# Control Mechanisms for Packet Audio in the Internet*

Jean-Chrysostome Bolot  Andrés Vega-García
INRIA
B. P. 93
06902 Sophia-Antipolis Cedex
France
{bolot, avega}@sophia.inria.fr

**Abstract**

The current Internet provides a single class best effort service. From an application's point of view, this service amounts in practice to providing channels with time-varying characteristics such as delay and loss distributions. One way to support real time applications such as interactive audio given this service is to use control mechanisms that adapt the audio coding and decoding processes based on the characteristics of the channels, the goal begin to maximize the quality of the audio delivered to the destinations. In this paper, we describe and analyze a set of such control mechanisms. They include a jitter control mechanism and a combined error and rate control mechanism.

These mechanisms have been implemented and evaluated over the Internet and the MBone. Experiments indicate that they make it possible to establish and maintain reasonable quality audioconferences even across fairly congested connections.

## 1 Introduction

The transmission of voice over packet switched networks was an active research area in the late 70's and the early 80's [29]. Much of the work then focused on using packet switching for both voice and data in a single network. Packet voice, and more generally packet audio applications, have recently become again of interest. This interest has been fueled by the availability of supporting hardware (microphones now come standard with most workstations), of increased bandwidth throughout the Internet, and by the development of the MBone [7]. A variety of audio tools such as *vat* [17] or *Nevot* [24] have been available for a few years, and they have been used to audiocast conferences. Recently, several more tools have been announced, which claim to provide toll-quality workstation or PC audio over the Internet for a fraction of the cost of a telephone call (see [5] for pointers to these tools and other information related to packet audio).

However, the Internet provides a simple single class best effort service. From a connection's point of view, the best effort service amounts in practice to offering a channel with time-varying characteristics such as delay and loss distributions [2, 21]. These characteristics are not known in advance since they depend on the (*apriori* unknown) behavior of other connections throughout the network. This makes it essentially impossible to provide performance guarantees such as minimum loss rate or maximum delay. Thus, it is not clear how well applications with minimum guaranteed requirements such as audio applications can work over the Internet. Experimental evidence suggests that, although the quality of the audio delivered by Internet tools has improved, audio quality is still mediocre in many audio conferences. This is clearly a concern since audio quality has been found to be more important than video quality or audio/video synchronization to successfully carry out collaborative work [15].

It should be pointed out that bad audio quality is often caused by problems having little to do with either the network service or the audio tools themselves. The experience accumulated with the audiocasting of MICE [20] and IETF meetings suggests that badly tuned or set up microphones and speakers are responsible for many such problems. However, all these can be addressed by users at their own sites. Furthermore, their impact is expected to decrease as users become familiar with the tools and the tools themselves become more user friendly. In any case, the most persistent problems with audio quality are caused by the network, or rather by the impact of traffic in the network on the stream of audio packets. Two approaches have emerged to tackle this problem.

One approach is to extend current protocols and switch scheduling disciplines to provide the desired requirements. This approach requires that admission control, reservation, and/or sophisticated scheduling mechanisms be implemented in the network. These mechanisms are not yet implemented in the Internet,

**In* Proc. IEEE Infocom '96, San Fransisco, CA, pp. 232-239, April 1996.

and their design, analysis, and evaluation is still an active research area [26]. Thus, we have not pursued this approach so far.

Another approach is to adapt applications to the service provided by the network. This amounts in practice to adapting applications to the time-varying characteristics of the connection over which the application data packets are sent, the goal being to maximize the quality of the data delivered to the destinations. Experimental evidence suggests that the quality of the audio depends essentially on the number of lost packets and on the delay variations between successive packets. Thus, the most important network characteristics for audio applications are the delay variance (or jitter), and the loss distributions. Furthermore, for live audio applications such as audioconferences, the average end-to-end delay must be small to allow interactions between participants.

The goal then in this approach is to develop mechanisms that attempt to eliminate or at least minimize the impact of packet loss and delay jitter on the quality of the audio delivered to the destinations. We have developed a set of such mechanisms. One mechanism adjusts the playout time of audio packets at the destination, the objective being to minimize the impact of delay jitter. A second mechanism adds redundancy information in the audio packets sent by the source, the objective being to minimize the impact of packet loss. A third mechanism controls the rate at which packets are sent over a connection, the objective being to match the send rate to the capacity of the connection and hence to minimize packet loss. The second and third mechanisms both attempt to minimize the impact of packet loss, and they really are two sides of a joint error/rate control mechanism.

