INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Experiments with a Layered Transmission Scheme over the Internet

Thierry Turletti   Sacha Fosse Parisis   Jean-Chrysostome Bolot

N° 3296

Novembre 1997

THÈME 1

Rapport de recherche

ISSN 0249-6399

Experiments with a Layered Transmission Scheme over the Internet

Thierry Turletti   Sacha Fosse Parisis   Jean-Chrysostome Bolot

Thème 1 — Réseaux et systèmes
Projets Rodeo

Rapport de recherche n° 3296 — Novembre 1997 — 26 pages

Abstract: Combining hierarchical coding of data with receiver-driven control appears to be an attractive scheme for the multicast transmission of audio/video flows in a heterogeneous multicast environment such as the Internet. However, little experimental data is available regarding the actual performance of such schemes over the Internet. Previous work such as that on receiver driven layered multicast uses join experiments to choose the best quality signal a receiver can subscribe to. In this paper, we present a receiver-based multicast rate control mechanism based on a recently proposed TCP-friendly unicast mechanism. We have implemented this mechanism and evaluate its performance in conjunction with a simple layered audio coding scheme. We find that it has interesting convergence and performance properties, but also bring out its limitations.

Key-words: Audio coding, congestion control, hierarchical coding, Internet, multicast transmission.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 93 65 77 77 - International : +33 4 93 65 77 77 — Fax : 04 93 65 77 65 - International : +33 4 93 65 77 65
à partir du 01/01/1998
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Etude et mise en oeuvre d'un schéma de transmission hiérarchique sur Internet

Résumé : La transmission audio/vidéo en multipoint dans un environnement hétérogène comme l'Internet soulève de nombreux problèmes. L'utilisation d'un schéma de transmission qui combine un codage hiérarchique et un contrôle de transmission orienté-récepteur apporte une solution élégante au problème de l'hétérogénéité des récepteurs. Cependant, peu de données expérimentales sont disponibles sur les performances effectives de tels schémas de transmission dans l'Internet. Dans l'une des approches existantes, la transmission hiérarchique orienté-récepteur, le récepteur choisit le nombre de couches auquel il s'abonne en fonction de d'abonnements test préalables. Dans cet article, nous décrivons un mécanisme de contrôle de transmission en multipoint orienté-récepteur basé sur un mécanisme *TCP-friendly* récemment proposé pour la transmission en point à point. Nous avons mis en oeuvre ce mécanisme en conjonction avec un schéma de transmission audio hiérarchique et nous évaluons ici ses performances et limitations.

Mots-clés : Audio, codage hiérarchique, contrôle de congestion, Internet, transmission multipoint hiérarchique

1 Introduction

The transmission of real-time audio and video over the Internet has been much in the news recently. In particular, the transmission of voice has achieved high visibility because of technical, financial, and regulatory issues related to so-called "Internet telephones" (e.g. refer to [17, 38]), to the rapidly increasing number of companies selling or giving away Internet telephony software, and to the increasing traffic generated by these telephones. Even though most of the commercial companies involved in the Internet telephony business date the beginning of Internet telephony to the first release of VocalTec's "Internet Phone" [37] in 1995, the transmission of unicast audio dates back to the 70's [39]*, while the transmission of multicast audio started "officially" at the March 1992 IETF [4]. However, it is true that only recently has audio and video traffic made up a non-negligible fraction of the traffic routed at some nodes in the Internet. In any case, this traffic is expected to increase for at least two reasons. First, a rapidly growing number of tools are available (some of them for free) that provide low bandwidth but decent to excellent quality audio and video coding [9]. Second, such tools are being incorporated in a growing number of applications such as collaborative working applications [9].

The wide use of unicast and multicast multimedia tools in the Internet raises two important related questions. First, what is the impact of a much increased multimedia traffic on other traffic? Second, what kind of multimedia quality can users expect given that the network does not provide guarantees on delay, jitter, bandwidth, or loss rate? The impact of multimedia traffic on network and application performance is potentially quite large because the vast majority of currently available tools send data at a rate which does not depend on network state. For audio tools, this rate would be a constant rate corresponding to the chosen coding scheme (64 kb/s for G.711 coding) in the absence of silence detection. Such a behavior is unlike that of rate controlled applications, such as TCP-based applications, which adjust their output rate and hence their bandwidth requirements depending on the state of the network. Thus, non rate controlled applications are "bad citizens" since they share network resources with TCP-based applications in unfair ways, and their rapidly increasing use in the Internet might cause long-lived, severe congestion problems (especially for TCP users).

There are two ways to prevent this from happening. One way is to replace the FIFO scheduling algorithms in Internet routers with RED-like algorithms that "punish" aggressive flows and reward rate controlled applications [13]. Another way is to incorporate in applications TCP-friendly rate control mechanisms which make sure resources are not shared unfairly with TCP connections, and suitable for both unicast and multicast transmission. This latter way can be thought of as a special case (namely rate adaptation) of a more general approach which aims at adapting application behavior to network characteristics, the goal being to maximize the quality of the data delivered to the destinations. Other

* Also see the discussion on the AVT mailing started on Sept. 16, 1996 by Ed Ellesson regarding a "Packetized Audio Patent".

kinds of adaptation include adaptation to delay jitter by playout adjustment schemes [29] or adaptation to loss by ARQ- or FEC-based error control schemes [11, 33, 28].

The second way above (namely incorporating rate control mechanisms in applications) does not require modifications to router software and hence can be implemented in the current Internet. However, none of the commercial tools we are aware of uses any kind of rate control †. This is not so surprising after all since non rate controlled applications tend to "steal" bandwidth from rate controlled applications, and thus provide better quality to their users (this will be illustrated in Section 4). This makes it all the more important that all applications be rate controlled since the current Internet does not provide incentives to behave in a fair way.

Unfortunately, the design of rate control schemes for multicast real time applications is not a simple extrapolation of the TCP-like source based control schemes used by unicast data applications for two main reasons. First, source based rate control schemes using feedback information about the state of the network do not scale well with the number of participants in the multicast group, implying that the rate of exchange of any information between participants needs to be carefully controlled [32, 3]. Second, source based schemes do not scale well with the level of heterogeneity in the branches of the multicast tree and/or in the participants. This is because the source can adjust the rate of a stream to match the requirements of one participant (i.e. to adapt to the state of one branch in the tree), but it cannot match the conflicting requirements of multiple heterogeneous participants (i.e. adapt at the same time to different states of different branches). Various approaches to this problem have been described in the literature, using gateways [27, 1], simulcast transmission [7], and layered transmission coupled with layered coding [22, 23].

We focus on the layered transmission / layered coding approach in this paper. While this approach is attractive and has been advocated for some time now, it has not yet been deployed over the Internet (although this is expected to change real soon now). The approach relies on the availability of two components, a layered coder and a layered transmission scheme. With *layered coding*, source data is encoded into a number of layers, or (sub)bands, that can be combined to reconstruct a signal that gets closer to the original signal as the number of combined layers increases. Layered coders for audio and video have been developed over the past few years, but few coders have low enough CPU requirements as to be useful in software only tools. Recent work has produced such low CPU layered video coders [23, 5]. However, we are aware of little equivalent work for audio coders. Thus, we have developed simple layered schemes for audio based on the idea of temporal subsampling, which are described in Section 2.

The idea of *layered transmission* then is to send each layer on a separate multicast group, with receivers deciding to joint/quit a group (i.e. to receive/drop another layer) on their own. The basic principles of layered transmission have been known for a while (see references in [22]). However, a specific scheme specifying how/when receivers are to join and quit layers has only recently been described and simulated. This scheme, referred to as

† It is hard to blame the commercial tools for behaving so as they only mimic research tools in this respect. None of the more popular MBone tools such as VAT[36] uses any kind of rate control either...

Receiver driven Layered Multicast (RLM), uses "probing" experiments similar to those used by TCP, to decide when to join and quit layers. We propose a layered transmission scheme in which RLM's join experiments are replaced by an explicit estimation at each receiver of the bandwidth that would be used by an equivalent TCP connection (the notion of equivalent connection will be defined later) between the source and the receiver. This estimation is done using the same technique as that recently described to design TCP-friendly unicast rate control scheme [20]. Thus, we obtain a TCP-friendly receiver-based multicast rate control mechanism. We have implemented this scheme and evaluated its performance and limitations on the MBone.

