Scalable Feedback Control for Multicast Video Distribution in the Internet

Jean-Chrysostome Bolot[1] Thierry Turletti[1] Ian Wakeman[2*]

[1] INRIA, B.P. 93, 06902 Sophia Antipolis Cedex, France
{bolot, turletti}@sophia.inria.fr

[2] University College London, Gower Street, London WC1E 6BT, UK
i.wakeman@cs.ucl.ac.uk

Abstract

We describe a mechanism for scalable control of multicast continuous media streams. The mechanism uses a novel probing mechanism to solicit feedback information in a scalable manner and to estimate the number of receivers. In addition, it separates the congestion signal from the congestion control algorithm, so as to cope with heterogeneous networks.

This mechanism has been implemented in the IVS videoconference system using options within RTP to elicit information about the quality of the video delivered to the receivers. The H.261 coder of IVS then uses this information to adjust its output rate, the goal being to maximize the perceptual quality of the image received at the destinations while minimizing the bandwidth used by the video transmission. We find that our prototype control mechanism is well suited to the Internet environment. Furthermore, it prevents video sources from creating congestion in the Internet. Experiments are underway to investigate how the scalable probing mechanism can be used to facilitate multicast video distribution to large numbers of participants.

1 Introduction

Multicast packet transmission has now been available in the Internet for some time [9]. The examination of how it is being used brings one to the conclusion that multicast distribution is not being widely used for the traditional bulk transfer applications of computer communications, but rather for the distribution of so-called real-time traffic, i.e. packet video and voice, along with resource location querying. Multicast groups for voice and video distribution currently have up to hundreds of recipients attached to them (although this number is expected to increase, for such applications as TV distribution), for which the end consumers of the data are people.

Ergonomic studies and anecdotal evidence from the Internet demonstrate that people can use audio or video signals as long as the information content is above some minimum level which depends on the task in hand (e.g. [30, 2]). Thus, it is possible to transmit audio and video signals with lower bandwidth requirements at the expense of a slight degradation in user satisfaction, although user task performance will not be significantly degraded until the information content in the signal slips below the minimum level mentioned above. One approach to distributing real-time streams is then to adjust the bandwidth, or rate, of a source based on the prevailing conditions in the network. These conditions change with time because connections are set up and terminated, and because sources do not send at a constant rate. The rate adjustment mechanism must be of course informed of such changes. One way is for the source 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 approach to control sources of real-time traffic. In this paper, we consider specifically sources of video traffic, i.e. video coders. Through the use of the mechanisms described here, we can prevent congestion of the Internet. This is important because the increased computing power of workstations and the availability of audio and video applications such as VAT [17], NV [11], NEVOT [26], and IVS [28] has led to a huge increase in the real-time traffic in the Internet. IETF meetings [4], seminars (e.g. the MICE [2] and Xerox seminars), shuttle launches, etc., are now regularly audio- and video-cast. The uncontrolled transmission of audio and video streams would easily (it already

* Authors in alphabetical order

has done so on a few occasions) swamp the resources of the Internet, cause congestion, and lead to unacceptable service for all users of the network.

We note that our approach does not require special support from the network such resource allocation. This is in contrast to another approach to delivering real-time streams, which has focussed on changing the network architecture to meet the expected bandwidth and delay requirements of audio/video applications by introducing new admission control and switch scheduling mechanisms (e.g. [7, 21]). These mechanisms may eventually be implemented if the Internet moves to a resource reservation model, but they are not expected to be available in the very near future.

Our proposed feedback-based approach is already used in the Internet to control sources of data traffic [16]. However, the use of wide area multicast for the delivery of the real time streams creates additional problems in getting feedback from the receivers. It is important to get timely notification of congestion, but if the congestion is close to the source then all receivers will detect the congestion and will send a notification to the source generating an implosion of messages at the source [8, 31]. To prevent this, we require a mechanism for soliciting feedback information in a scalable way from the receivers.

Given this feedback information, it is important to relate the state to the entire group of receivers within the context of the application. If only a single receiver is suffering congestion on its last hop in the delivery tree, should the transmitter degrade the video for the entire group? Or should it request that the troubled receiver leave the video group, and rely on some other information stream? Thus mechanisms are needed to estimate the number of receivers suffering, and for the application to use this information in deciding how to adjust its output rate.

Our contributions

We describe a scalable feedback mechanism, i.e. a scalable method for eliciting information from the receivers in a multicast transmission. With our method, a source of real-time traffic can estimate the number of receivers and control the load generated by the feedback traffic. The method combines a probabilistic polling mechanism with increasing search scope and a randomly delayed reply scheme. The feedback obtained is then used to adjust the parameters of the source codec to control the output rate of the codec. The mechanisms have been implemented in the H.261 coder of IVS. IVS is a software[1] videoconference system for the Internet developed at INRIA. IVS uses IP multicast, UDP and RTP [25]. It is being used over the Internet to hold videoconferences, to multicast seminars, etc. We have found that the scalable control mechanism is well suited to the Internet environment. It makes it possible to establish and maintain videoconferences of reasonable quality even across congested connections in the Internet.