These mechanisms have been implemented in a new audio tool developed at INRIA. For lack of space (and as suggested by reviewers) we do not describe in the paper the jitter control mechanism. We focus instead on the rate and error control mechanisms. In Section 2, we describe the structure of the audio tool. In Section 3, we characterize the loss process of audio packets, and describe and evaluate a packet loss recovery scheme. In Section 4, we describe and evaluate a joint error and rate control scheme. Section 5 concludes the paper.

## 2 The audio tool

The structure of the audio tool is shown in Figure 1 below. It is being developed within the MICE project in collaboration with a group at University College London (UCL). Work at UCL has focused on device-independent audio input, efficient mechanisms for silence detection, automatic gain control, and echo cancellation, and on the evaluation of the

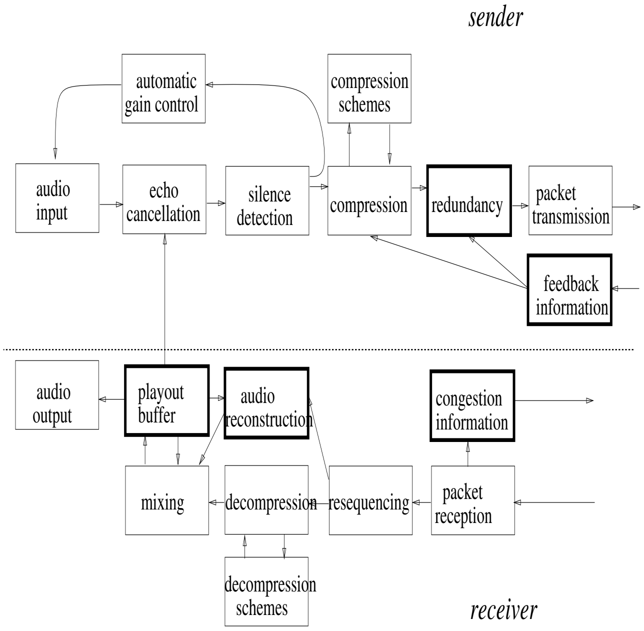


Figure 1: Structure of the audio tool

auditory quality of the signal delivered to the destinations. Work at INRIA has focused on coding schemes, and on jitter, rate, and error control mechanisms.

The coding schemes available at this time use 8-kHz sampled speech with bit rates varying from a few kb/s to 64 kb/s. Specifically, they include a 64-kb/s $\mu$-law PCM, various adaptive delta modulation (ADM) coders with rates varying from 16 kb/s (for ADM2) to 56 kb/s (for ADM6), a 13 kb/s GSM coder, and a 4.8 kb/s LPC low bit rate coder. Work is underway to include wideband speech coders. The PCM, ADM6, ADM5, and GSM coders deliver high quality audio with MOS scores above 3.5. The ADM2, ADM3, and LPC coders delivers audio with a somewhat lower quality. However, even a mediocre low bit rate coder turns out to be useful for error control purposes (refer to Section 3). The boxes in the figure which involve one of the control mechanisms of interest in the paper have been highlighted. They include the redundancy box (which involves the error control mechanism), the congestion information and feedback information boxes (which involve the error/rate control mechanism), and the playout buffer box (which involves the jitter control mechanism).

The audio packets are sent from the source to the destination(s) using IP (or its multicast extension), UDP, and RTP. To each audio packet is associated a timestamp and a sequence number. The timestamp is used to measure end-to-end delays, and the sequence number is used to detect packet losses.

## 3 A loss recovery mechanism

Anecdotal evidence suggests that audio quality is still mediocre in many audio connections because of packet losses. This makes it important to implement an efficient loss recovery mechanism for audio applications. We address this problem in this section. Our main result is that open loop error control mechanisms based on forward error correction are adequate to reconstruct most lost audio packets. We describe one such mechanism and report on improvements of audio quality obtained with it.

### Analysis of the loss process

Many different quantities can be used to characterize the loss process of audio packets. The obvious measure is the average loss, or unconditional loss probability. Let $l_n$ denote a boolean variable which is set to 1 if packet $n$ is lost, and 0 otherwise. The average loss is thus equal to the expected value of $l_n$. We denote $ulp = E[l_n]$. However, $ulp$ does not characterize the burstiness of the loss process, or equivalently the correlation between successive packet losses. One way to capture such correlation is to consider the conditional probability that a packet is lost given that the previous packet was lost. We denote $clp = P[l_{n+1} = 1 | l_n = 1]$.

