Layered Video Multicast with Retransmissions (LVMR): Evaluation of Hierarchical Rate Control [1]

Xue Li, Sanjoy Paul†, and Mostafa Ammar**

∗: College of Computing, Georgia Tech, Atlanta,GA 30332, {lixue, ammar}@cc.gatech.edu
†: Bell Laboratories, Holmdel, NJ 07733, sanjoy@dnrc.bell-labs.com

Abstract

Layered Video Multicast with Retransmissions (LVMR) is a system for distributing video using layered coding over the Internet. The two key contributions of the system are: (1) improving the quality of reception within *each* layer by retransmitting lost packets given an upper bound on recovery time and applying an adaptive playback point scheme to help achieve more successful retransmission, and (2) adapting to network congestion and heterogeneity using hierarchical rate control mechanism. This paper concentrates on the rate control aspects of LVMR. In contrast to the existing sender-based and receiver-based rate control in which the entire information about network congestion is either available at the sender (in sender-based approach) or replicated at the receivers (in receiver-based approach), the hierarchical rate control mechanism *distributes* the information between the sender, receivers, and some agents in the network in such a way that each entity maintains only the information relevant to itself. In addition to that, the hierarchical approach enables intelligent decisions to be made in terms of conducting concurrent experiments and choosing one of several possible experiments at any instant of time based on minimal state information at the agents in the network. Protocol details are presented in the paper together with experimental and simulation results to back our claims.

[1]This work is supported by grants from IBM,Intel and the National Science Foundation under grant NCR-96283793

1 Introduction

In recent years, there has been a fast expansion of the Internet and intranets, and significant growth in both computer processing power and network bandwidth. Such infrastructure improvement introduced opportunities for new multimedia applications over networks, such as video conferencing, distance learning, remote presentation, and media on demand. These application usually involve realtime video distribution.

Approaches proposed to handle the realtime aspect of video distribution over networks fall into two categories: (1) the use of a network capable of resource reservation to provide *performance guarantees* [BFGH+95] [FBZ94] [ZDE+93] [Sha92], and (2) the use of *adaptive control* to adjust multimedia traffic characteristics to meet the current network's capacities [BT94] [BTW94] [KMR93] [CAL96] [LPPA97] [MJV96]. This latter approach, which has been the focus of our work, is more compatible with the near-term architecture and capabilities of the Internet.

Both RSVP [ZDE+93] and ATM , which offer network-level reservations, are far from being deployed on a wide area network and even further away from being used ubiquitously for real-time video distribution. Even when reservations are available, there are two reasons to apply adaptation techniques: (1) it is hard to make very accurate reservations so some adaptation is required to allow tolerance in reservation accuracy; (2) considering the cost of resource reservation, it is possible to reserve only enough resources to provide the basic video quality and transmit other enhancement layers with best-effort network support and adaptive control. Thus the main motivation of this research is to explore the feasibility of video distribution over wide area networks with the best effort delivery support and end-to-end control mechanisms.

Due to the heterogeneity of the Internet, multicasting the same video stream to receivers distributed in a wide range may bring a potential problem: the video stream might congest certain low-capacity or heavily loaded areas of the network, while some other high-capacity or lightly loaded areas are still under-utilized. However, in a *fair* scheme [CAL96], each receiver should receive a video stream with a quality commensurate with its processing power and the bandwidth capacity on the path leading to it. One way of ensuring fair distribution is to multicast different layers of video (which contain progressive enhancements) using different multicast addresses and let the receivers decide which multicast group(s) to subscribe to [MJV96]. An alternative approach is to multicast *replicated* streams and use destination set grouping (DSG) protocol [CAL96][LA96] for improving fairness of video distribution in a heterogeneous network with a small overhead in terms of bandwidth cost. Layered Video Multicast with Retransmissions (LVMR) [LPPA97] addresses the network congestion and heterogeneity problem using layered video coding techniques by allowing each receiver to subscribe to a subset of the video layers according to its processing power and network bandwidth availability. LVMR also deploys an error recovery scheme using smart retransmission and adaptive playback point. That part of the work, together with an overview of LVMR, is presented in [LPPA97], while this paper focuses on the rate control schemes for layered video multicast.

There have been two principal approaches to address the rate control problem in video multicast: sender-initiated control [BTW94] and receiver-initiated control [MJV96]. In the sender-initiated approach, the sender multicasts a single video stream whose quality is adjusted based on feedback information from receivers. The receiver-initiated approach is usually based on a layered video coding scheme, in which the sender multicasts several layers of video (typically a base layer and several enhancement layers) in different multicast groups, and a receiver subscribes to one or more layers based on its capabilities. This scheme is "receiver-initiated " in the sense that a receiver decides on its own whether to drop an enhancement layer or to add one. There are also hybrid approaches like [CAL96] in which the intra-stream protocol is sender-initiated, while the inter-stream protocol is receiver-initiated.

