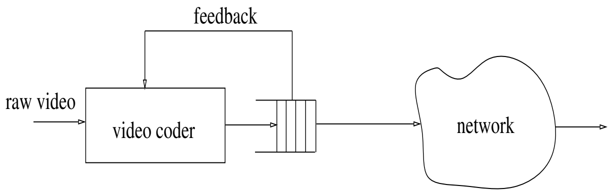

Figure 1: Feedback control for networks with fixed capacity channels

Feedback mechanisms for video sources have also been proposed for networks with variable capacity channels such as the Internet. There, the goal is to adjust the parameters (and hence the output rate) of video coders based on feedback information about changing network conditions, i.e. changing capacity in the network (Figure 2). References [9, 25] propose

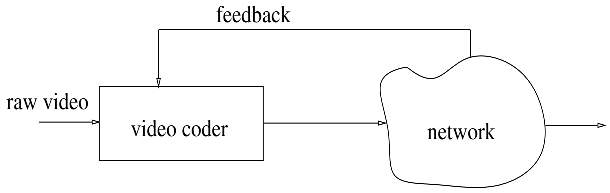

Figure 2: Feedback control for networks with variable capacity channels

to use feedback control, but they do not describe specific control mechanisms. Reference [26] describes a specific scheme. However, this scheme requires that the source of a connection know the service rate of the bottleneck on this connection. This service rate can be estimated in networks where the switches use a so-called Fair Queueing or equivalent discipline [18]. However, it is not available in networks with FCFS switches such as the Internet. Reference [14] describes a feedback control scheme which requires that switches send their buffer occupancies and service rates back to the source. Reference [13] describes a mechanism in which the time at which video packets are sent (and hence the rate at which the image is refreshed at the destination) is controlled, rather than the coding process of the video frames (and hence the quality of the image received at the destination). None of the above references describes how their mechanisms would operate in a multicast environment.

Feedback mechanisms have also been used in so-called continuous media players. The goal there is to control the delivery of audio and/or video streams from a multimedia server to client workstations. However, these mechanisms are designed for local or metropolitan area networks with zero or negligible delay jitter [19, 20].

Our contributions

In this paper, we describe a feedback control mechanism for variable bit rate video in which the parameters of a video coder, and hence the output rate of the coder, are adjusted in response to changing conditions in the Internet. We describe the feedback information used by the controller, and the control algorithm proper. We also examine how the need to operate in a multicast environment impacts the design of the mechanism. Our control mechanism has been implemented in the H.261 coder of IVS. IVS is a "all-software" videoconference system[1] for the Internet developed at INRIA. IVS uses IP multicast, UDP and RTP [6]. It is being used over the Internet, to hold videoconferences, to multicast conferences (e.g. the 4th Joint European Networking conference held in Trondheim, Norway, in May 1993), etc. We find that the control mechanism is well suited to the Internet environment. In particular, it makes it possible to establish and maintain videoconferences of reasonable quality even across congested connections in the Internet.

The rest of the paper is organized as follows. In Section 2, we describe ways to control the output rate of video coders, and specifically of H.261 coders. In Section 3, we describe the feedback rate control mechanism used in IVS, and we evaluate its performance. Section 4 concludes the paper.

[1] However, IVS is compatible with hardware coders [24].

2 Output rate control in video coders

Many algorithms have been proposed for the coding and transmission of video traffic. Some of them have been standardized, such as the ISO JPEG [27] for single frame images, the ISO MPEG [17] for moving images, and the CCITT H.261 [10] for moving images sent over an integrated services digital network (ISDN).