[1]However, IVS is compatible with hardware codecs [28].

The rest of the paper is organized as follows. In Section 2, we discuss issues related to congestion control for real-time traffic in a multicast environment. In Section 3, we describe the scalable feedback mechanism. In Section 4, we describe the video control mechanism used in IVS. In Section 5, we evaluate the performance of our feedback control mechanism and discuss its limitations. The results presented currently are from a prototype IVS which did not incorporate the current scalable feedback mechanism. We delay the discussion of related work until Section 6 in order to build up sufficient context to compare our results to others published in the literature. Section 7 concludes the paper.

2 Congestion control for real-time applications in a multicast environment

Feedback control mechanisms are used in the Internet to control the unicast distribution of non real-time traffic, specifically in TCP. There, 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 [16]. The control of multicast real-time data presents a new set of problems.

The 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. The goal for real-time traffic is to optimize the utility of the information delivered to the receivers. These goals are not necessarily equivalent. Therefore, the mechanisms proposed for the control of data traffic cannot be expected to be suitable for real-time traffic.

We observed earlier that real-time applications differ from network applications in that they require a certain minimum level of service to offer utility to the end user, whereas traditional applications will take whatever service they can obtain. Thus, there is a floor to the rate at which a real-time source can transmit and still send a useful stream. This presents the problem of who to satisfy when two applications compete for the same bandwidth, and whose combined minimum bandwidth requirements, i.e. combined floor rates, exceed the available bandwidth. The decision is a policy issue, and it is up to external forces to resolve the problem, either by turning off an application, or by negotiating with the network provider to get more bandwidth.

One solution is to say that who pays the piper calls the tune, i.e. whoever has administrative control over the network should be able to decide who shall get priority. However, in the absence of a visible reservation scheme - one in which a manager can place “policy” constraints as to who can make reservations when, and who can pre-empt who - alternative mechanisms must be used, as well as a default decision making process. The mechanism we describe in this paper cannot distinguish between flows, but it does make the problem of congestion visible to the application and to the user. It is then up to the application or user to sort out the policy issues as to who can use the available bandwidth[2].

We also make congestion visible to the user in order to cope with the problems of using heterogeneous applications, e.g. an audio and a video application. This is intended to meet the needs of video conferences, in which the audio is generated by some other application, or where the video streams are prioritized according to some conference management protocol[3]. By allowing the user to discover when congestion is experienced they can institute the conference policy on the priority of the particular video stream. For instance, they may choose to suggest to the congested users that they leave the video stream, and use only audio.

Control mechanisms for real-time traffic must be able to work in a multicast environment, and specifically to scale up with the number of receivers in a multicast group. Clearly, it is only if we push the responsibility for detection of congestion in part of the multicast tree onto the receivers that we can expect any scalability in a scheme. Although receiver-based detection of congestion is possible, it is rarely seen since unicast communication generally involves sending information about data received back to the sender anyway, which can be processed by the sender and used to initiate control actions. Another advantage for the receiver-based congestion detection is that the receiver can use metrics suitable for the local network technologies, creating the possibility of heterogeneous congestion signals in use on a single multicast distribution tree. To aid in the design of a generic algorithm, we use generic network state variables, namely UNLOADED, LOADED and CONGESTED. We use such variables because the heterogeneity of today's Internet gainsays any attempt to use a single metric to determine state. Instead we allow each host to make a local decision as to how it shall measure the state of the network, using an appropriate local measure.

The congestion control algorithm then becomes a gradual increase in bandwidth from the source to either the maximum useful rate or to the rate that drives one of the receivers to decide that the network is LOADED. The source then transmits at this rate, continually polling its users in a scalable manner to ensure that the network doesn't become congested and can take advantage of when the network is no longer LOADED. When a receiver detects that the network is CONGESTED, it tells the receiver, and the sender takes the appropriate action.

[2] The class-based queueing mechanism is one possible solution to this problem.

[3] For example in MICE conferences, the audio is handled by VAT, and the video is handled by IVS.

3 A scalable feedback mechanism

In this section, we describe a mechanism for eliciting feedback information from the receivers in a multicast group. We consider the general case when the size of the group is not known by the source or by any of the receivers. In some cases, additional information is available about the group. For example, many conferencing applications use a tight session control in managing the conference, in which case the identities of all the members are known. This information can be used to tune the feedback mechanism, and in particular to select good initial values. However, this information is not required since, as will be shown, the algorithm can be used to estimate both the size of the group and the number of receivers experiencing a particular network state.

3.1 Avoiding implosion

Soliciting information from receivers in a multicast group of indeterminate size might create a so-called implosion problem, in which a potentially large amount of feedback information is sent almost synchronously from the receivers back to the source [8]. Solutions to this problem have included probabilistic querying, randomly delayed responses, and an expanding scoped search [31].

