Experiments with a Layered Transmission Scheme over the Internet

Thierry Turletti, Sacha Fosse Parisis, Jean-Chrysostome Bolot

INRIA

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

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

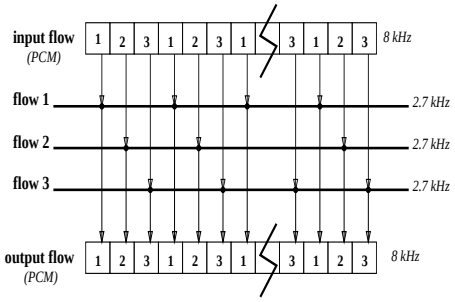

Figure 1: Hierarchical coding scheme

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.

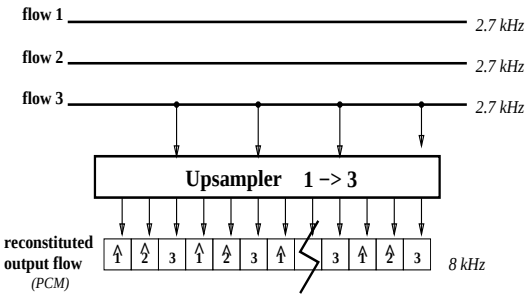

Figure 2: Reconstruction 1 to 3

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]

$$\lambda_{equ} = 1.22 * \frac{MTU}{RTT * \sqrt{Loss}} \quad (1)$$

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