The MPEG and H.261 standards include somewhat similar coding algorithms. The structure of these algorithms is shown in Figure 3. A picture is divided

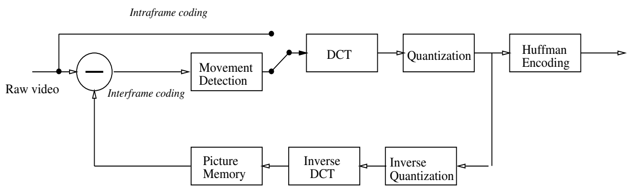

Figure 3: Structure of a H.261 coder

into blocks of 8×8 pixels. A discrete cosine transform (DCT) converts the blocks of pixels into blocks of frequency coefficients. These coefficients are quantized and then encoded using a Huffman encoding technique. In addition, images can be coded using intraframe or interframe coding. The former encodes each picture in isolation. The later encodes a picture by predicting it from other pictures in the sequence of pictures. In H.261, the prediction is made using the previous picture in the sequence. In MPEG, the prediction is made using the previous and/or the next picture in the sequence. In this paper, we consider more specifically video coders based on the H.261 standard.

Many parameters of a H.261 coder can be adjusted to change the output rate of the coder [10]. In this paper, we consider three such parameters, specifically the refresh rate, the quantizer, and the movement detection threshold.

The *refresh rate* characterizes the fraction of the frames arriving at the coder from the camera which are actually encoded and sent over the network. Clearly, decreasing the refresh rate decreases the average output rate of the coder. However, it also decreases the frame rate observed at the receiver(s), and hence the quality of the received image sequence.

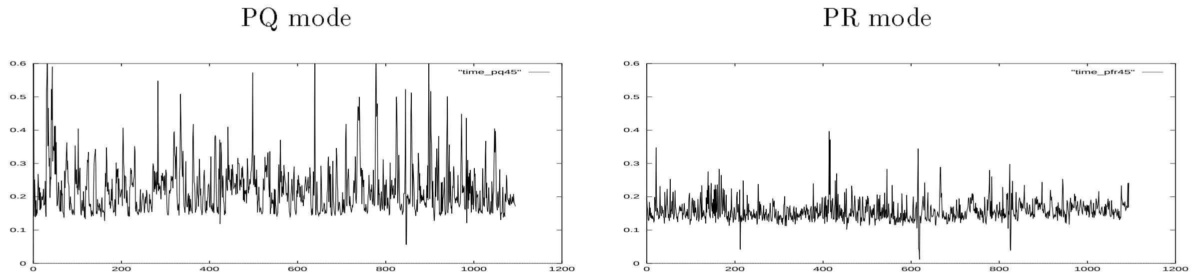

Figure 4: Evolutions of the interframe time (in s) versus the frame number in the PQ and PR modes

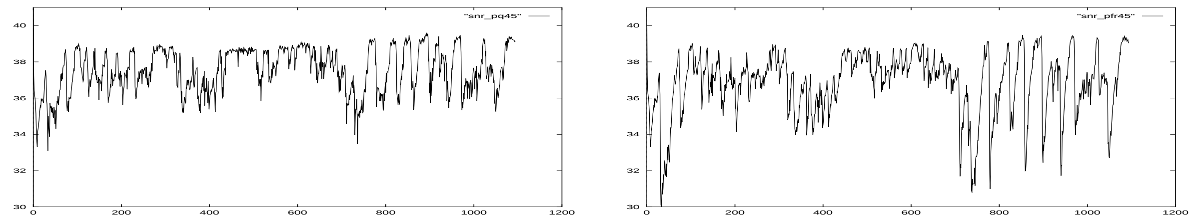

Figure 5: Evolutions of the SNR (in dB) versus the frame number in the PQ and PR modes

The *quantizer* characterizes the accuracy with which the coefficients obtained from the discrete cosine transformation are encoded. Increasing the quantizer is equivalent to encoding the frequency coefficients more coarsely, and thus reducing the quality of the transmitted image. However, it is also equivalent to reducing the number of bits used to encode pixels, and thus reducing the output rate of the coder.