In a probabilistic scheme, a receiver responds to the request from a source with a given probability. Typically, the request is sent again by the source after some timeout interval if no reply has been received. The probabilistic scheme is easy to implement, but it has limitations. For example, the source is not guaranteed to receive the worst news from the group within a certain time period. Furthermore, it is not clear how to set the probability of replying when the size of the group is not known. However, one advantage is that it is possible to bias the probability associated with a receiver according to the importance of the receiver.

In the random delay response scheme, each receiver delays the time at which it sends its response back to the source by some random amount of time. Clearly, this scheme is not sufficient to prevent the implosion problem, especially if the random delay is chosen too small. However, the scheme is very appealing, in the sense that it allows to receive the responses from all the receivers in the multicast group, if the delay can be adapted using some knowledge of the size of the group. This is done in VAT to time the state multicasts from multiple instantiations. There, the delay is set so that the bandwidth used by the state announcements is 1% of that by the audio stream [17].

In the increasing scope search scheme, the time-to-live (ttl) of the packets sent by the source is gradually increased. Therefore, packets travel further and further along the branches of the multicast tree. This scheme is clearly efficient when all that is required is to find the receiver closest to the source. However, we typically need to locate the receiver with the worst view of the state of the network. This receiver is unlikely to be located close to the source given the typical topology of a wide area network. Furthermore, the scheme is not expected to scale up well since the distribution of receivers in the ttl bands is not uniform, as most receivers are likely to be close to the source if only because of timezone differences.

Our mechanism attempts to combine the best part of the above schemes. We describe it next.

3.2 Description of the mechanism

The mechanism consists of a series of rounds of probing of the multicast group, in which we find the state of the network corresponding to the worst positioned receiver, estimate the number of receivers and elicit the worst round trip time (rtt) to any receiver in the group so that we know how to time the series of rounds. We term a series of rounds an epoch. Once the state of the network is discovered, we signal upwards to the application for the adjustment of the rate of the output stream if necessary, and we terminate the current epoch. How the rate is adjusted is left unspecified, since this is a matter of policy for the application. A specific example is described in Section 4.

The algorithm we describe here relies on probabilistic arguments for scalability. Both the source and all the receivers generate random[4] keys of length 16 bits at the start of an epoch (This size is justified in Section 5.1). When the source wishes to solicit responses from its receivers, it sends out its key and a number indicating how many of the digits of the key are significant. Initially, all digits are significant. If the the source key equals the receiver key, using the declared number of significant digits, then the receiver can send a response, according to the rules described below. If there is no response within a timeout set at twice the largest round trip time in the receiving group, then the number of significant digits is reduced by one, and the source issues the same key for another time period, and so on, until either the required number of responses indicating congestion has been received[5], or we have passed a round in which no digits were declared significant, in which case any receiver can send a response. This ends the epoch.

Figure 1 shows the header of the packets sent by the source. Figure 2 shows the header of the packets sent by the receivers. Our algorithm requires that packets sent by the source include a timestamp and a sequence number, both of which are available in RTP[6] [25].

The receiver responds on matching keys in two cases to allow the sender to estimate the size of the group and to learn the worst case state of the network.

- If the SIZESOLICITED bit is set in the header sent out, then a matching receiver will send a response back to the sender. This allows the sender to estimate the receiver group size, since there is a simple relationship between the average round upon which a receiver first matches the key, and the size of the group. The SIZESOLICITED bit is unset in all packets sent out in subsequent rounds to receiving a response, to prevent unwanted packets coming back.

- The current worst state of the network seen by the source is sent out in the STATE field. If the receiver matches the key and the state that they perceive in the network is worse than the current advertised state from the sender, then the receiver will send back a response, reporting the state that they currently see. The sender will then set the state sent out to the worst case reported state.

Once a receiver has responded in a given epoch, it will not allow responses to the same match for a period equal to the advertised maximum rtt. If the same match is made after this period, then it responds again. This protects the scheme against problems of lost responses.

When the sender receives a response containing a LOADED state, it sets the new state in all packets that it sends out, and continues to solicit responses. The application is presumed to be continuing to send at the optimum rate for the health of the network and the application. If a response indicates CONGESTION, then the sender signals upwards to the application (in the hope that the application will do something sensible such as reduce its bandwidth), and terminates the current series of solicitations. It then commences a new epoch of solicitations, with the state reset to UNLOADED and the SIZESOLICITED bit set. If the sender manages to get all the way through the series of rounds, including the no matching required round without hearing of any part of the tree in a LOADED or CONGESTED state, then it signals upwards to the application that the network is UNLOADED, and that the application can increase its rate if it wishes. Once an epoch is completed, the sender initiates a new epoch. By continually probing the receivers with every packet, the sender will receive timely notification of network state changes.

Because the sender has an estimate of when the first match is likely to occur, the sender can adjust the number of significant digits sent out with the key so as to sensibly set the number of rounds taken to determine the state of the network in the first of a series of rounds. This significantly reduces the time taken to discover and react when things are going wrong. In addition, the round at which a response with a given state is received can be used as a hint to build an estimate of the number of receivers in a particular state. These estimates can be tracked using time averaging techniques.