We have analyzed $clp$ and $ulp$ in the Internet using measurements and analysis. The measurements have all be done using the PCM coder with 320-byte packets (or 40 ms of speech) between INRIA Sophia Antipolis and University College London (UCL) in the UK. Figure 2 shows the evolutions of the number of consecutively lost packets as a function of $n$ measured at 3:00 pm. The average loss $ulp = 0.21$ is quite high

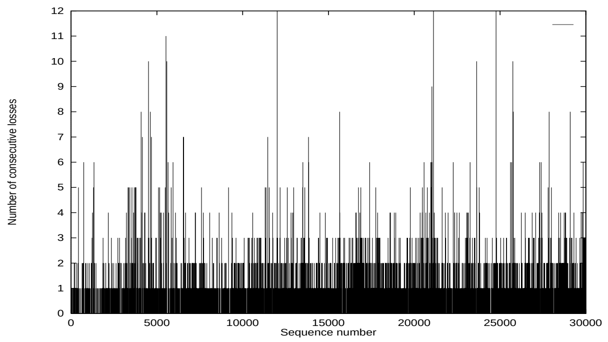


Figure 2: Evolutions of the number of consecutively lost packets

because the INRIA-UCL connection is heavily loaded during daytime. However, it appears that most loss periods involve one or two packets. This observation is confirmed by looking at the frequency distribution in Figure 3, which shows the frequency distribution of the number of consecutive losses (i.e. the number of occurrences of $n$ consecutive losses for different $n$) corresponding to the trace in Figure 2. The slope

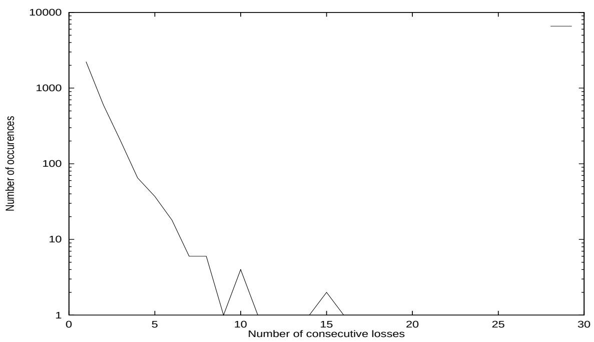


Figure 3: Frequency distribution of the number of consecutively lost packets

of the distribution decreases linearly near the origin. Since the figure is drawn on a log scale, this indicates that the probability distribution decreases geometrically fast away from the origin.

We have examined the loss process of audio packets over unicast connections other than the INRIA-UCL connection, and over multicast connections as well. In all cases, we have found that the frequency distribution of the number of consecutively lost packets is similar to that described above [4].

### Packet loss recovery schemes

A loss recovery scheme is required if the number of lost audio packets is higher than that tolerated by the listener at the destination. Loss recovery is typically achieved in one of two ways. With closed loop mechanisms such as Automatic Repeat Request (ARQ) mechanisms, packets not received at the destination are retransmitted. With open loop mechanisms such as Forward Error Correction (FEC) 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.

ARQ mechanisms are not generally acceptable for live audio applications because they increase end to end latency. Furthermore, they do not scale well to large multicast environments. FEC is an attractive alternative to ARQ for providing reliability without increasing latency [1]. However, the potential of FEC mechanisms to recover from losses depends crucially on the characteristics of the packet loss process in the network. FEC mechanisms are more effective when lost packets are dispersed throughout the stream of packets sent from a source to a destination. Thus, our measurements above indicate that FEC is particularly well suited for audio applications over the Internet.

Many FEC mechanisms proposed in the literature involve exclusive-OR operations, the idea being to send every $n$th packet a redundant packet obtained by exclusive-ORing the other $n$ packets [25]. This mechanism can recover from a single loss in a $n$ packet message. It is a very simple mechanism, but it increases the send rate of the source by a factor of $1/n$, and it adds latency since $n$ packets have to be received before the lost packet can be reconstructed.

Within the MICE project, we have developed a novel mechanism for loss recovery. Consider for example the case when audio is sent using PCM encoding. In our mechanism, packet $n$ includes in addition to the PCM encoded samples, a redundant version of packet $n-1$ typically obtained with a low bit rate coder such as the LPC or GSM coder. LPC is a CPU-intensive coding algorithm. However, it adds very little overhead (24 bytes) per 320-byte PCM encoded packet. Our mechanism improves upon an earlier mechanism in which the redundant information in packet $n$ includes short-term energy envelopes as well as the number (and possibly the location) of zero crossings of packet $n-1$ [10].