The *movement detection threshold* characterizes the number of blocks encoded in each image. Recall that in interframe coding, the DCT is applied on a frame-to-frame difference signal. The movement detection threshold controls the selection of the blocks which are "sufficiently different" from the previous frame. If the threshold value increases, then the number of blocks which are considered to have changed since the last frame decreases. Therefore, the number of bytes required to encode each image, and hence the output rate of the coder, decreases. However, increasing the threshold decreases the sensitivity of the coder to movement and yields lower image quality.

The specific requirements of a video application will dictate which of the three parameters described above should be modified when adjusting the output rate of a coder. The refresh rate is modified if the precise 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 specifies the maximum rate at which the video stream can leave the coder, which we denote by *max_rate*, and a mode. 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 refresh rate is adjusted. The values of the quantizer and the movement detection threshold are fixed. The coder then waits for a sufficient amount of time before encoding the next image in a sequence so that the output rate stays below *max_rate*. We refer to this amount of time as the interframe time. In the *Privilege Rate* mode (PR mode), only the quantizer and the movement detection threshold are adjusted.

The figures above illustrate the impact on image quality created by changing coding parameters. These figures were obtained using a pre-recorded sequence of 200 frames. In all cases, the value of *max_rate* is set to 45 kb/s. Figure 4 shows the evolutions of the interframe time (i.e. the inverse of the frame rate, expressed in seconds) as a function of the frame number in both PQ and PR modes. Figure 5 shows the evolutions of the signal to noise ratio (SNR, expressed in dB) as a function of the frame number in both modes. As expected, the SNR is higher in PQ mode than in PR mode. However, the rate at which frames are sent is much higher on average in the PR mode.

3 A feedback control scheme for the Internet

In Section 2, we described how the parameters of the H.261 codec in IVS can be adjusted to control the output rate of the coder. In this section, we describe a mechanism in which the parameters, and hence the output rate, of the coder are controlled based on feedback information about the state of the network. The general structure of the controlled coder is similar to that shown in Figure 2. The specific structure of the coder is shown in Figure 6.

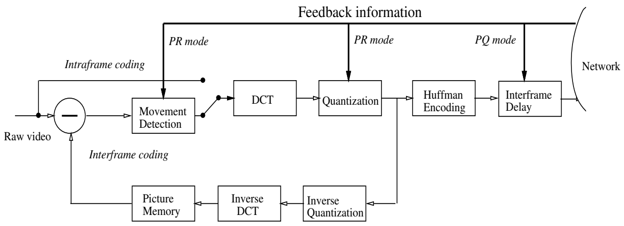

Figure 6: Feedback control of a H.261 coder

A feedback control scheme is characterized by the feedback information available about the state of the network, and by the specific control mechanism used by the video coder. We describe both of these next.

The feedback information

A fundamental characteristic of any feedback mechanism is the amount of feedback information available at the source. The Internet infrastructure typically does not provide sources of traffic with explicit feedback information about the state of the network (e.g. queue occupancies at the switches). The only easily available information is implicit information such as measures of losses and/or roundtrip delays. Some control schemes for video traffic described in the literature intend to use packet delays as feedback information. However, previous experiments and simulations have shown that in the Internet, the throughput of connections using delay-based feedback is lower than the throughput of connections such as TCP connections using loss-based feedback (e.g. [4]). In other words, delay-based schemes "loose" bandwidth when they compete with TCP. This led us to choose a feedback information in our scheme which is based on packet losses measured at the receivers[2]. In IVS, a packet loss is detected using the RTP sequence number [22]. Specifically, up to 4 packets are buffered at a receiver. If packets numbered $n+1$, $n+2$, and $n+3$ have been received, and packet n is still missing, then packet n is considered to be lost.

In order to monitor how many video packets arrive at their destinations, the source should obtain information from each receiver indicating packet loss on the path from the source to that receiver. One possible approach is to let each receiver send a negative acknowledgement (NACK) packet whenever it detects the loss of a packet. However, a potentially large number of NACK packets will be sent in a "large multicast" environment, i.e. when the number of receivers is large. Furthermore, the number of NACK packets will increase as network congestion increases, and hence further increase congestion. This is referred to as the NACK explosion problem. Our approach to this problem in IVS is as follows.