[4] Regenerating keys at the beginning of every epoch eliminates the possibility of bias.

[5] This is currently set at one, although a more sophisticated analysis may use more responses

[6] We are currently carrying these headers as application-defined options in RTP.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|R| u |Sta| u |rttshft|keyshft|        key                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     maximum rtt in milliseconds                               |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

S = SIZESOLICITED; R = RTTSOLICITED; Sta = STATE from {UNLOADED, LOADED, CONGESTED}; u = unused bits; rttshft = the number of seconds to pick a delay from $2^{rttshft}$; keyshft = number of bits to use in matching the keys; maximum rtt = the maximum round trip time yet found in milliseconds.

Figure 1: Format of packet sent by the source

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|R| u |Sta| u |  delay in milliseconds                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     timestamp (seconds)       |     timestamp (fraction)      |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

S = response to SIZESOLICITED; R = RTTSOLICITED; Sta = STATE from {UNLOADED, LOADED, CONGESTED}; u = unused bits; delay = delay in a RTTSOLICITED response, 0 otherwise; timestamp = timestamp of the source packet that forced the response.

Figure 2: Format of packet sent by receivers

3.3 Maximum round trip time discovery

Round trip time discovery is implemented by the receiver saving the timestamp of a packet, drawing a random delay from a uniform distribution from a time period set by the sender, waiting for this delay, then returning a packet containing the original timestamp and the actual delay which the receiver waited. Since the sender knows the approximate number of receivers it is sending to, it can select a time period so as to get a reasonable rate of responses back. In terms of the packet headers, if the sender sets the RTTSOLICITED flag in the header, then if the receiver is not currently waiting to send a response, it selects a delay using the size of the shift that the sender has set in the RTTSHIFT field to set the time period as $2^{RTTSHIFT}$ seconds from which it draws the random waiting period.

The sender solicits rtt responses over periods defined as $2^{RTTSHIFT}$ seconds. It sets the maximum rtt as the worst received. To ensure the maximum rtt reflects the current state of the network and the receiver group and to eliminate the effects of outliers, the maximum rtt is gradually aged out every rtt solicitation period. The maximum rtt is used to determine the timeouts and duration of everything in the algorithm, to ensure that every receiver will respond.

4 A feedback control mechanism for multicast video

In Section 3, we described a mechanism to provide scalable feedback in large multicast environments. In this section, we describe how this mechanism is used to control the video coder in IVS. In Section 4.1, we describe how to control the output rate of a coder by adjusting parameters in the coding process. In Section 4.2, we describe how the output rate of the IVS coder are controlled using the feedback information.

4.1 Output rate control in video coders

Many algorithms have been proposed and standardized for the coding and transmission of video traffic. Example standards include ISO JPEG for single frame images, and ISO MPEG and CCITT H.261 for moving images [1]. In this paper, we consider video coders based on the H.261 standard.

A central part of a codec is the compression/coding algorithm. In H.261, a picture is divided 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 the difference between successive pictures [15].

The H.261 standard documents describe how parameters of a coder can be adjusted to change the output rate of the coder [15]. In IVS, we adjust three such parameters: the refresh rate; the quantizer; and the movement detection threshold.

The refresh rate characterizes the speed at which frames are grabbed from the camera. Decreasing the refresh rate decreases the average output rate of the coder. However, it also decreases the frame rate and hence the quality of the video delivered at the receivers.

The quantizer characterizes the granularity used to encode the coefficients from the discrete cosine transformation. 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 blocks in a frame which are "sufficiently different" from those in 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 decreases. However, increasing the threshold decreases the sensitivity of the coder to movement and yields lower image quality.

Thus adjusting the parameters of the video coder is an easy way, particularly in a software coder such as IVS, to trade off a lower output rate (i.e. lower bandwidth requirements) for a lower image quality. The specific requirements of a video application will dictate which of the three parameters described above should be adjusted. The refresh rate is adjusted if the precision of the rendition of individual images is important. The quantizer and the movement detection threshold are adjusted if the frame rate or the perception of movement is important. These requirements are taken into account in IVS as follows. The source 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. In the *Privilege Quality* mode (PQ mode), only the refresh rate is adjusted. 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*. In the *Privilege Rate* mode (PR mode), only the quantizer and the movement detection threshold are adjusted.

4.2 Feedback control of the H.261 coder in IVS

We next describe the mechanism which controls the output rate of the H.261 coder in IVS using the feedback information. We consider first the feedback information and then the control algorithm used by the coder.

In Section 3, we described the mechanism used by the receivers to send information about the state of the network. However, we did not specify the assignment of the generic variables LOADED, UNLOADED, and CONGESTED. Recall that our goal here is to maximize the perceptual quality of the image received at the destinations while minimizing the bandwidth used by the video transmission. Therefore, we need to map the state variables to the perceived quality of the received image.