The rest of the paper is organized as follows. In Section 2, we describe simple layered audio coding schemes. In Section 3, we describe the receiver-based control scheme. In Section 4, we describe the experimental settings and discuss the results. Section 5 concludes the paper.

2 Simple layered audio coding schemes

Layered coding is a family of signal representation techniques in which the source information is partitioned into a sets called layers. The layers are organized so that the lowest, or base layer, contains the minimum information for intelligibility. The other layers, called complementary layers, contain "add-on" information which improves the overall quality of the signal. Usually, layered encoding schemes are organized so that some layers (in particular the base layer) are mandatory to reconstruct a coherent signal. Using such schemes then requires either that the net be able to discriminate between packets and provide packets that carry important information with a guaranteed performance (in particular guaranteed maximal loss rate) service, or that these packets be "protected" against loss. This can be done using a variety of so-called unequal error protecting schemes, which have been the subject of much research effort recently (e.g. [16, 10].

Our goal was not to develop or experiment such schemes, but instead to experiment with rate control schemes. Thus, we have focused on simple coders with low cpu requirements and no need for unequal error protection schemes. We describe next a simple layered encoding scheme in which all layers have the same importance, meaning that the loss of information from one layer does not impact quality more than the loss of information from another layer.

2.1 The basic subsampling scheme

The simplest balanced scheme one can think of is based on straight subsampling The coding algorithm is based on a temporal subband decomposition. Consider for example the case of a PCM (Pulse Coded Modulation) encoded 8kHz audio signal decomposed into 3 PCM 2.7 kHz flows as shown in Figure 1. The temporal decomposition algorithm is carried out at the source done for each audio chunk (which typically includes 20ms, 40ms or 80ms of audio). At a destination, a receiver which receives all 3 flows can retrieve the original input signal. If one or two flows are missing, the destination uses the samples received to reconstruct an

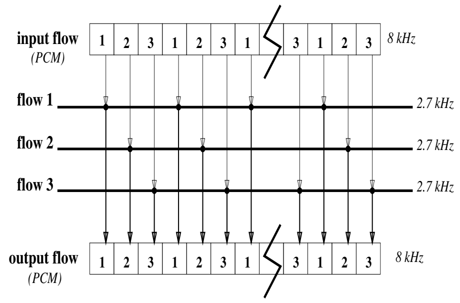

Figure 1: Hierarchical coding scheme

approximation of the original signal. An upsampling one-to-three flows is shown in Figure 2. Clearly, the larger is the number of received flows, the better the quality of the reconstructed signal.

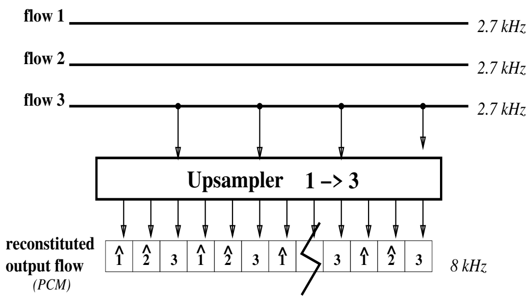

Figure 2: Reconstruction 1 to 3

Note that temporal decomposition handles signals sampled with different sampling rates. For example, a 48 kHz audio signal can be decomposed into three 16 kHz subflows, each of which can be decomposed into two 8 kHz subflows, which finally yields eighteen 2.7 kHz audio layers. Of course, temporal subband decomposition can be coupled with compression schemes[‡].

2.2 Robustness against packet loss

The audio coding scheme described above has the interesting balanced property mentioned above: all the layers have the same importance in the network. However, in case of severe congestion, the congestion control algorithm may select to subscribe to only one flow and

[‡] Refer to the Web page http://www.inria.fr/rodeo/turletti/audio/ to retrieve samples showing the impact on quality of the number of received layers, and of the compression scheme used to encode the layers

in this case, the coding has no mechanism at all to reconstruct lost samples of audio. To achieve robustness against packet loss, the receiver could decide to subscribe to at least 2 flows. We have decided to add to the base flow of each sample, the flow L corresponding to the previous sample (L is the number of layers of the coding). So, in such a scheme, the first layer has twice the bandwidth of other layers and the transmission control scheme handles $(L - 1)$ flows instead of L. Note that contrary to other redundancy schemes, there is no bandwidth wasted when packet loss is null since the "redundancy" information appended to the first layer is always used to improve the reconstruction of the sample.

Note also that the redundancy added to flow # 1 does not imply that this flow has a higher priority since flow # 1 is not required to receive it correctly to decode the other flows. Whereas there is no flow more important than another one, all receivers have to adopt the same order in the subscription algorithm (in order to perform efficient pruning to limit the bandwidth sent).

2.3 CPU and bandwidth cost

The cpu cost of the hierarchical coding algorithm mainly depends of the compression scheme used following the temporal subband decomposition. In our experiments, we have successfully tested the PCM, ADM6, ADM5, ADM4, and ADM3 compression schemes, which have very low cpu requirements [14]. More efficient compression coder could as well fit into the temporal subband decomposition described above. However, very high compression coding algorithms do not appear to fit well with layered coding over the Internet because the overhead of IP, UDP and RTP[32] headers decrease significantly their bandwidth savings.

Table 1 shows the bandwidth overhead for layered and non-layered audio coding using PCM and ADM4 compression schemes. The IP/UDP/RTP bandwidth headers overhead is shown for three sizes of packets corresponding to 20ms, 40ms and 80 ms of compressed speech.

Interactivity needed by audio/video conferencing applications is often cited as requiring low packetization intervals of 20ms. However, such a low packetization interval increases the number of packets sent per second, especially if several layers are used. Then the bandwidth requirement of the IP/UDP/RTP headers may be higher than the bandwidth requirement of the actual payload, see Table 1. For example, the bandwidth corresponding to the headers for 6 flows is (for 20ms packetization interval) equals to 106 kbps with IPv4 and 144 kbps with IPv6. Table 2 recalls the size of IPv4, IPv6, UDP and RTP headers.

However, experience with audioconferencing tools in the MBone shows that 40 ms and 80 ms are convenient values especially if redundant techniques [28] are used to minimize the effect of large "holes" in the speech. 80 ms appears to be a good compromise between interactivity needs and bandwidth overhead generated by the hierarchical coding.

In some cases (e.g. very low bandwidth links), the IP/UDP/RTP headers may be compressed to reduce the bandwidth overhead [8]. However, such techniques increase the CPU consumption at the routers and they are expected to be used on point to point links, but not on an end-to-end basis on the MBone.

Speech Coding/ Transmission	Payload (kbps)	20ms buffer Total (kbps) IPv4/IPv6	20ms buffer Efficiency ($\frac{payload}{header}$) IPv4/IPv6	40ms buffer Total (kbps) IPv4/IPv6	40ms buffer Efficiency ($\frac{payload}{header}$) IPv4/IPv6	80ms buffer Total (kbps) IPv4/IPv6	80ms buffer Efficiency ($\frac{payload}{header}$) IPv4/IPv6
PCM	64	81.6/88.0	3.64/2.67	72.8/76.0	7.27/5.33	68.4/70.0	14.55/10.67
ADM4	32	50.1/56.5	1.82/1.33	40.8/44.0	3.64/2.67	36.4/38.0	7.27/5.33
PCM 1 fl. 2.7kHz	21.3	38.9/45.3	1.21/0.89	30.1/33.3	2.42/1.77	25.7/27.3	4.84/3.55
PCM 3 fl. 2.7kHz	64	117/136	1.21/0.89	90.3/100	2.42/1.77	77.7/81.9	4.84/3.55
ADM4 1 fl. 2.7kHz	10.7	28.3/34.7	0.61/0.45	19.5/22.7	1.22/0.89	15.1/16.7	2.43/1.78
ADM4 3 fl. 2.7kHz	32	84.9/104	0.61/0.44	58.5/68.1	1.22/0.89	45.3/50.1	2.43/1.78
Overhead (1 fl.)	N/A	17.6/24.0	N/A	8.8/12.0	N/A	4.4/6.0	N/A
Overhead (3 fl.)	N/A	52.8/72.0	N/A	26.4/36.0	N/A	13.2/18.0	N/A
Overhead (6 fl.)	N/A	106/144	N/A	52.8/72.0	N/A	26.4/36.0	N/A

Table 1: Bandwidth overhead for several audio coding/transmission schemes

Overhead per packet	IP v4	IP v6	UDP	RTP
(in bytes)	24	40	8	12

Table 2: IP/UDP/RTP header size

2.4 Implementation details

The hierarchical encoded flows are carried as payload data within the RTP protocol [32]. The application follows the recommendations defined in the “RTP usage with layered multimedia streams” draft [35]. There is no need to manage a different sequence numbering per flow. So, in our hierarchical audio application, we further impose that all flows use the same sequence number to encode an audio sample. The receiver handles one circular buffer for all the layers, so only one playout is computed using the algorithms described in [29].