In a "small multicast" environment, i.e. when the number of receivers is below some threshold[3], receivers send a negative acknowledgement (NACK) back to the source whenever they detect the loss of a packet. In IVS, we set the threshold for the number of receivers to be equal to 10. In a "large multicast" environment, i.e. when the number of receivers exceeds 10, receivers do not send NACKs. Instead, they periodically send back to the source a quality of service (QoS) measure which is the average packet loss rate they observed during a time interval of length T. In the current implementation of IVS, we take T to be the time required at a receiver to receive 100 packets. As suggested in the RTP draft document [22], we also make sure that each receiver send feedback information at least once every 2 minutes.

We note that in most cases, receivers will send feedback information much less frequently than if they sent a NACK whenever they detect the loss of a packet. Therefore, sending periodic QoS measures will alleviate the NACK explosion problem. To further decrease the impact of feedback traffic on the network, a

[2]Note that this criterion does not take into account possible random losses [21] which would be mistakenly considered as indicative of network congestion.

[3]The source uses the Real-Time Control Protocol [22] implemented in IVS to determine the number of receivers in a multicast session.

receiver sends its QoS measure only with some probability [28]. This probability is chosen so that the load of the feedback traffic is approximately constant. One way to do this is to set the probability to be inversely proportional to the total number of receivers.

Each receiver sends its measured average loss rate back to the source in the QoS field in the RTP header [22]. The source then converts the different measures of QoS into a "global" measure characterizing how well packets arrive at the receivers. One approach would be to take some average of the loss rates measured by each receiver. However, this does not seem adequate in a network where links can have widely different bandwidths, and where loss rates can be widely different on different branches of the multicast tree. Another approach would be to take a percentile QoS, i.e. a percentile of the loss rates measured at the receivers. Special cases of this would be to take the highest, or the median loss rate. In the current implementation of IVS, we use the median loss rate.

All the above approaches are *ad hoc*, and they lead to situations where a (possibly large) fraction of the receivers experience high loss rates, yet the source does not decrease its max_rate. This suggests that more elaborate criteria be used by the source. Ideally, the source should be able to single out those parts of the multicast tree that experience high loss rates (i.e. congestion), and to adapt its output rate accordingly. One way to do this is to use sub-band coding. Unfortunately, the H.261 standard is not designed to support sub-band coding (However, it is possible to mimic sub-band coding in H.261, for example with multiple passes of the coding algorithms done with different values of the quantizer). Nevertheless, this approach is promising and we are investigating it.

The control algorithm

Control actions are taken by the coder at discrete points in time, specifically whenever a sequence of 100 packets has been encoded and sent. Of course, the number 100 is chosen so that the interval between successive controls is approximately equal to the interval over which the feedback information is computed at the receivers.

During a control action, the control algorithm adjusts the maximum output rate of the coder max_rate so that the median loss rate stays below a tolerable loss rate. Denote the median loss rate by med_loss, and the tolerable loss rate by tol_loss. Specifically, max_rate is decreased by half if the median loss rate is larger than tol_loss. Otherwise, it is increased by a fixed fraction of its current value. We also make sure that the output rate is always larger than some minimum rate to guarantee a minimum quality of the videoconference at the receivers. Thus, the control algorithm is as follows:

If $med\_loss > tol\_loss$

$$max\_rate = \max(max\_rate/2, min\_rate)$$

else

$$max\_rate = gain * max\_rate$$

In IVS, we set $gain = 1.5$ and $tol\_loss = 10\%$. The maximum value of max_rate and the value of min_rate can be specified by the user. In the experiments described below, we set these values to 100 kb/s and 10 kb/s, respectively.