A subjective measure of the perceptual quality, referred to as the Mean Opinion Score (MOS), has been defined and used extensively to design and compare video coding algorithms [18]. However, a MOS-based feedback mechanism would be impractical, since it would have to include the user in some kind of continual rating procedure. We thus have to rely on objective measures. Unfortunately, objective measures typically do not reflect the user's perception of an image [18]. The signal to noise ratio (SNR) is an objective measure of the spatial quality of the image. However, numerous experiments have shown that it is an imperfect measure because the perceptual quality in a sequence of frames depends on the quality of each frame in the sequence. Another objective measure of perceptual quality is the loss rate of blocks encoded by the H.261 coder. Yet another objective measure is the frame rate, i.e. the rate at which video frames arrive at the destinations.

In the absence of a reliable objective measure of perceptual quality, we characterize the quality of the video delivered to the receivers by a function of all the measures mentioned above, namely the SNR, the block loss rate, and the frame rate. However, we note that the SNR cannot be computed by the receivers since it requires that the original image be available. The loss rate of the encoded blocks cannot be computed by the receivers either since the coder only 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[7]. Finally, we note that the rate at which frames are sent by the source is known by the coder. Therefore, it seems reasonable to choose a feedback information based on measured packet losses at the receivers[8]. Specifically, each receiver measures an average packet loss rate observed during a time interval equal to an epoch. If the receiver can respond according to the rules in Section 3 (i.e. if its key matches the source key over the appropriate number of significant digits), it sends the state (i.e. UNLOADED, LOADED, or CONGESTED) corresponding to the measured loss rate.

An optimization of the information feedback mechanism when there is little danger of implosion has also been implemented. If the number of receivers is below a threshold, receivers send a negative acknowledgement (NACK) whenever they detect a packet loss. Upon receipt of a NACK, the source encodes the blocks which were in the lost packet using intramode coding. Observe that this will result in a refresh of the lost blocks (as opposed to a retransmission of these blocks, since the encoding is done on a new frame), and hence faster and better error recovery. The threshold is currently set

[7] It would be possible to measure the loss rate for the groups of blocks (GOBs) at the receivers. However, there is no direct relationship between the number of GOBs in a packet and the number of blocks actually encoded in the packet. This is because a GOB header must be sent by the source even if no block in the GOB was actually encoded. Therefore, the GOB loss rate is not a good approximation for the loss rate of blocks. Refer to [15] for details.

[8] Note that this criterion does not take into account possible random losses [24] which would be mistakenly considered as indicative of network congestion.

at 10 receivers. An analysis of the feasibility of this sort of scheme can be found in [8].

Once the source has received feedback information from receivers, it must convert the values into a global measure that it can use to adjust its output rate. The approach used within IVS is to reduce the output rate only if greater than a threshold percentage of the receivers are suffering excessive packet loss. We use the probabilistic properties of the probing algorithm to detect this, using the difference between the round in which a response to the SIZESOLICITED request was sent and the round in which a CONGESTED match was found. A logarithmic relationship obtained in Section 5.1 can be used to show that the above difference is related to the relative proportion of congested receivers. The current incarnation of IVS sets this difference to 6 rounds in a "large multicast" environment (i.e. when the number of receivers is large), or equivalently the control reacts when more than 1.4% of the receivers experience congestion (refer to Section 5.1).

In an earlier version of IVS, the source reacted when up to than 50% of the receivers experienced congestion. Clearly, this could lead to situations where up to half the destinations receive low quality video. Both choices of the 50 or 98.6 percentile are *ad hoc*, and none seems 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. This suggests that more elaborate criteria be used by the source. Ideally, the source should be able to single out the parts (i.e. branches) of the multicast tree that experience high loss rates, and to treat these branches separately. One way to do this is to use sub-band coding [18]. However, this would require some cooperation from the routers. Furthermore, the H.261 standard is ill-suited to support sub-band coding[9]. Nevertheless, this approach is promising, and we are currently investigating it.

We now describe the control algorithm used by the video coder. Control actions are taken by the coder at the end of an epoch. The control algorithm strives to keep the average "quality" across the receivers above some value, or equivalently it strives to keep the loss rate below some tolerable value for the majority of receivers so as to keep the network in the LOADED region. This is done by adjusting the maximum output rate of the coder max_rate.

In TCP, the window size, and to a first approximation the rate at which packets are sent into the network, is adjusted using a linear increase/multiplicative decrease algorithm. This is done so as to ensure stability and to provide both efficiency and fairness in the sense that n sources sharing the same link would be allocated on average a fraction $1/n$ of the capacity of the link [6]. While this might be non-optimal for video streams (refer to our discussion in Section 2), we choose to be conservative and we use a similar linear increase/multiplicative decrease algorithm. In this algorithm, max_rate is decreased by half if the fraction of congested receivers $frac_{CONGESTED}$ is greater than some threshold value, which we denote by $thresh_{CONGESTION}$. max_rate is increased by a fixed value $rateIncrement$ if all the receivers detect the network as UNLOADED. 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 $frac_{CONGESTED} > thresh_{CONGESTION}$
    $max\_rate = \max(max\_rate/2, min\_rate)$
else if $(frac_{UNDERLOADED} == 100\%)$
    $max\_rate = rateIncrement + max\_rate$

In IVS, the $rateIncrement$ is set to 10 kb/s. We set default values for min_rate, $thresh_{CONGESTION}$, and for the maximum value of max_rate to 15 kb/s, 1.4% and 150 kb/s, respectively. Note that these figures are heavily dependent on the policy decisions used in the network. For example, it is reasonable to set the value of $thresh_{CONGESTION}$ to much higher than 1.4% when the number of receivers is small. Different policies may result in different figures or even different adjustment algorithms.