3 The receiver-based control scheme

3.1 The RLM scheme

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

The receiver detects network congestion when it observes increasing packet losses. In the absence of loss, the receiver estimates spare capacity, or rather the existence of spare capacity, with so-called join experiments. A join experiment means that a receiver joins the next group and measures the loss rate over an interval referred to as the decision time (to avoid the synchronization of join experiments, experiments are carried out at randomized times). However, the load created by join experiments increases as the size of the multicast group increases. To prevent a load explosion, the authors in [22] use "shared learning", in which a receiver about to start a join experiments multicast its intent to the group. Thus all receivers get to know the result of join experiments carried out by other receivers. To prevent join experiment for different layers to interfere with each others, only receivers that are doing join experiments for layers equal or below a newly advertised experiment can actually conduct their own experiments. RLM with shared learning scales with the size of the group, but it raises a few questions. In particular, it is not clear how effective shared learning is in the absence of knowledge about the structure of the multicast delivery tree. Furthermore, shared learning increases the convergence time to steady state layer subscription especially for receivers with spare capacity (since they will have to wait for slow receivers to reach their steady state before they can join additional upper layers). Also, the times at which layers are added and dropped determine the rate increase and decrease along the branches of the tree. Thus, RLM should choose these times appropriately if it wants to achieve fairness with TCP traffic. However, this appears hard to do in practice.

3.2 Basic idea

Our control scheme aims at providing the benefits of the RLM scheme without (at least some of) the costs associated with it. The scheme uses the same idea of a receiver based rate control scheme, however join experiments are replaced by an explicit estimation at each receiver of the bandwidth that would be used by an equivalent TCP connection between the source and the receiver. This estimation is done using the same technique as that recently described to design TCP-friendly unicast rate control scheme [20]. Specifically, the bandwidth λ_{equ} of a TCP connection can be well represented by Equation 1 below [20, 21, 25]

$$\lambda_{equ} = 1.22 * \frac{MTU}{RTT * \sqrt{Loss}} \tag{1}$$

where MTU is the maximum packet size used on the connection, RTT is the mean round trip time, and $Loss$ is the mean packet loss rate. Now assume that each receiver knows the rate λ_i of the layer i flow generated by the source over the multicast tree. Then, each receiver executes the following algorithm:

Step 0: Upon joining the group, subscribe to the base layer

Step 1: Measure or estimate MTU, RTT, and $Loss$

Step 2: Compute λ_{equ}

Step 3: Find L, the largest integer such that $\sum_{i=1}^{L} \lambda_i \leq \lambda_{equ}$. Join the first L groups.

Note that the rate at which data flows between the source and any receiver is equivalent to that of a TCP connection running over the same path. Thus, we have obtained a TCP-friendly scheme suitable for multicast delivery. Furthermore, note that the scheme does not rely on active probing schemes such as join experiments, nor does it require exchange of information between participants as is done with shared learning. We examine below details associated with steps 1 and 3 above (e.g. the estimation of parameters in step 1). However, before doing this, it is worth going back to Equation1 and note that i) the equation in fact only provides an *upper bound* to the rate of an equivalent TCP connection, and ii) however it has been shown to fit well with measured and simulated performance of a wide variety of TCP variants [13].

3.3 Parameter estimation

Step 1 in the list above deals with estimating network parameters. So let us consider all of them in turn.

MTU: The MTU value may be set to the the minimum acceptable value for TCP of 576 bytes or could be determined using the MTU discovery algorithm [24].
Loss: The mean loss rate $Loss$ is easily computed from observed losses (detected using the RTP sequence number field) using an exponential filter with a half life equal to n packets (or equivalently to $d * n$ sec, where d denotes the inter-packet arrival). The trick of course is to choose an appropriate value for n. We use a value that varies over time as shown in [22].

RTT: The mean round trip delay RTT cannot be computed as easily as $Loss$ since the information received from the source in RTP SRs (*Sender Reports*) only refers to the one-way delay between source and receiver. Unfortunately, the one way delay can be a poor estimate of $RTT/2$ [6, 26]. There are 3 ways to tackle this problem. The first way is to estimate RTT using twice the delay from source to receiver (which is known using RTP timestamps), and assume a symmetric path. However, this clearly breaks down, in particular with asymmetric links (such as satellite links). The second way is to augment the capabilities of RTP so as to be able to estimate the actual round trip delay. this can be done as follows: the receiver sends an rtt-request packet; the sender replies immediately with an rtt-reply packet to the receiver; the receiver notes the delay corresponding to the current RTT value and updates the RTT estimator with it. Unfortunately, this scheme does not scale to large multicast sessions. First, the frequency at which the receiver generates the rtt-request packets should be function of the session size (as with RTCP messages [32]) in order to avoid the well-known feedback implosion problem. Second, the period between two rtt-request packets should not be constant in order to avoid RTT synchronized requests and periodic congestion [12]. Third, unicast rtt-requests from the sender may be less efficient than multicast joint rtt-requests. We could have used the SR and RR (*Receiver Report*) packets to append the information required to estimate RTT. However, these packets include several parameters that do not need to be sent as often than the RTT information. So, in order to save bandwidth, we have created two new RTCP packets which include the minimal information needed to estimate RTT: *rttreq* for the rtt-request packets and *rttrep* for the rtt-reply packets sent by the sender. *rttreq* packets include a 4-bytes RTCP header which identifies the packet and encodes the packet length, the SSRC of the receiver, and the time at which the rtt-request has been sent at the receiver ts_{rttreq}. *rttrep* packets include the 4-bytes RTCP header, the SSRC of the sender and a set of RTT information report blocks corresponding to each *rttreq* packets received since the last *rttrep* report. Each RTT information report block includes SSRC of the receiver, the corresponding t_{rttreq} parameter and the delay δ in millisecond between the time at which the *rttreq* has been received at the sender and the time at which the *rttrep* is sent to the group, see Figure 3.

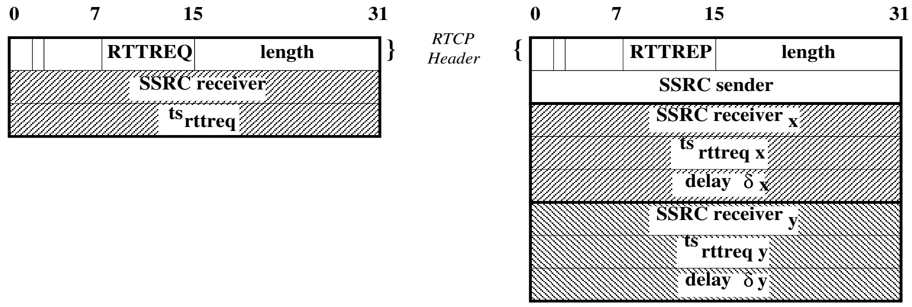

Figure 3: rttreq and rttrep packets

As soon as a receiver receives an *rttreq* report, it notes the corresponding receiving time tr_{rttrep}. Then, if its SSRC is included in the packet, it is able to update the estimation of the RTT with the new value:

$$RTT = tr_{rttrep} - ts_{rttreq} - \delta \qquad (2)$$

Note that this mechanism does not assume that sender and receivers clocks are synchronized.