Both mechanisms can recover from isolated losses. If packet $n$ is lost, the destination waits for packet $n+1$, decodes the redundant information, and sends the reconstructed samples to the audio driver. In our case, the audio output consists of a mixture of PCM-, LPC-, or GSM- or ADM-coded speech. The subjective quality of this reconstructed speech has been carefully evaluated by our MICE colleagues at UCL. They have obtained results (detailed in [13]) that show that audio quality as measured by intelligibility hardly decreases as the loss rate reaches 30% even when a relatively low quality LPC coder is used to obtain the redundant information[1].

Even though most loss periods involve one packet (i.e. most losses are isolated), it is important to recover from multiple consecutive losses. This is intuitively clear since longer loss periods have a larger (negative) impact on audio quality than do shorter periods do. Of course, we could combine the above mechanism with packet repetition to recover from two consecutive losses. However, this does not yield much audio quality improvement over the original mechanism. Furthermore, we have found that in high loss and high load situations, the most important task of an audio tool is to deliver decent quality audio to the destinations. Thus, our approach in such cases is to increase the amount of redundancy carried in each packet. We can use as redundant information in packet $n$, LPC or GSM versions of packets $n-1$ and $n-2$, or of packets $n-1$, $n-2$ and $n-3$, or of packets $n-1$ and $n-3$, etc.

Clearly, adding redundant information increases the CPU and the bandwidth requirements of the coder. Table 1 shows the cost in terms of CPU and bandwidth of various coding schemes. The CPU cost is the time, measured on a 75-Mhz Sparc20 workstation, needed to encode a packet relative to that required to encode a PCM packet.

| Coding scheme | Relative CPU cost | Bandwidth (kb/s) |
|---|---|---|
| PCM | 1 | 64 |
| ADM6 | 13 | 48 |
| ADM4 | 11 | 32 |
| ADM2 | 9 | 16 |
| GSM | 1200 | 13 |
| LPC | 110 | 4.8 |

Table 1: Relative CPU cost and bandwidth requirements of various coders

For convenience of exposition, we use throughout the rest of the paper the notation (coding algorithm, redundant algorithm ($i$)) to indicate that audio packet $n$ includes as redundant information audio packet $n-i$ encoded with the appropriate coding algorithm. For example, (PCM, ADM(1)) or (PCM, GSM(1), GSM(3)). In the later example, both packets $n-1$ and $n-3$ encoded with the GSM coder are included in packet $n$ along with the main PCM-coded information. To each combination of main and redundant information we associate a CPU cost, bandwidth requirements, a delay, and a reward. The CPU cost and the bandwidth requirements have been examined above. The delay characterizes the time required at the destination to reconstruct lost packets. This time can become quite large if the redundant information includes packet $n-i$ with $i$ large, since $i$ packets have to be received after a loss to reconstruct the lost packet. In this section, we use the maximum value of $i$ in a packet as the delay. The reward characterizes the auditory quality perceived at the destination given the chosen combination of main and redundant information. Given the lack of objective measures of audio quality, we use the loss rate after reconstruction as the reward (note that the reward then unfortunately becomes independent of the scheme used to encode the redundant information).

Table 2 shows for a few combinations the associated delay and reward. In all cases, the reward is relative to that of the base combination, i.e. (PCM). For example, a relative reward equal to 3 indicates that the loss rate at the destination after packet reconstruction is 1/3 that before reconstruction. The results have been measured over 10 experiments carried out over the INRIA-UCL connection at various times of the day. The

[11] The efficacy of this scheme can be checked at URL http://www.inria.fr/rodeo/personnel/huitema/audio.html.

bandwidth requirements and the relative CPU cost of each combination are easily obtained from Table 1.

| Combination | Delay | Reward |
|---|---|---|
| (PCM) | 1 | 1 |
| (PCM, ADM4(1)) | 2 | 2.5 |
| (PCM, GSM(1)) | 2 | 2.5 |
| (PCM, LPC(1)) | 2 | 2.5 |
| (PCM, ADM4(2)) | 3 | 6 |
| (PCM, ADM4(1), ADM2(2)) | 3 | 6 |
| (PCM, ADM4(1), ADM2(3)) | 3 | 10 |
| (PCM, ADM4(1), ADM2(2), ADM2(3)) | 4 | 18 |

Table 2: Delay and reward for various combinations of main and redundant information

As expected, adding redundant information increases the CPU cost, the bandwidth requirements, and the delay as well as the reward. We note that the last few combinations in the table are clearly overkills if the network load is low and packet losses are rare occurrences. 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 describe one such mechanism next.