5 Evaluating our feedback control mechanism

In this section, we evaluate the performance of our feedback control mechanism. We analyze the scalable feedback mechanism in Section 5.1, and the video control algorithm in Section 5.2.

5.1 Evaluating the feedback mechanism

Let n denote the number of receivers in the tree, and i denote length of the key, expressed in bits, sent by the source. We now derive the average round in which we get a first hit, i.e. the round in which the key of a receiver matches the key of the source over the appropriate number of significant digits. We number rounds in an epoch starting from 0. Let r_j denote the number of replies sent back to the source during round j and p_j denote the probability a receiver replies during round j. We assume that no message is lost in the network. The probability that we get at least one response in a round, given that there were no responses in the previous round is

$$\Pr(r_j > 0 | r_{j-1} = 0) = 1 - (1 - p_j)^n$$

where

$$\begin{aligned} p_j &= 2^{-i} & ; \ j = 0 \\ &= 2^{j-1}/(2^i - 2^{j-1}) & ; \ j > 0 \end{aligned}$$

This is because on the jth attempt, we have already tried 2^{j-1} numbers in the search space[10]. The average

[9] However see [13]. Other hierarchical schemes can be developed.

[10] Note that we would have $p_j = 2^{j-i}$ for $j \geq 0$ if a new key were generated at the source every round, as opposed to every

round $E(\text{First})$ when we get a first hit is then

$$E(\text{First}) = \sum_{j=1}^{i} j(1-(1-p_j)^n)(1-2^{j-1-i})^n + 1 - (1-2^{-i})^n$$

Figure 3 shows the graph of $\log E(\text{First})$ as a function of n for $0 \leq n \leq 10000$. The graph can be fitted to a

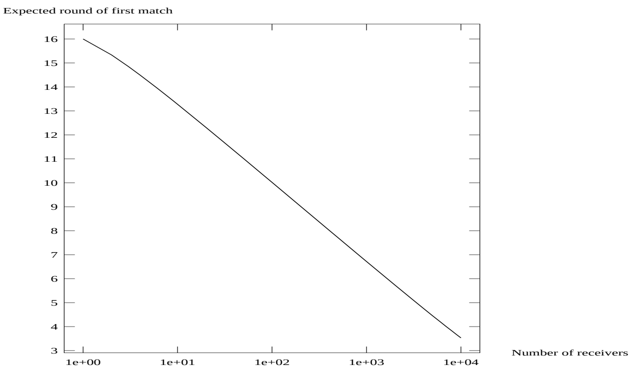

Figure 3: Expected round in which we get a hit vs number of receivers

straight line, and we obtain

$$n \sim e^{16.25 - E(\text{First})/1.4} \quad (1)$$

A simple manipulation of this equation shows that a difference (in means) of 6 rounds corresponds to a ratio of sample sizes of around 1 in 72, or approximately 1.4%. In IVS, we assume that the network is congested if the difference between the round in which a response to the SIZESOLICITED request was sent and the round in which a CONGESTED match was received is equal to 6. The result above shows that this occurs when approximately 1.4% of the receivers are in the CONGESTED state (refer to Section 4.2).

The model above also shows that our algorithm essentially eliminates the implosion problem. Consider a simplified algorithm in which an epoch ends as soon as a reply is received at the source[11]. In this case, we have

$$\Pr(r_j = m) = \Pr(r_j = m | r_{j-1} \neq 0)\Pr(r_{j-1} = 0)$$

The conditional probability $\Pr(r_j = m | r_{j-1} \neq 0)$ is distributed according to binomial distribution with parameter $p = 2^{j-1}/(2^i - 2^{j-1})$ and number n. The table below shows the probability of receiving more than 10 replies during the different rounds in an epoch, when the total number of receivers is 10 000. The rounds not shown have probabilities lower than 10^{-10}. The numerical values clearly indicate that the probability of receiving a large number of replies during the first round in which a hit occurs is very low.

epoch.

[11] In the real algorithm, a reply only changes the state advertised by the sender, which in turn reduces the potential population that can reply to subsequent source messages.