Evaluating the control scheme

Next, we present measurements and results from experiments carried out with the feedback-controlled coder of IVS over the MBone[4].

In these experiments, the video source is an IVS source at INRIA. The number of receivers is such that the environment is a "large multicast" environment, i.e. receivers send QoS packets periodically back to the source. We analyze in the multicast tree the connection between INRIA (the IVS source) and University College London (the receiver). Figure 7 shows the evolutions of the maximum output rate max_rate at the source (plain line) and the average packet loss computed at the receiver (dashed line). The unit on the x-axis is the frame number. Figure 8 shows the same evolutions measured at a different time of the day. As expected, we observe that the value of max_rate at the source decreases as losses are detected at the receiver. Therefore, the bandwidth requirements of the source decreases when network congestion increases. In Figure 8, we see that the receiver detected a sustained loss

[4]The MBone, or Multicast Backbone, is a virtual network running on top of the IP layer in the Internet. The MBone is technically a network of hosts running a multicast IP daemon. Functionally, the MBone provides a multicasting facility in the Internet [6].

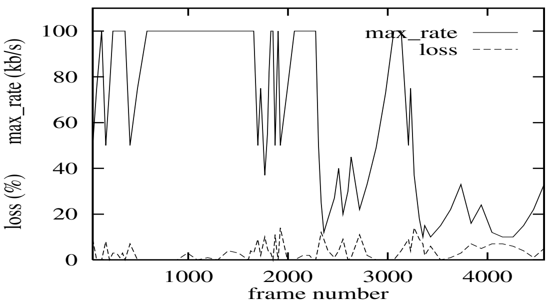

Figure 7: Evolutions of *max_rate* (in kb/s) and the loss rate at the receiver (in %).

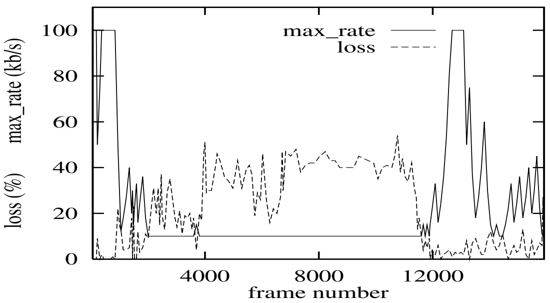

Figure 8: Evolutions of *max_rate* (in kb/s) and the loss rate at the receiver (in %).

rate of 40% for a very long period of time (the time elapsed between receiving the frames number 2000 and 11800 is approximately 30 minutes). This high loss rate was created by another videoconference multicast from North Carolina which caused congestion in the MBone. In the event of long term congestion, *max_rate* settles to its minimum value of 10 kb/s as expected. These results clearly show that the control mechanism prevents video sources from swamping the resources of the Internet.

The path from INRIA (in France) to UCL (in the UK) on the MBone is very long. In fact, the packets are routed through the US before they reach UCL. Therefore, it is not clear whether the feedback information sent back by the receiver at UCL is still relevant, i.e. whether it still reflects the state of the network, by the time it reaches the sources at INRIA. This problem is a very important problem in feedback flow control when the feedback delay is high [7].

Recall that receivers compute a loss rate averaged over some time interval of length T. Assume that a receiver sends a QoS packet at time nT and that this packet is received at the source at time $nT + \delta$. Intuitively, the feedback information sent at time nT is still relevant when it reaches INRIA if the loss rate computed by the receiver around time $nT + \delta$ is close (i.e. correlated) with that computed at time T. This can be expressed in mathematical terms as follows: the autocorrelation of computed loss rates decreases slowly compared to the mean feedback delay, i.e. the mean delay between the receiver and the source.