# 4 A combined rate and error control mechanism

Let us first consider the case of a unicast audio connection. Most packet losses observed over such a connection are caused by congestion in the network. The state of congestion might have been created by the audio connection, by other connections sharing common resources with the audio connection, or by both. Typically, sources react to congestion by decreasing their bandwidth requirements in an attempt to reach a state where network utilization is high. If all sources use the same bandwidth control mechanism, and if this mechanism is designed adequately, then all sources share the resources of the network fairly [6]. In practice, however, not all sources use the same mechanism. Furthermore, some sources do not use any control mechanism at all. As a consequence, an audio source which decreases its bandwidth requirements upon detecting congestion might not observe any decrease in its loss rate as a result of its action. Thus, it is difficult to evaluate and to control the impact of a bandwidth (or rate) control action taken by a source on the state of the network in general, and on the loss rate for this source-destination pair in particular. Note that the problem essentially stems from the current stateless architecture of the Internet illustrated by the FIFO discipline at the switches which makes the delay and loss processes of a connection strongly dependent on the arrival processes of other connections.

Minimizing audio packet losses then cannot be done reliably by simply controlling the send rate of the audio connection. Our approach is to minimize packet losses by controlling the send rate as well as the loss recovery process at the destination, which in practice is done by controlling the amount of redundant information added in audio packets at the source based on feedback information about the loss process measured at the destination. We use the same approach in the case of a multicast audio connection. However, the feedback information there reflects the loss process measured at all the destinations. We consider this general case of multicast audio delivery in the remainder of this section.

There has been much work in the past on control mechanisms for packet audio. However, most of it has focused on dynamic rate control mechanisms in a homogeneous environment, i.e. with audio sources only (e.g. [9, 30]), or on selective packet discarding in integrated environments, i.e. with data and audio sources (e.g. [28, 22]). Work on dynamic error control has focused on control mechanisms for ARQ-type protocols (e.g. [18]). We believe our tool is the first packet audio tool that uses joint dynamic error and rate control mechanisms.

In practice, we combine the rate control and the error control mechanisms into one joint rate/error control mechanism. The goal then is to adjust at the source both the send rate and the amount of redundant information to minimize the perceived loss rate at the destinations. To achieve this goal, we must be able to i) adjust the rate at which packets at sent into the network, ii) adjust the amount of redundancy information added in these packets, iii) elicit feedback information about the loss rates measured at the destinations, and iv) define a control mechanism which takes this feedback information to adjust the redundant information and send rate at the source accordingly. We consider these issues next.

Regarding point i), there does not seem to exist an audio coder the output rate of which can be controlled over a wide range of bit rates with a relatively fine granularity[2]. This is unlike for video, where efficient algorithms have been devised to produce embedded bit streams that can be easily controlled by dropping less important bits [19, 8]. One way around this problem is to use a panoply of audio codecs. As mentioned earlier, we use PCM (at 64 kb/s), various ADM (between 16 kb/s and 48 kb/s), GSM (at 11 kb/s), and LPC (below 5 kb/s) coders. This makes it possible to choose the coding scheme with a bandwidth requirement closest to that desired. However, the granularity of the rate adjustment is coarse.

Regarding point ii), we have seen in Section 3 that combinations of redundant information can be used to

[2][2]However, variable bit rate coders with a limited range of output bit rates do exist (e.g. [14]).

provide different levels of error correction.

Regarding point iii), we have chosen a feedback information based on packet loss rates (measured before possible packet loss reconstruction) at the destinations. Specifically, each receiver measures an average loss rate observed during a given time interval. This measure is then sent back to the source and to the other destinations using the QoS reporting mechanism of RTP v2 [23]. This reporting mechanism is interesting because it is scales well to a large number of receivers [11], and because it provides interoperability with other audio tools.

Regarding point iv), the first task of the control mechanism is to relate the state of the entire multicast group within the context of the application. In other words, the source must convert the QoS (i.e. the loss rates) received from each destination into a global QoS measure which represents the overall quality of the audio received at the destinations. We take our global QoS measure to be a 90 percentile QoS, i.e. the smallest QoS better than 90% of the QoS values reported by the destinations. The control algorithm then will strive to keep this value below some tolerable loss rate. Packet loss rates of between 1 and 10% can be tolerated depending on the way in which voice is coded and missing packets are masked [16].

The choice of the 90 percentile QoS as the global QoS is *ad hoc*, and it might not always be adequate in a heterogeneous network. Clearly, a source-based rate control mechanism as that considered here attempts to deliver at any given time a uniform quality of audio to all the destinations on the multicast tree even with different branches of the tree have widely different bandwidth (and thus experience widely different loss rates). A solution to this problem [27, 19] is to use a layered coder in conjunction with a receiver-based control mechanism in which receivers adapt to network conditions (i.e. to loss rates on different branches of the tree) by adding and dropping layers. Unfortunately (refer to our discussion earlier), very few layered audio coders are available.