Round	4	5	6	7	8
Prob.	$2\ 10^{-8}$	$4\ 10^{-6}$	$9\ 10^{-5}$	$2\ 10^{-5}$	$3\ 10^{-9}$

The scalability of the scheme is illustrated by Equation (1), i.e. by the logarithmic relation between the probability of receiving a reply and the number of receivers. For comparison purposes, consider the random delay scheme with 10 000 receivers. If the source is to receive no more 10 replies per second in response to one of its requests, then the random delay has to be drawn from the range 0-1000 seconds . The upper bound of one thousand seconds on receiving feedback information effectively rules out adapting the source data rate to network conditions. However, in our scheme, the maximum maximum response time is equal to 32 times the worst case round trip time. For a typical worst case rtt of 500 milliseconds, the worst case state of the receivers can be found within 16 seconds. Since the probability of a response is dependent on the population size and the rules of the algorithm will always allow the first congested receiver to respond, the more receivers who are congested, the faster a response is returned.

5.2 Evaluating the control mechanism

In this section, we present results from experiments carried out over the MBone[12] with the feedback-controlled coder of IVS. The experiments aim at illustrating the impact of the control algorithm on both the bandwidth used in the network by the video application, and on the quality of the video stream delivered to the receiver. For simplicity, we set the number of receivers to be equal to one. Specifically, the video source is an IVS source located at INRIA in south-eastern France. The receiver is located at UCL in London, UK. Note however that the connection between UCL and INRIA is a multicast connection, i.e. the packets sent over the connection are carried over the MBone. The sequence of multicast routers between INRIA and UCL is shown in the table below.

Machine name	Institution and/or country
sobone.inria.fr	INRIA, France
fmroute11.exp.edf.fr	Paris, France
test-RS.ripe.net	Amsterdam, NL
broodjeham.surfnet.nl	Amsterdam, NL
noc.ulcc.ja.net	University London, UK
laphroaig.cs.ucl.ac.uk	UCL, UK

Of course, the path between two multicast routers might be somewhat circuitous, e.g. the path from Paris to Amsterdam goes through CERN in Geneva, Switzerland.

Figure 4 shows the evolutions as a function of time of the maximum output rate *max_rate* at the source. The video sequence sent during this experiment lasted

[12] 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.

for more than 11 hours. Figure 5 shows the corresponding evolutions of the packet loss rate measured at the receiver.

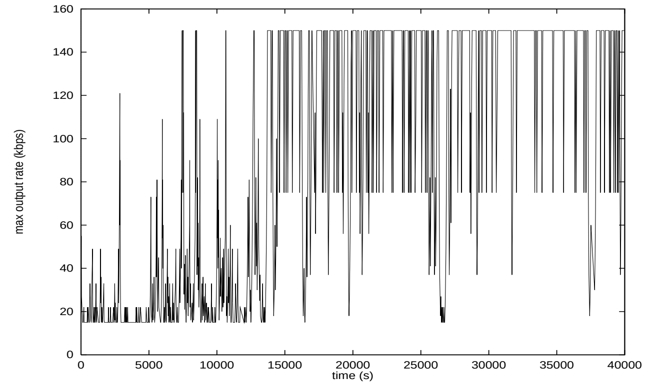

Figure 4: Evolutions of max_rate (in kb/s) vs. time

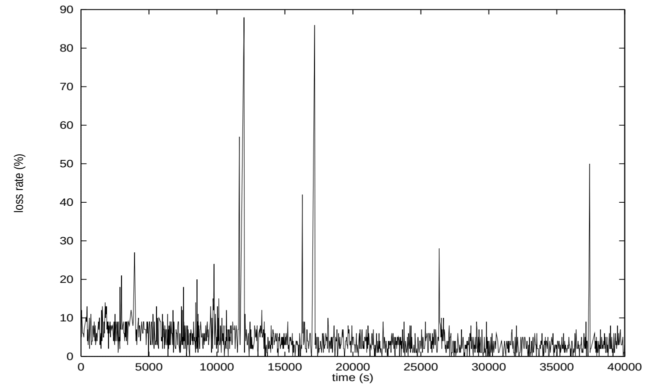

Figure 5: Evolutions of the loss rate (in %) vs. time

As expected, we observe that a higher loss rate yields a lower value of max_rate, and hence a lower bandwidth requirement by the source. If the state of the network is found by the receiver to remain CONGESTED (i.e. the loss rate remains greater than 5%) for a long time, then the value of max_rate eventually reaches its minimum value $min\_rate = 15$ kb/s. This is visible for $t \leq 5000$ s. If the packet loss rate remains low, then the value of max_rate eventually reaches its maximum value of 150 kb/s. This is visible for $t > 15000$ s. The figures clearly show that the video control mechanism prevents the source from swamping the resources of the network. Indeed, a congested network will result in high loss rates, and hence a lower send rate at the source.

As we mentioned earlier, a low bandwidth at the source translates into a lower image quality. There remains to quantify this bandwidth-gained/quality-lost tradeoff. The main problem is to find a way to estimate the quality of the video data delivered to the receiver. We argued in Section 4.2 that the loss rate is a good indication of this quality. Experiments show that the control mechanism decreases the bandwidth requirements as well as the loss rate at the receiver, as long as the video traffic makes up a significant part of the total traffic on the path from INRIA to UCL.