Let us now examine how this parameter estimation mechanism scales with the number of receivers N in the session. To do this, we compute next the interval of time T_{RTT} between two RTT estimations for a receiver in the session. Assume that r % of the payload bandwidth BP is reserved for this usage.

$$BP_{RTT} = \frac{W_{rttrep} + N.W_{rttreq}}{T_{RTT}} \qquad (3)$$

with W_{rttrep}, the size of the $rttrep$ packet and W_{rttreq}, the size of the $rttreq$ packet. From Figure 3, we have $W_{rttrep} = 64 + 96.N$ bits and $W_{rttreq} = 96$ bits. So, using equation 3:

$$T_{RTT} = \frac{64 + 192.N}{BP_{RTT}} \qquad (4)$$

Figure 4 shows the delay between two RTT updates with N in $[1, 200]$ with the following parameters: $BP_{RTT} = 64kb/s$, $r = 5\%$. With $N = 16$ receivers, the RTT estimate is

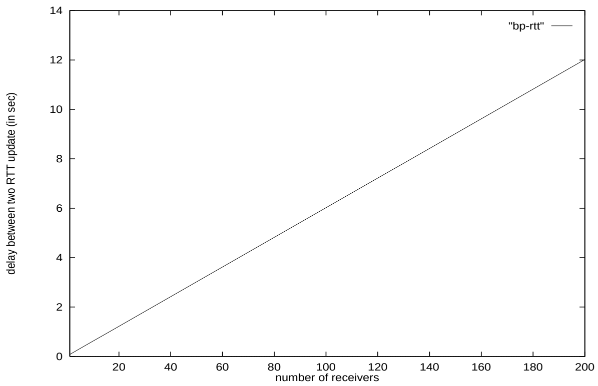

Figure 4: Delay between two RTT estimations (in sec) vs N

updated about each second. When the number of receivers reaches 100, the interval of time increases to 6 seconds. Clearly our scheme is not convenient for large groups since it suffers from the same scaling problems than RTCP [31]. All the more so since some of the $rttreq$ and $rttrep$ packets can be lost during transmission.

This has head us to examine a third way to estimate RTT, which in fact attempts to bypass RTT entirely in Equation 1. The goal there is to obtain a relation for λ_{equ} in ways similar to those described in [25, 19] that does not involve RTT, but rather the source to destination delay and the inter-packet jitter observed at the destination. The idea is to measure RTT infrequently (e.g using the scheme above), then to track variations in the one-way source-destination delay using RTP timestamps, and to track variations in the one-way destination-source delay using the jitter at the destination. Indeed, variations in the destination-source delay are translated in the source variations in the times at which packets are sent and thus into jitter at the destination. Unfortunately, we have not been able to derive an approach formula for λ_{equ} yet, and this remains an area for future research.

A fourth approach, which we only mention, is to forget about TCP-friendliness altogether and instead wait for schemes such as RED to be deployed in the Internet....

3.4 Control algorithm

At the beginning, when a receiver joins a session, it subscribes to the base layer in order to estimate the parameters as described above during a time interval T_i. Once parameters have been estimated, Equation 1 is used to estimate the rate (λ_{equ}) of an equivalent TCP connection. The value of λ_{equ} in turn indicates how many layers the algorithm may subscribe to so as to behave (at least throughput wise) like a TCP connection. At the initialization, the T_i time interval is set to the length corresponding to the exponential filters (for RTT and $Loss$ parameters). Then, we ensure that the T_i interval be at least greater than the mean RTT value estimated in order that the receiver can detect the resulting impact of the latest action performed.

A receiver actually subscribes to extra multicast group(s) when it is allowed to do so. Once this is done, it resumes the observation and estimation of the parameters so as to determine the impact of the join on network congestion. The receiver won't take any actions until it can estimate the effect of its latest action. This period of time equals to the maximum between the estimated RTT and the delay corresponding to the filter length.

The situation is similar, in the opposite direction, when the equivalent rate tells the receiver to drop a layer. Finally, if the equivalent rate is less than the minimal rate the application can send (i.e. the base band rate), the algorithm should pop up a message to the receiver and invite (or force) him to leave the session in order not to congestion the network and swamp all the network resources. – Internet collapse could happen for example if a large number of receivers continue to subscribe to the minimal quality flow (i.e. no more rate decrease in case of high packet loss observed). To follow the behavior of TCP [18], the application can wait for a delay T_r before probing the network state by joining the first layer. $T_r = 2^k + \sigma$, where k is the number of consecutive failed attempts, and σ a random value added to avoid synchronization of receivers.

4 Experimental evaluation over the MBone

The layer codec and the control which described above have been implemented in an experimental version of the FreePhone audio tool [15]. Packets are sent using the usual IP/UDP/RTP stack. All the measurements results presented in the paper have been done with a hierarchical and non-hierarchical ADM4 coder. Audio packets sent by the coder include speech samples corresponding to 80 ms of speech. They are sent at regular 80 ms intervals while the audio source is active.

We have carried out measurements over several connections including local connections (between machines located on the local area networks at the sender side) as well as wide-area connections. In the long distance experiments over the MBone, we set the source at MIT (Massachussetts Institute of Technology) and receivers are MIT, INRIA (in France), UCL (University College London in the UK), and UTS (University of Technology in Australia).

The MBone delivery tree between source and destinations are shown in Figure 5.

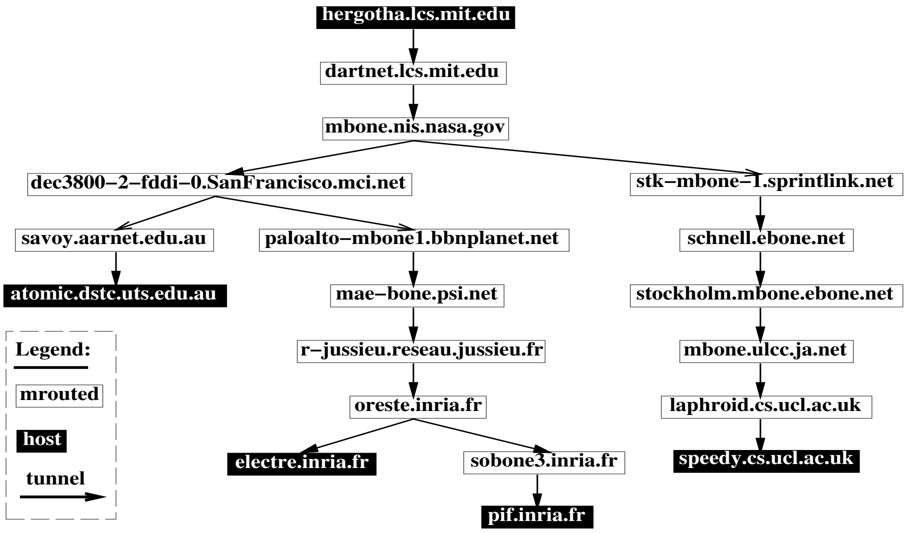

Figure 5: MBone delivery tree between sender at MIT and receivers

During experiments with non-controlled rate flows, we have restricted the number of layers to 3 in order to limit the network congestion in the network. The experiments done with the congestion control scheme enabled are allowed to use more flows.

4.1 Influence of the $shift$ parameter

In the two following experiments, we examine the characteristics of the data received on each flow, according to the way they have been sent at the sender side (per burst or at regular time interval, i.e. using the $shift$ parameter). In the first experiment, packets are sent per burst in the 3 flows at each 80ms interval, whereas in the second experiment, the packets from flow 2 and 3 are delayed in such a way that the overall transmission is periodic. In both experiments, a total of 32 kbps of payload (corresponding to an ADM4 8kHz flow) has been split into 3 flows sent in 3 different multicast group during 7 minutes (which roughly corresponds to 5000 packets).