We now describe the control algorithm used by the audio coder. Let us first consider the case when the coder adjusts only its output rate in response to feedback information. The algorithm gradually increases the send rate at the source if the global QoS (i.e. the 90 percentile loss rate) is below a threshold. It decreases the bandwidth if the global QoS is above another threshold. Figure 4 shows example evolutions of the send rate and the global QoS measured between INRIA and UCL. In this case, the low threshold was set to 5% and the high threshold to 15%. The interval between the reception of successive QoS reports, and hence between successive control actions, is equal to 5 seconds. As expected, we observe that the send rate

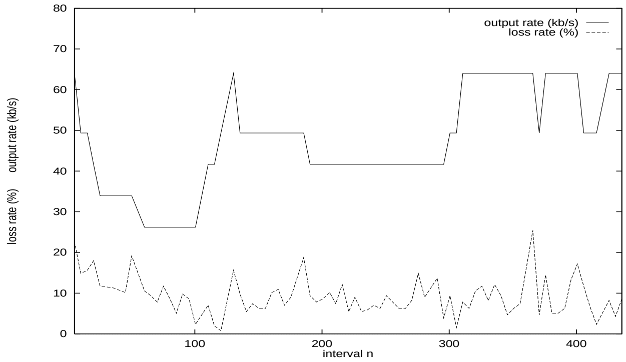


Figure 4: Evolutions of the loss rate at the destination and the output rate of the coder

and hence the bandwidth requirements of the source decrease when network congestion increases. Unfortunately, we have little control over the traffic and the network load between INRIA and UCL. To better understand the behavior of this and other control mechanisms, we have set up a test network at INRIA with known topology and link bandwidths, and with controlled traffic sources.

We have carried out an experiment over this network in which a variable number of audio connections share a 100 kb/s link. The initial coding scheme used by the sources is the 64 kb/s PCM scheme. If the source coders do not use the error/rate control mechanism, then packet losses increase rapidly with the number of active sources. We have found that audio quality at the destinations becomes mediocre as soon as this number exceeds 3. With the rate control mechanism switched on, source coders decrease their output rates upon receipt of bad QoS reports by switching from PCM coding to ADM6, then ADM5, down to ADM4 and GSM coding. As a result, the loss rates at the destinations remain close to 0 and audio quality is kept relatively constant since all the PCM, ADM5/6, and GSM schemes have MOS scores above 3.5. Figure 5 shows the evolutions of the output rates of the coders[3]. Source 1 is active during the entire experiment. Source 2 starts transmission at time $t = 45$ and stops at time $t = 345$. Source 3 starts transmission at time $t = 160$ and stops at time $t = 245$. As expected, active sources share the bandwidth fairly evenly. The large oscillations in the output rate for $t \geq 250$ s occur because all connections share the same links and buffers. Therefore, QoS reports from different destinations tend to arrive at the sources at the

[33]It appears that at time 0 a single source sends data at 66 kb/s as opposed to the 64kb/s associated with PCM. This is because the output rate figure has been obtained by including the RTP header data in addition to the audio data proper.

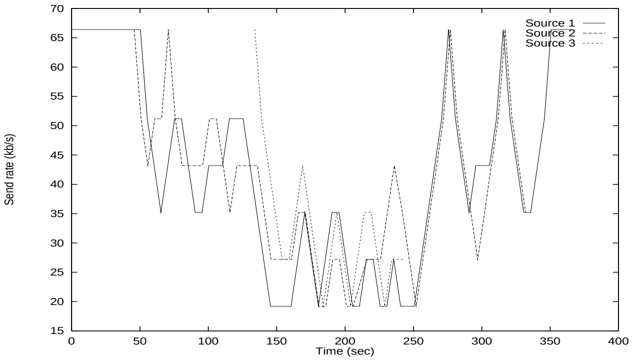


Figure 5: Evolutions of the output rate of multiple sources sharing a 100 kb/s link

same time. Thus, sources tend to take control actions synchronously, which creates large-amplitude oscillations.

We demonstrated earlier that a rate control mechanism alone is not sufficient to deliver good quality audio at the destinations given the current Internet architecture. Thus we now consider the general case when the coder adjusts the amount of redundant information as well the output rate in response to feedback information.