However, the loss rate is a not enough to evaluate the quality of the video at the receivers. Therefore, we have been carrying out an evaluation of the effectiveness of our control scheme within the MICE project [2] using questionnaires to discover user satisfaction with the quality of the images delivered to the receivers, and monitoring traffic rates and packet losses to detect network congestion. This is being with tools such as rtpqual and an instrumented version of IVS during the seminars and weekly meetings held between MICE partners. The first questionnaires were filled out only weeks ago. Therefore, it might be a bit early to draw conclusions from the limited number of replies available at the time of writing. Nevertheless, our preliminary results are encouraging, indicating that 90% of the time, our pool of 13 users believe that the conference quality has been improved by the new scheme.

6 Related work

Until recently, video was typically transmitted over CBR networks. Since the rate of a video sequence can vary rapidly with time, the problem was to obtain from a variable rate sequence a constant rate stream of data that could be sent into the network. This is typically done by sending the video stream 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 output rate of the coder to prevent buffer overflow or underflow (e.g. [5]). Recently, packet-switched networks have been available that can provide channels with deterministic guarantees [21]. A control mechanism suitable for such networks is described in [10].

Feedback control mechanisms have also been proposed for networks with VBR channels such as the Internet. There, feedback information about the changing state of the channels is used by a controller to adjust the output rate of the video coders [14] (feedback controllers have also been developed for audio coders, e.g. [32]). Examples of such controllers are described in [19, 20, 29]. However, none of the mechanisms described in these references handles multicast distribution. Furthermore, the scheme in [29] requires that the source know the service rate of the bottleneck on the path from the source to the destination. This rate can be estimated in networks where the switches use a round robin or equivalent discipline. However, it cannot be estimated reliably in networks with FCFS switches such as the Internet. The scheme in [20] requires that switches send their buffer occupancies and service rates back to the source. The scheme in [19] 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). This is similar to our control scheme when in the PQ mode. Other schemes are designed for specific applications, typically to control the delivery of video streams from a server to client workstations [22, 23].

7 Conclusion

Video applications are thought to fall in the class of adaptive applications [7, 30], which can be shown to benefit from sharing rather than reserving the resources of the network [27]. However, sharing might lead to over-consumption of a resource by a greedy application. Our work provides a way to control video applications in a way that scales up well with the number of receivers in the multicast tree. The control mechanism makes it possible to hold videoconferences with reasonable quality even across congested links without requiring special support from the network such as admission control or resource reservation mechanisms. Furthermore, our mechanism prevents video sources from swamping the resources of the Internet.

The algorithms and techniques we described can be used in other tools such as NV or VAT. Specifically, our mechanism can be used to control the output rate of MPEG or JPEG-based coders. Also note that the scalable feedback mechanism can be viewed as a distributed searching technique. As such, it can be used in other applications where we are attempting to elicit the worst or best service from a large number of servers, such as an attempt to find the least loaded server for a load balancing system amongst a large number of servers. By using RTP options and offering a standard application programming interface to the code, the probing mechanism can be incorporated easily within the code of these applications.

The scalable feedback mechanism and the video control algorithm described in the paper are included in release 3.3 of IVS. The latest release is available by anonymous FTP from zenon.inria.fr in the directory rodeo/ivs.

References

[1] R. Aravind et al., "Image and video coding standards", *AT&T Tech. Journal*, pp. 67-89, Jan/Feb. 1993.

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

[3] J-C. Bolot, T. Turletti, "A rate control mechanism for packet video in the Internet", to appear in *Proc. IEEE Infocom '94*, Toronto, Canada, June 1994.

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

[5] 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.

[6] D-M. Chiu, R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks", *Comp. Networks and ISDN Systs.*, vol. 17, no. 1, pp. 1-14, 1989.

[7] 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.

[8] J. Crowcroft, K. Paliwoda ,"A multicast transport protocol", *Proc. ACM Sigcomm '88*, Stanford, CA, pp. 247-256, Aug. 1988.

[9] S. Deering, "Host extensions for IP multicasting", RFC 1112, January 1989.

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

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

[12] S. Floyd, V. Jacobson, "Random Early Drop Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, August 1993.

[13] M. Ghanbari, "Two-layer coding of video signals for VBR networks", *IEEE JSAC*, vol. 7, no. 5, pp. 771-781, June 1985.

[14] 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.

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

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

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

[18] N. Jayant, J. Johnston, R. Safranek, "Signal compression based on models of human perception", *Proc. IEEE*, vol. 81, no. 10, pp. 1385-1422, Oct. 1993.

[19] 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.

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

[22] 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.

[23] 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.

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

[25] 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.

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

[27] S. Shenker, Presentation at the November 1993 IETF meeting, Houston, TX.

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

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

[30] F. Wilson, I. Wakeman, W. Smith, "Quality of Service Parameters for Commercial Application of Video Telephony", Human Factors in Telecommunication Symposium, Darmstadt, Germany, March 1993.

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

[32] N. Yin, M. Hluchyj, "A dynamic rate control mechanism for source coded traffic in a fast packet network", *IEEE JSAC*, vol. 9, no. 7, pp. 1003-1012, Sept. 1991.