Table 3 shows the values of RTT (in ms) and $Loss$ (in %) corresponding to the 3 receivers. We note that in the second experiment, the values of RTT and $Loss$ measured at a receiver are essentially constant. However, in the first experiment, and especially when the network is loaded, RTT and $Loss$ increase with the flow number[§]. To explain this phenomenon, we can use the cumulated loss (CL) rate parameter which represents the probability to loose all the packets (from different flows) corresponding to a sample. The cumulated loss rate is computed at the receiver side, after reconstructing all samples with packets received from all the flows. The results are shown in Table 3 for the two experiments. The cumulating loss rate for the 3 flows received is about eight times lower in the first experiment than in the second one. This means that the higher packet loss rate observed in the second experiment is mainly due to the loss of consecutive packets sent. The low value obtained in the first experiment (less than 0.5 %) clearly shows that this hierarchical coding scheme uses efficiently each flow as a redundancy for the others flows received. We have plotted in Figures 6, 7 and 8, the evolution of the cumulating loss rate vs the sequence number (or sample number) corresponding respectively to INRIA, UCL and UTS receivers. The left and the right figures show the cumulating packet loss rate when the packets are sent respectively per burst and at regular time interval, i.e. with $shift$ option enabled. We can observe that at each receiver, the probability to loose 3 consecutive packets (i.e. an sample sent in 3 flows) is very low.

So the sender's application needs to include a mechanism to avoid sending bursts of packets in different flows (in the same way an efficient non-hierarchical sender's application acts). Note that the periodical packet loss observed at UTS (Figure 8 shows three consecutive packet loss roughly every 370 packets, which corresponds to 30 seconds) is likely due to the router's synchronism bug in which routers periodically exchange their routing table over the network [12].

Table 3 also shows that by using the $shift$, the mean RTT value and the mean packet loss rate are nearly constant for the three flows. This makes a nice point about the scalability of the scheme since we can choose a single flow to estimate the parameters for the receiver's congestion control algorithm. In the rest of the paper, the estimation of the RTT and packet loss rate parameters is done on the base flow and experimentations are done using the $shift$ option enabled.

4.2 Evaluation of the congestion control algorithm

Once we have studied the correlations of flows on the MBone, we can evaluate the congestion control algorithm described in section 3.

In a third experiment, the source at MIT sends 5 flows corresponding to the hierarchical audio coding rate described in section 2. The compression algorithm selected is ADM4 and the audio flow is originally at 16 kHz (the output flows corresponds to two ADM4 / 8 kHz audio see Table 1.). The receivers implement the congestion control algorithm described in section 3 to decide how many flows they are allowed to subscribe to.

[§] We have noticed that the difference increases with the network load.

Receiver	flow #	with shift			without shift		
		RTT (ms)	loss (%)	CL (%)	RTT (ms)	loss (%)	CL (%)
INRIA	1	301	6.0	6.0	306	5.9	5.9
	2	301	5.9	1.2	309	7.5	3.6
	3	301	6.3	0.4	313	9.0	3.1
UCL	1	357	4.4	4.4	360	3.9	3.9
	2	358	4.1	0.3	363	4.3	1.2
	3	358	5.0	0.1	367	4.6	0.8
UTS	1	308	0.9	0.9	308	1.1	1.1
	2	307	0.9	0.2	312	1.1	0.4
	3	308	0.7	0.1	312	0.9	0.3

Table 3: RTT, mean and cumulated packet loss intercorrelation between flows

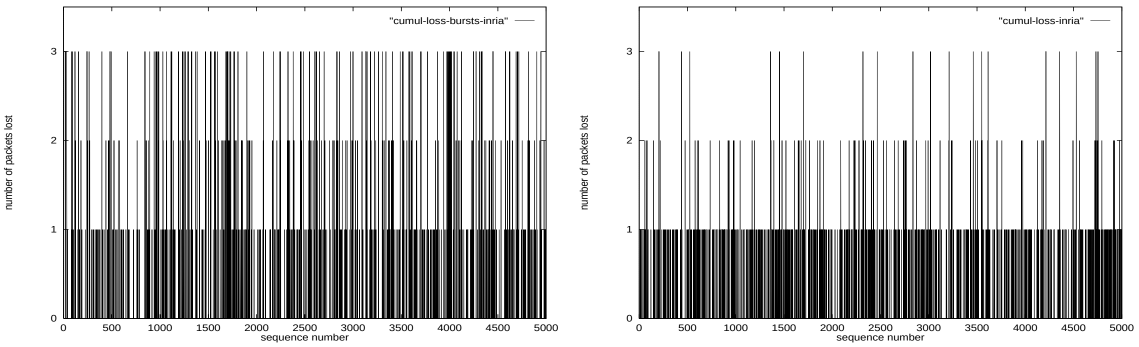

Figure 6: Cumul loss rate versus sequence number at INRIA

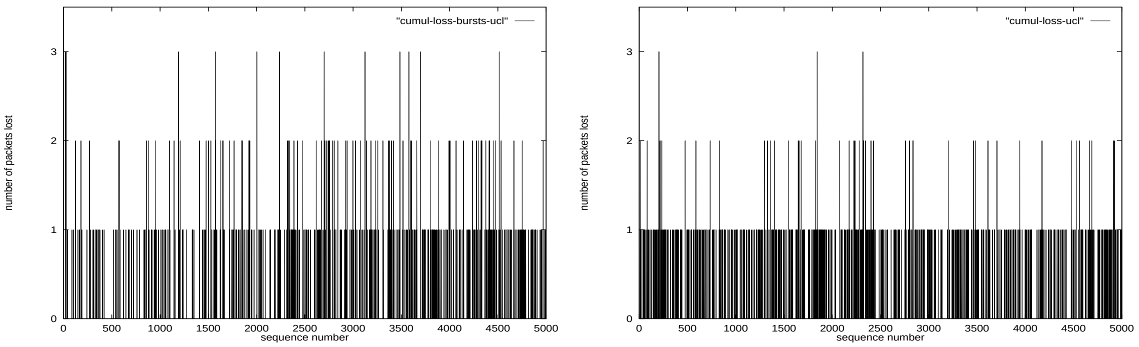

Figure 7: Cumul loss rate versus sequence number at UCL

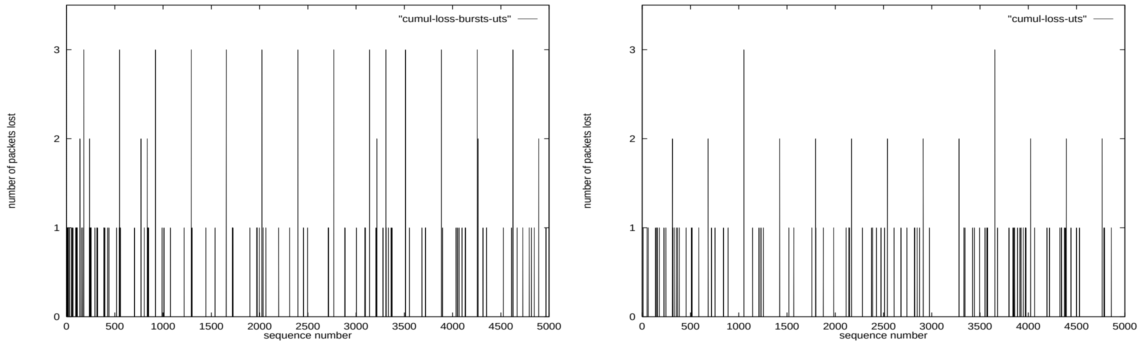

Figure 8: Cumul loss rate versus sequence number at UTS

Figures 9 and 10 show the evolutions of the RTT and the packet loss rate estimators respectively according to the sequence number (corresponding to the audio sample number) for both UTS and UCL receivers. The corresponding maximal rate estimation allowed and the subscription decision is plotted in Figure 11. The results from the local receiver at MIT have not been plotted but show as expected a very low RTT value and a null packet loss rate which set the level of flow subscription to its maximal value (i.e. 5).

As expected, we observe that during periods of congestion (with higher RTT and packet loss rate estimates), the algorithm selects a lower number of flows. When the network is unloaded, and as soon as the first estimations of the RTT and the packet loss rate are available, the algorithm can select the higher rate it is allowed to receive (e.g. for the UCL's receiver). The algorithm can also decide to drop several flows at the same time in case of severe congestion.