Figure 9 shows the autocorrelation of loss rates corresponding to the traces shown in Figure 7 at the receiver. The lag in the x-axis is expressed in units of multiples of T. It turns out that $T \approx 15$ s, whereas the mean delay between UCL and INRIA via the MBone is approximatively one second. We see in Figure 9 that the autocorrelation stays above 0.4 for lags up to 40T, i.e. for much longer than a few feedback delays. This indicates that the feedback information still accurately represents the state of the network by the time it reaches the controller. Figure 10 shows the autocor-

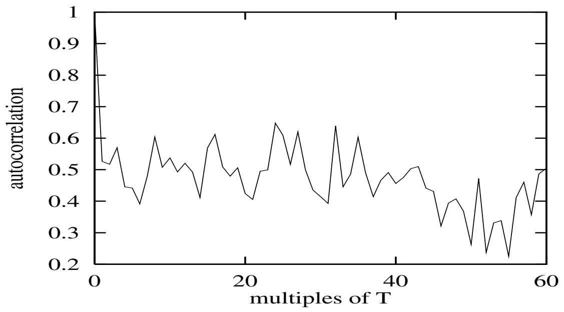

Figure 9: Autocorrelation of loss rates

relation of the loss rates corresponding to the traces shown in Figure 8 at the receiver. The autocorrelation

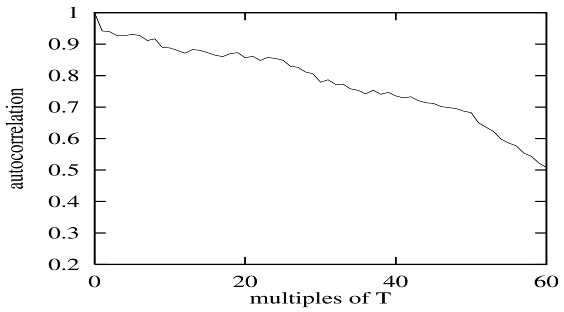

Figure 10: Autocorrelation of loss rates

curve decreases even slower than that of Figure 9. We conclude that it is reasonable to use feedback control for video coders over the MBone.

4 Conclusion

Our work demonstrates that a simple feedback control mechanism can be used to prevent video sources from swamping the resources of the Internet. Furthermore, this mechanism makes it possible to establish and maintain videoconferences with reasonable quality even across congested links without requiring special support from the network such as admission control, resource reservation, etc.

We are extending this work in several directions. One direction is to investigate layered coding algorithms (refer to Section 3). Another direction is to quantify the tradeoff between the bandwidth gained by reducing the source send rate, and the lower quality of the video delivered to the receivers as a consequence of the lower send rate. The main problem is to find an objective measure of the quality of the video delivered to the receivers. We are considering a measure that combines the SNR, the loss rate, and the frame rate. Preliminary results indicate that controlling a video source decreases network congestion (since it decreases the bandwidth requirements of the source) yet essentially preserves the quality of the video flow delivered to the destinations. We are currently investigating these issues.

The feedback control mechanism will be included in the next release of IVS[5].

References

[1] S. Casner, "First IETF Internet audiocast", *Computer Communication Review*, vol. 22, no. 3, pp. 92-97, July 1992.

[2] C-T. Chen, A. Wong, "A self-governing rate buffer control strategy for pseudoconstant bit rate video coding", *IEEE Trans. Image Processing*, vol. 2, no. 1, pp. 50-59, Jan. 1993.

[3] D. D. Clark, S. Shenker, L. Zhang, "Supporting real-time applications in an integrated services packet network: Architecture and mechanism", *Proc. ACM Sigcomm '92*, Baltimore, MD, pp. 14-26, Aug. 1992.

[4] W. Dabbous, "Analysis of a delay-based congestion avoidance algorithm", *Proc. 4th IFIP Conf. on High Performance Networking*, Liege, Belgium, Dec. 1992.

[5] C. Elliott, "High quality multimedia conferencing through a long-haul packet network", *Proc. ACM Multimedia '93*, Los Angeles, CA, pp. 91-98, Aug. 1993

[6] H. Eriksson, "MBone - The multicast backbone", *Proc. INET '93*, San Fransisco, CA, Aug. 1993.