The control mechanism chooses one of the combinations in Table 4 next page. It changes from combination $i$ to combination $i+1$ if the global QoS (i.e. the 90 percentile loss rate) is below a threshold. It changes from combination $i$ to $i-1$ if the global QoS is above another threshold. Note that the control mechanism is essentially an error control mechanism for low values of $i$, and a rate control mechanism for large values of $i$. Upon detecting congestion, the source send rate does not decrease. Instead, the amount of redundant information increases. Thus, the audio streams uses more network resources than it would with a TCP-like control mechanism. This amounts in practice to giving priority to the audio stream over other streams. Not surprisingly, we have found this specific mechanism to provide slightly better audio quality than other TCP-like mechanisms we experimented with.

Our algorithm is a very unsophisticated version of a joint source/channel coding algorithm. The use of such algorithms has been advocated in both the networking and the signal processing communities (e.g. [12]), and we are investigating better designed and more efficient algorithms. However, the algorithm above is useful because it is very simple to implement and it is computationally cheap[4]. Furthermore, it serves as a baseline to evaluate future algorithms.

[44] The combinations in Table 4 involve ADM-coded redundant information precisely so as to minimize the CPU requirements at the source.

Figure 6 shows measurements obtained between INRIA and UCL during daytime. The low and high loss thresholds were both set to 3%, meaning that the control mechanism attempts to keep the loss rate at the destination after packet reconstruction around 3%. The figure shows the evolutions of the loss rate at the destination before and after reconstruction. It also shows which combination (identified by the combination number in the table above) was used at any given time. We observe that the mechanism achieves its goal of keeping the loss rate between 0 and 5% most of the time even though the loss rate in the network varies from 15 to higher than 40%. This clearly shows that our mechanism makes it possible to maintain good quality audioconferences even across fairly congested links in the Internet.

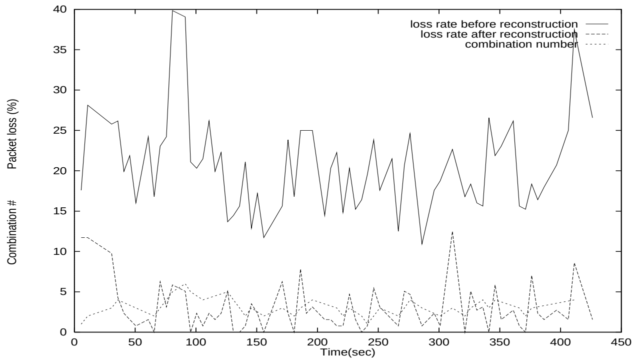


Figure 6: Evolutions of the loss rate at the destination (before and after reconstruction) and of the combination used by the control mechanism

## 5 Conclusion

We believe that the work presented in the paper suggests that it is possible to build an "Internet telephone" that can provide good quality audio even across fairly congested links. Of course, the load (and thus the delay and the loss rate) in the network are sometimes so high that audio delivered at the destinations is not intelligible. This situations seems difficult to avoid until better support is available from the network, and in particular until discriminatory scheduling disciplines such as Fair Queueing have been widely deployed.


### Acknowledgements

The work presented in the paper has been shaped by many discussions with members of the networking groups at INRIA and UCL. We would like to single out and thank Christian Huitema at INRIA, as well as Vicky Hardman and Mark Handley at UCL for many fruitful interactions. Thanks also to the anonymous reviewers for valuable comments and feedback. Jon Crowcroft at UCL and Vern Paxson at LBL provided machines that made some of the mea-

| | Combination | Bandwidth (kb/s) |
|---|---|---|
| 0 | (PCM) | 64 |
| 1 | (ADM6, ADM3 (1)) | 72 |
| 2 | (ADM6, ADM3 (2)) | 72 |
| 3 | (ADM5, ADM3 (1), ADM2 (2)) | 80 |
| 4 | (ADM5, ADM3 (1), ADM2 (3)) | 80 |
| 5 | (ADM4, ADM3 (1), ADM2 (2)) | 72 |
| 6 | (ADM4, ADM3 (1), ADM2 (3)) | 72 |
| 7 | (ADM3, ADM3 (1), ADM2 (2), ADM2 (3)) | 80 |
| 8 | (ADM3, ADM2 (1), ADM2 (2), ADM2 (3)) | 72 |
| 9 | (ADM2, ADM2 (1), ADM2 (2), ADM2 (3), ADM2 (4)) | 80 |
| 10 | (ADM2, ADM2 (1), ADM2 (2), ADM2 (4)) | 64 |
| 11 | (ADM2, ADM2 (1), ADM2 (3)) | 48 |
| 12 | (ADM2, ADM2 (1)) | 32 |
| 13 | (ADM2, ADM2 (2)) | 32 |

Table 3: Combinations and associated bandwidth requirements

surements possible. Early work on audio coding at INRIA was done by Thierry Turletti, Hugues Devries, and Hugues Crépin.