In the fourth experiment, we examine how multiple sessions interact while running in a same network. We have run two different multicast sessions over a low-bandwidth PPP link¶ set between two PCs and used only by these sessions. The two sessions use an ADM4 (32 kb/s payload) coding at 8 kHz sent over 3 layers with 80 ms of speech encoded per packet. Figure 12 shows the subscription decision and the rate received for the two different sessions. We note that both sessions have roughly the same behaviour, i.e. the subscription decision oscillates between 1 and 2 flows and the mean bandwidth received is about 12 kb/s for session 1 and 13 kb/s for session 2. In case of several sessions are run under the same network conditions (i.e. same packet loss and RTT), we observe that each session gets roughly the same amount of the total network capacities. During tests, we have also noticed that the PPP link is more likely to loose bursts of packets rather than isolated packets such as common in the MBone. Since the measure of quality of the coding depends on the packet loss properties in the network, the following tests have been done on the MBone, even if we cannot ensure the same network conditions for two consecutive experiments.

¶ The maximal UDP bandwidth measured over this low-rate link was 83 kb/s using a packet size equal to the MTU=1024 bytes.

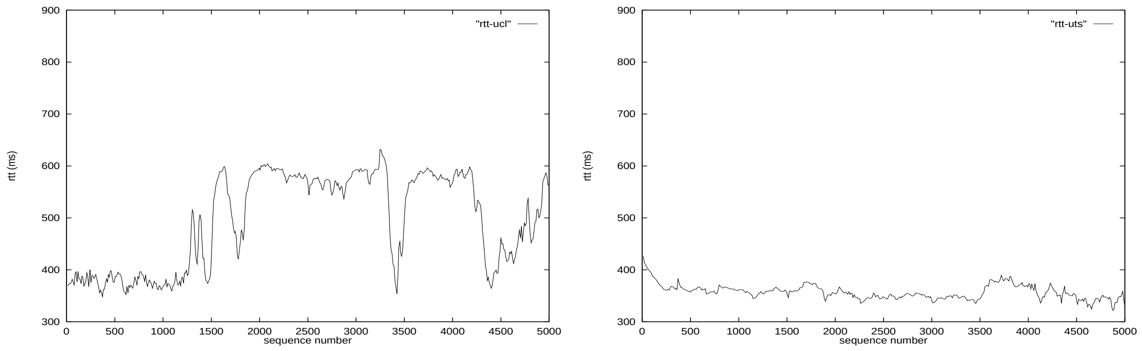

Figure 9: RTT versus sequence number at UCL (left) and UTS (right)

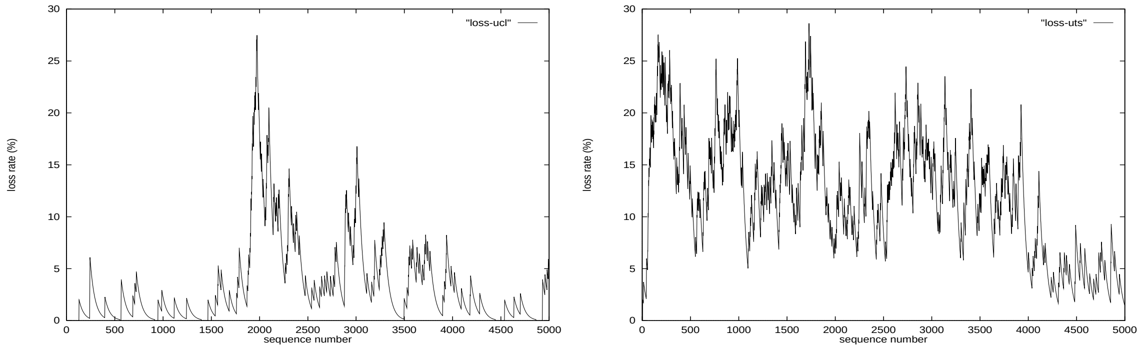

Figure 10: Loss rate versus sequence number at UCL (left) and UTS (right)

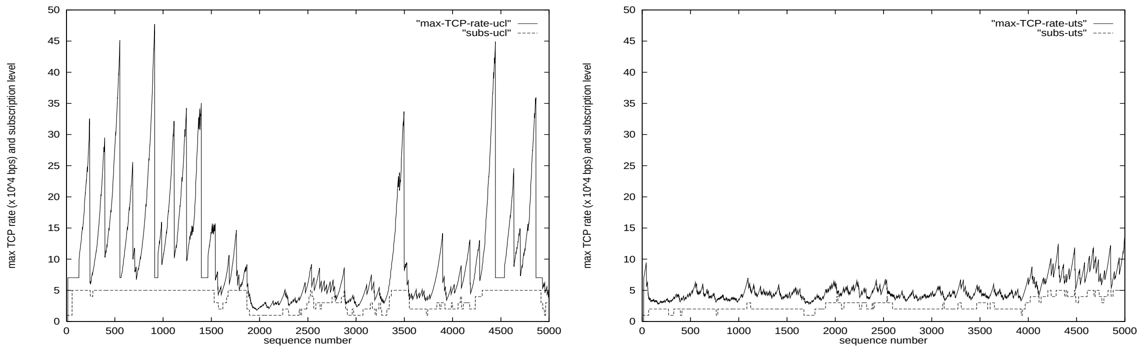

Figure 11: Subscription level versus sequence number at UCL (left) and UTS (right)

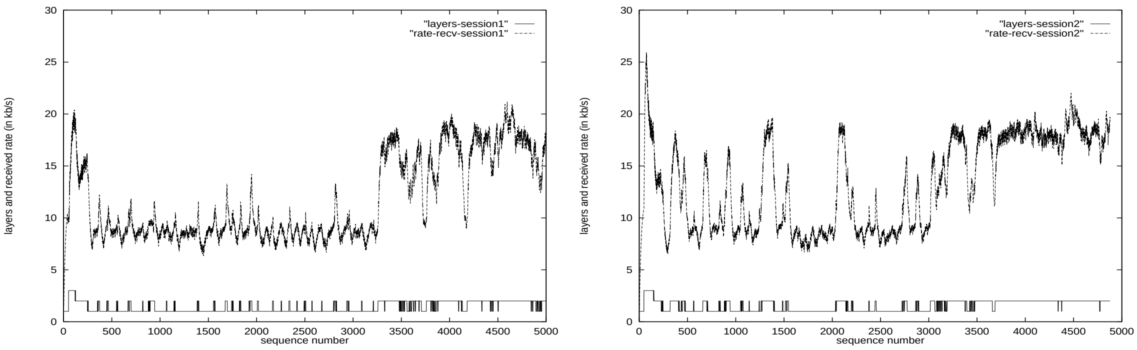

Figure 12: Flows subscribed and Received rate for two different sessions over PPP

4.3 Comparison with "traditional" schemes

It is interesting to compare our scheme with the transmission scheme used in "traditional" applications such as vat. In the fifth experiment, the source at MIT sends the equivalent ADM4 compressed 16 kHz data in one flow. This experiment has been done a couple of minutes after the third one, and not in the same time to limit our sending rate on the MBone. Of course, network conditions can be slightly different between two experiments, so we cannot simply use the rates observed in both experiments to compare quality obtained.

Using this layered coding scheme in conjunction with a rate control transmission algorithm will achieve a variable audio quality according to the network conditions. Further tests are needed with real users in order to observe their behavior during an audio conference. During a session the audio quality may sometimes appear "metallic" when only one flow is received and may sometimes be HiFi (e.g. all flows received corresponding to a 32 kHz / PCM audio flow). The best way to measure the quality of the audio flows received would have been to use a MOS measurement [34]. However, there is no such measure available yet for the experimental PCM and ADM hierarchical encoded schemes described in section 2. To coarsely quantify the quality obtained at receivers, we use the instantaneous rate received for each sequence number. The quality of the audio signal increases with the number of packets received by sample. The instantaneous quality is null when the sample cannot be reconstructed and is maximal when all the packets constituting the sample are received.

Figures 13 and 14 show respectively the evolutions of the packet loss rate and the corresponding quality observed at UCL and UTS for the non-hierarchical sending experiment (fifth experiment).