[7] K. Fendick, M. Rodrigez, A. Weiss, "Analysis of a rate-based control strategy with delayed feedback", *Proc. ACM Sigcomm '92*, Baltimore, MD, pp. 136-148, Aug. 1992.

[8] R. Frederick, "nv", Manual Pages, Xerox Palo Alto Research Center.

[9] M. Gilge, R. Gusella, "Motion video coding for packet-switching networks - An integrated approach", *Proc. SPIE Conference on Visual Communications and Image Processing*, Boston, MA, Nov. 1991.

[10] Recommendation H.261: Video codec for audiovisual services at p*64 kb/s, CCITT White Book, 1990.

[11] V. Jacobson, "Congestion avoidance and control", *Proc. ACM Sigcomm '88*, Stanford, CA, pp. 314-329, August 1988.

[12] V. Jacobson, S. MacCanne, "vat", Manual Pages, Lawrence Laboratory, University of California, Berkeley, CA.

[13] K. Jeffay, D. L. Stone, T. Talley, F. D. Smith, "Adaptive, best-effort delivery of digital audio and video across packet-switched networks", *Proc. 3rd Intl. Wkshp on Network and OS Support for Digital Audio and Video*, San Diego, CA, Nov. 1992.

[14] 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.

[15] L. Kleinrock, "Power and deterministic rules of thumb for probabilistic problems in computer communications", *Proc. ICC '79*, vol. 3, pp. 43.1.1-10, April 1979.

[16] J. Kurose, "Open issues and challenges in providing QoS guarantees in high speed networks", *Comp. Comm. Review*, vol. 23, no. 1, pp. 6-15, Jan. 1993.

[17] D. J. LeGall, "MPEG: a video compression standard for multimedia applications", *CACM*, vol. 34, no. 4, pp. 47-58, April 1991.

[18] A. Parekh, R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks", *Proc. IEEE Infocom '93*, San Fransisco, CA, pp. 521-530, March 1993.

[19] L. A. Rowe, B. C. Smith, "A continuous media player", *Proc. 3rd Intl. Wkshp on Network and OS Support for Digital Audio and Video*, San Diego, CA, Nov. 1992.

[20] S. Ramanathan, P. V. Rangan, "Adaptive feedback techniques for synchronized media retrieval over integrated networks", *IEEE/ACM Trans. Networking*, vol. 1, no. 1, Jan. 1993.

[21] D. Sanghi, A. K. Agrawala, B. Jain, "Experimental assessment of end-to-end behavior on Internet", *Proc. IEEE Infocom '93*, San Fransisco, CA, pp. 867-874, March 1993.

[22] H. Schulzrinne, "Issues in designing a transport protocol for audio and video conferences and other multiparticipant real-time applications", Internet draft, Audio-video transport working group, March 1993.

[23] H. Schulzrinne, "Voice Communication Across the Internet : A Network Voice Terminal", Research Report, Dept. of Electrical Engineering, University of Massachussets at Amherst, July 92.

[24] T. Turletti, "H.261 software codec for videoconferencing over the Internet", INRIA Research Report 1834, Jan. 1993.

[25] I. Wakeman, "A combined admission and congestion control scheme for variable bit rate video", UCL Technical Report, June 1993.

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

[27] G. Wallace, "Overview of the JPEG still image compression standard", *Proceedings of the SPIE*, vol. 1244, pp. 220-233, Feb. 1990.

[28] R. Yavatkar, L. Manoj, "Optimistic strategies for large-scale dissemination of multimedia information", *Proc. ACM Multimedia '93*, Anaheim, CA, pp. 1-8, Aug. 1993.

[5] The latest release of IVS is available by anonymous FTP from zenon.inria.fr in the directory rodeo/ivs. The current release is IVS 3.2.