The stand-alone audio tool described in the paper will be released in 1996. Details on the tool and its performance are available in [5].

# References


[1] E. W. Biersack, "Performance evaluation of FEC in ATM networks", *Proc. ACM Sigcomm '92*, pp. 248-257, Baltimore, MD, Aug. 1992.

[2] J-C. Bolot, "End-to-end packet delay and loss behavior in the Internet", *Proc. ACM Sigcomm '93*, San Fransisco, CA, pp. 189-199, Aug. 1993.

[3] J-C. Bolot, T. Turletti, "A rate control scheme for packet video in the Internet", *Proc. IEEE Infocom '94*, pp. 1216-1223, Toronto, Canada, June 1994.

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

[5] http://www.inria.fr/rodeo/personnel/bolot/audio

[6] F. Bonomi, D. Mitra, J. B. Seery, "Adaptive algorithms for feedback-based flow control in high speed, wide area ATM networks", *IEEE JSAC*, vol. 13, no. 7, pp. 1267-1283, Sept. 1995.

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

[8] N. Chaddha, G. Wall, B. Schmidt, "An end to end software only scalable video delivery system", *Proc. NOSSDAV'95*, Durham, NH, pp. 139-150, April 1995.

[9] R. Cox, R. Crochiere, "Multiple user variable rate coding for TASI and packet transmission systems", *IEEE Trans. Comm.*, vol. 28, no. 3, pp. 334-344, March 1980.

[10] N. Erdöl, C. Castelluccia, A. Zilouchian, "Recovery of missing speech packets using the short-time energy and zero-crossing measurements", *IEEE Trans. Speech Audio Proc.*, vol. 1, no. 3, pp. 295-303, July 1993.

[11] S. Floyd, V. Jacobson, S. McCanne, L. Zhang, "A reliable multicast framework for lightweight sessions and application level framing", *Proc. ACM Sigcomm '95*, Cambridge, MA, pp. 342-354, Sept. 1995.

[12] 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.

[13] 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.

[14] R. D. de Iacoro, D. Sereno, "Embedded CELP coding for variable bitrate between 6.4 and 9.6 kb/s", *Proc. ICASSP'91*, Toronto, Canada, pp. 681-684, May 1991.

[15] E. I. Isaacs, J. C. Tang, "What video can and cannot do for collaboration: a case study", *Multimedia Systems*, vol. 2, pp. 63-73, 1994.

[16] N. Jayant, "Effects of packet losses in waveform-coded speech", *IEEE Trans. Comm.*, vol. 29, no. 2, pp. 101-109, Feb. 1981.

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

[18] S. Kallel, "Efficient hybrid ARQ protocols with adaptive forward error correction", *IEEE Trans. Comm.*, vol. 42, no. 2, pp. 281-289, Feb. 1994.

[19] S. McCanne, M. Vetterli, "Joint source/channel coding for multicast packet video", *Proc. ICIP '95*, Washington, DC, Oct. 1995.

[20] The MICE home page is at URL http://www.cs.ucl.ac.uk/mice/mice.html

[21] A. Mukherjee, "On the dynamics and significance of low frequency components of Internet load", *Journal of Internetworking: Research and Experience*, vol. 5, no. 4, pp. 163-205, Dec. 1994.

[22] H. Saito, "Optimal control of variable rate coding in integrated voice/data packet networks", *Performance Evaluation*, vol. 10, pp. 115-128, 1989.

[23] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A transport protocol for real-time applications", Internet draft, Audio-video transport working group, March 1995.

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

[25] 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.

[26] S. Shenker, C. Partridge, Internet drafts from the INT-SERV IETF working group, March 1995.

[27] T. Turletti, J-C. Bolot, "Issues with multicast video delivery over heterogeneous networks", *Proc. 6th Packet Video Workshop*, Portland, OR, Sept. 1994.

[28] F. Vakil, A. Lazar, "Flow control protocols for integrated networks with partially observed voice traffic", *IEEE Trans. Auto. Control*, vol. 32, no. 1, pp. 2-14, Jan. 1987.

[29] C. Weinstein, J. Forgie, "Experience with speech communication in packet networks", *IEEE JSAC*, vol. 1, no. 6, pp. 963-980, Dec. 1983.

[30] 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.