We note that in the fifth experiment, the quality received instantaneously is either minimal (i.e. sample lost) or maximal (sample totally reconstructed). Some of the receivers (e.g. UTS) receive lots of "blanks" (e.g. 1 loss encodes 80ms, 5 consecutive losses 0.4 second) which make the audio flow sometimes unintelligible. Audio samples corresponding to several experiments with various packet loss rates can be retrieved at *http://www.inria.fr/rodeo/turletti/audio/* so as to be able to listen to the quality of our scheme versus "traditional schemes". Moreover,

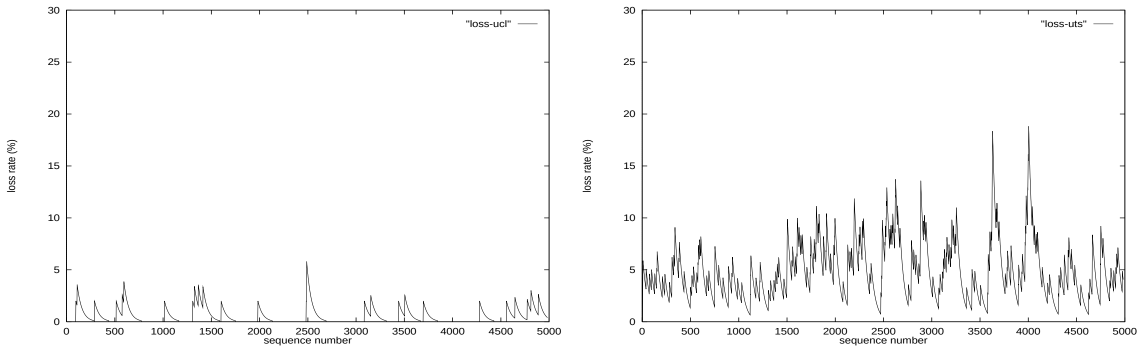

Figure 13: Packet loss rate for non hierarchical flows at UCL (left) and UTS (right)

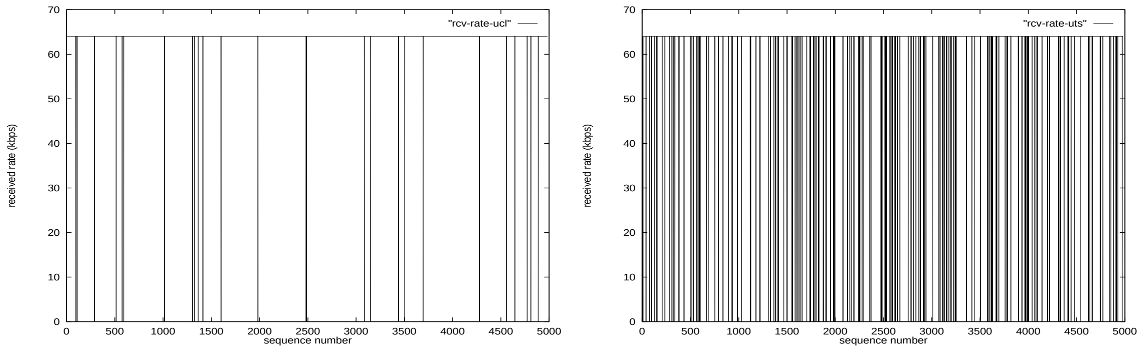

Figure 14: Received rate for non hierarchical at UCL (left) and UTS (right)

the output rate sent in the network does not back off and is always set to its maximal value (i.e. 64 kbps), yielding a "bad citizen" behavior which may contribute to congestion in the Internet.

Now let us examine how our layered transmission scheme behaves in presence of packet loss. Figures 15 and 17 show respectively the evolutions of the packet loss rate and the corresponding quality observed at UCL and UTS for the third experiment.

In period of high congestion (when the receiver subscribes to only one flow), the robustness of this hierarchical coding scheme allows to reconstruct all the packet lost at UCL and UTS. Clearly, this will not be the case if congestion is yet more severe, but in this case, the maximal subscription rate allowed could overrun the minimal sending rate of the application (e.g. here 21 kb/s of payload) and the receiver should be invited to quit the session and restart later.

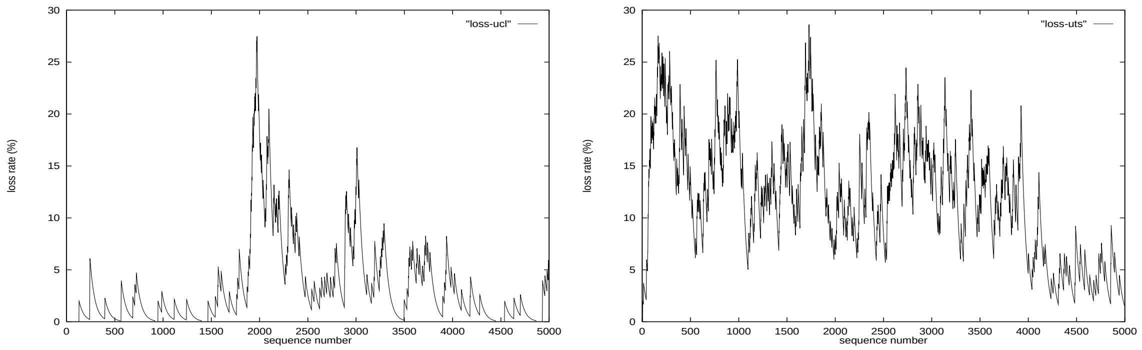

Figure 15: Packet loss rate for hierarchical flows at UCL (left) and UTS (right)

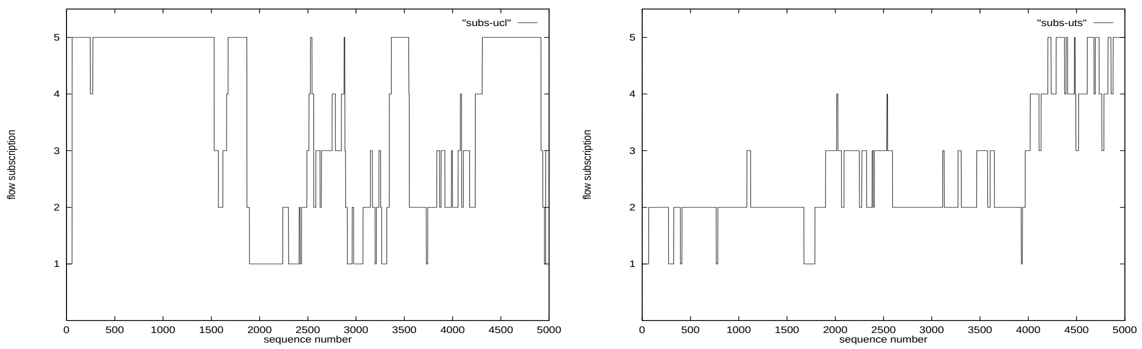

Figure 16: Subscription level for hierarchical flows at UCL (left) and UTS (right)

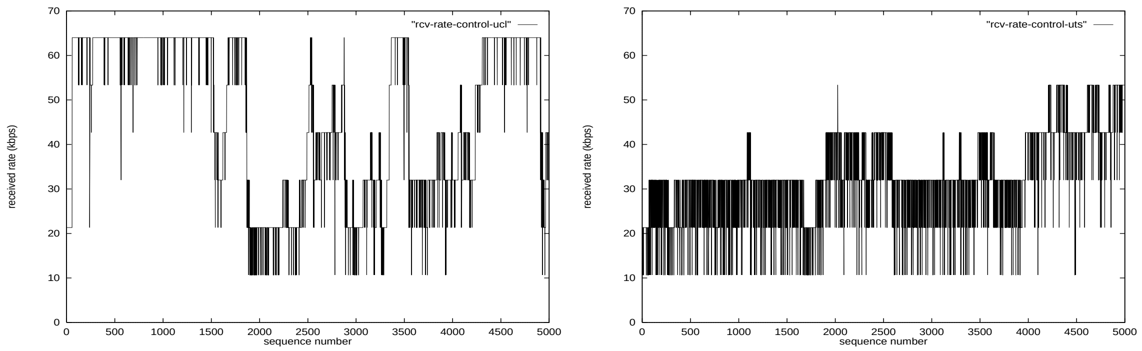

Figure 17: Received rate for hierarchical flows at UCL (left) and UTS (right)

5 Conclusion

The control scheme presented in the paper has a number of interesting features compared to RLM. In particular, it has "built-in" TCP friendliness, it does not require coordination between receivers, and we have seen that results from an actual implementation show good convergence properties (in addition to the expected result that the combination of a layered coding and a layered transmission scheme does help to handle the problem of multicast delivery of heterogeneous networks). However, there remain a number of problems, most of them actually related to Equation 1. First, the equation only provides an upper bound on the rate of the equivalent TCP connection; there is a need to investigate how far the bound is to the actual value. Second, it is not yet clear exactly how a rate based control scheme based on equation 1 and an actual window based TCP scheme interact in practice. Third, the equation relies on parameters such as the mean roundtrip delay RTT that are problematic to estimate accurately in large multicast environments. This, combined with questions about the relevance of TCP-friendliness in networks with RED gateways, means that the jury is still out on receiver based control schemes for multicast environments, and that this remains an important (and somewhat urgent) area for research.

References

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

[2] Archive of the rem-conf mailing list at ftp://nic.es.net/pub/mailing-lists/mail-archive/rem-conf

[3] J-C. Bolot, T. Turletti, I. Wakeman, "Scalable feedback control for multicast video distribution in the Internet", *Proc. ACM/SIGCOMM'94*, Vol. 24, No 4, pp. 58-67, October 1994.

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

[5] N. Chaddah, "Software only scalable video delivery system for multimedia applications over heterogeneous networks", *Proc. ICIP 95*, Wash. DC, Oct. 1995.

[6] K. Claffy, H.-W. Braun, G. Polyzos, "Measurement considerations for assessing unidirectional latencies", *Journal of Internetworking*, vol. 4, no. 3, September 1993.

[7] S. Y. Cheung, M. H. Ammar, X. Li, "On the Use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution", *Proc. IEEE Infocom '96*, San Fransisco, CA, pp. 553-560, Apr. 1996.

[8] S. Casner, V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", *Internet-Draft*, Jul. 14, 1997.

[9] CNET reviews:
Internet phones: http://www.cnet.com/Content/Reviews/Compare/Webphone/
Videoconferencing tools: http://www.cnet.com/Content/Reviews/Compare/Video/
Collaborative tools: http://www.cnet.com/Content/Reviews/Compare/Netconference/

[10] J. Danskin, G. Davis, X. Song, "Fast lossy Internet image transmission", *Proc. ACM Multimedia '95*, Boston, MA, pp. 321-332, Nov. 1995.

[11] B.J. Dempsey, J. Liebeherr, and A.C. Weaver, "On Retransmission-Based Error Control for Continuous Media Traffic in Packet-Switching Networks", *Computer Networks and ISDN Systems*, Vol. 28, No. 5, pp. 719 - 736, March 1996.

[12] S. Floyd, V. Jacobson, "The synchronization of periodic routing messages", *Proc. ACM Sigcomm '93*, San-Francisco, CA, pp. 33-44, Sep. 1993.

[13] S. Floyd, K. Fall, "Router Mechanisms to Support End-to-End Congestion Control" *Technical Report*, Feb. 15, 1997.

[14] A.V. Garcia, "Control Mechanisms for Audio Transmission over the Internet" *PhD Thesis*, October 96.

[15] Free Phone, http://zenon.inria.fr/rodeo/fphone/

[16] 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.

[17] Internet Telephony Consortium (ITC), http://itel.mit.edu/

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

[19] T.V. Lakshman, U. Madhow, B. Suter, "Window-based error recovery and flow control with a slow acknowledgment channel: a study of TCP/IP performance", *IEEE Infocom'97*, Kobe, Japan, 1997.

[20] J. Mahdavi, S. Floyd, "TCP-Friendly Unicast Rate-Based Flow Control" *Draft posted on end2end mailing list*, Jan. 8, 1997.
http://www.psc.edu/networking/papers/tcp_friendly.html

[21] M. Mathis, J. Semke, J. Mahdavi, T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm" To appear in *Computer Communication Review*, Vol. 27, No 3, July 1997.

[22] S. McCanne, V. Jacobson, M. Vetterli, "Receiver-driven Layered Multicast", Proc. ACM Sigcomm '96, Stanford, CA, pp. 117-130, Sept. 1996.

[23] S. McCanne, M. Vetterli, V. Jacobson, "Low-complexity Video Coding for Receiver-driven Layered Multicast", to appear in IEEE JSAC '97.

[24] J. Mogul, S. Deering, "Path MTU Discovery", RFC 1063, Nov. 1990.

[25] S. Ott, J. Kemperman, M. Mathis, "Window Size behavior in TCP/IP with Constant Loss Probability", *DIMACS Workshop on Performance of Realtime Applications on the Internet*, Plainfield NJ, Nov. 6-8, 1996.

[26] V. Paxson, "Measurements and analysis of end-to-end Internet dynamics", *PhD Thesis*, University of California, Berkeley, Apr. 97.

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

[28] C. Perkins et al, "RTP Payload for Redondant Audio Data", *Internet-Draft*, Jan. 3, 1997, expires June 1997.

[29] R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks", *Proc. IEEE Infocom '94*, Toronto, Canada, pp. 680-688, April 1994.

[30] Robust-Audio Tool (RAT), http://www-mice.cs.ucl.ac.uk/mice/rat/

[31] J. Rosenberg, H. Schulzrinne, "Scaling Feedback to Very Large Groups" *Research report available via http://www.cs.columbia.edu/~hgs/papers/Rose9701:Scaling.ps.gz*, Feb. 13 1997.

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

[33] N. Shacham, P. McKenney, "Packet recovery in high-speed networks using coding and buffer management" *Proc. IEEE Infocom '90*, San Francisco, CA, pp. 124-131, June 1990.

[34] A. Spanias, "Speech coding: A tutorial review", proc. IEEE, vol. 82, no. 10, pp. 1541-1582, Oct. 1994.

[35] M.F. Speer, S. McCanne, "RTP usage with layered multimedia streams", Internet-Draft, June 1, 1996, expires Dec. 1996.

[36] LBNL Audio Conferencing Tool (VAT), http://www-nrg.ee.lbl.gov/vat/

[37] VocalTec http://www.vocaltec.com/

[38] Voice On the Net (VON), http://www.von.com/

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

A The receiver-based control pseudo code

```
...
int N;
int NbFlow = 0;
int k = 0;
ACTION = JOIN;
...

// Receive Data and/or Control Packet

void Packet_Received()
{
  if (Rtp_Packet()) {
     SeqNb = Get_Sequence_Number();
     Update_Loss();
     Compute_TCP_Rate();
     Cpt_RTT -= 1;
     if (! Cpt_RTT) {
        Cpt_RTT = N;
```

```
        Send_RR_Packet(SeqNb, CurrentTime());
     }
  } else
      if ((Rtcp_Packet()) && (Is_SR_Packet()) &&
          (Packet_Ssrc() == MySSRC)) {
        UpdateRTT();
        Compute_TCP_Rate();
      }
}

// Rate Control Algorithm

void Algo(...)
{
  switch(ACTION) {
    case JOIN:
        if (NbFlow)
          k = 0;
        L = Nb_Of_Layer(TCP_Rate(), Our_Rate());
        Join_Layers(L, &NbFlow);
        Wait(n*d);
        ACTION = NONE;
        break;
    case QUIT:
        L = Nb_Of_Layer(TCP_Rate(), Our_Rate());
        Quit_Layers(L, &NbFlow);
        if (NbFlow >= 1) {
            Wait(n*d);
            ACTION = NONE;
        } else
            ACTION = RETRY;
        break;
    case NONE:
        if (TCP_Rate() > Our_Rate()+NEXT_Layer_Rate())
          ACTION = JOIN;
        else
          if (TCP_RATE() < Our_Rate())
            ACTION = QUIT;
          else
            {
              ACTION = NONE;
              k=0;
            }
        break;
    case RETRY:
        tr = power(2, ++k);
```

```
        wait(tr);
        Join_Layers(1, &NbFlow);
        Wait(n*d);
        ACTION = NONE;
        break;
    }
...
}
```

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399