Comparing LVMR with RLM [MJV96], both systems deploy layered video multicast schemes, with however, some substantial differences in terms of the mechanisms used for adding or dropping a layer. In RLM [MJV96], a fully distributed approach is advocated in which a receiver, by itself, makes decisions to add or drop an enhancement layer. This decision is enhanced by a "shared learning" process in which information from experiments conducted by other receivers is used to improve performance. The idea of shared learning, although an improvement to adding and dropping layers indiscriminately, requires each receiver to maintain a variety of state information which it may or may not require. In addition, the use of multicasting to exchange control information may decrease usable bandwidth on low speed links and lead to lower quality for receivers on these links. In LVMR, we take a hierarchical approach in the receivers' dynamic rate control schemes, so as to allow receivers to maintain minimal state information and decrease control traffic on the multicast session. It also provides more functionality compared to the simple receiver-driven schemes, as in RLM [MJV96]. In particular, it allows multiple experiments to be conducted simultaneously, and also helps drop the correct layer(s) during congestion in most cases, which are not possible in RLM because of its completely distributed approach.

The network mechanism used for video distribution is IP-multicast, which, by virtue of using the Internet Group Management Protocol (IGMP) [D89], and a multicast routing protocol, such as, Distance Vector Multicast Routing Protocol (DVMRP) [D88] [DC90], Protocol Independent Multicast (PIM) [DEF+94] or Core-Based Tree (CBT) [BFC93], sets up a multicast tree spanning all receivers. The multicast tree is defined by the IP-address of the sender and a Class-D IP-address of the group. Receivers can dynamically join and leave a multicast group, thereby leading to the dynamic reconfiguration of the multicast tree. The sender of the multicast group need not know the receivers, and therefore can keep sending IP packets using the Class-D IP-address representing the group. On the other hand, it is the responsibility of the receivers to inform their nearest router(s) that they want to be part of a specific group.

In this paper, we focus on the hierarchical rate control scheme for layered video multicast. We will introduce layered video compression techniques and LVMR system overview in Section 2, and discuss the main ideas of hierarchical layer rate control in Section 3. In Sections 4 and 5, we present the hierarchical rate control protocol and some extension features, followed by a discussion of a prototype implementation and some experimental results in Section 6. After that, we present and analyze simulation results in Section 7. Finally we conclude the paper in Section 8.

2 Background and System Overview

2.1 Layered Video Compression

The use of layered encoding schemes enables video multicast schemes to deliver optimal quality to receivers with heterogeneous capabilities. Layered encoding schemes separate an encoded video stream into two or more layers. There is one *base layer* and one or more *enhancement layers*. The base layer can be independently decoded and it provides a "basic" level of video quality. The enhancement layers can only be decoded together with the base layer and they provide improvements to video quality.

Layered multicasts provide a finer granularity of control compared to using a single video stream, because a receiver may subscribe to one, two, or more layers depending on its capabilities If a receiver experiences packet loss as a result of network congestion, dropping one or more layers will reduce congestion, and hence will reduce potential packet loss.

The MPEG-2 International Standard supports layered encoding by defining scalable modes. The four scalable modes: spatial scalability, data partitioning, SNR scalability and temporal scalability, each provides the mechanism to perform layering. In general, from a combination of these scalability modes it is possible to design a layered coding system with a large number of layers.

In our prototype system, we propose the use of a simpler layering mechanism for MPEG video. Although our approach is not strictly compliant with standards, we have found it to work on several software and hardware MPEG decoders. The technique provides a simple way to achieve layering in a manner similar to temporal scalability. In MPEG video coding, frames are coded in one of three modes: intraframe (I), predictive (P) or bidirectionally-predictive (B). These modes provide intrinsic layering in that an I frame can be independently decoded, while P frames require I frames, and B frames generally require I and P frames to decode. By using a multicast group for each frame type a simple layering mechanism is obtained.

Our choice of layering technique is strongly influenced by the need for easy integration of any scheme with current MPEG-based systems. The layering is therefore implemented as a postprocessing filter for a standard MPEG bit stream. After an MPEG stream is encoded, a filter parses the output MPEG bit stream to identify markers that demarcate the start of a frame. Next, the frame type field is decoded and, based on the frame type, bits of the MPEG stream are directed to the appropriate multicast group until the next marker identifying start of a new frame. At the decoder, a multiplexer is used to sequence video data from the different multicast groups so that an MPEG decoder is able to decode the resulting multiplexed stream.

2.2 System Overview

LVMR system architecture addresses application, application-control, and transport layers. The application layer consists of the video server which is responsible for digitizing and coding video frames at the sending end, and the video client which is responsible for decoding and displaying video frames at the receiving end. Application control consists of a demultiplexer at the sending end to demultiplex a video stream into several substreams, and a multiplexer at the receiving end to multiplex back one or more substreams into a single stream for the video client. In particular, in our system, the demultiplexer generates three substreams: I, P, and B. At the transport layer, each of these substreams is transported using a separate flow, where each flow uses a separate IP-multicast group address. The multiplexer at the application control of the receiving end multiplexes one, or more of these substreams depending on the network load and the resources of the end hosts, and presents the multiplexed stream to the decoder. The application-control layer also includes *Playback Synchronizer* that adapts playback point for error recovery [LPPA97] and *Hierarchical Rate Controller*, which is the focus of this paper.

The basic idea from the networks point of view is illustrated in Figure 1(a). Three different multicast trees are set up at the network layer, one for each substream.[2] Since the network load changes during a session, a receiver may decide to join/leave a given multicast group, thereby

[2]Note that it is not necessary for all the multicast trees to be identical.

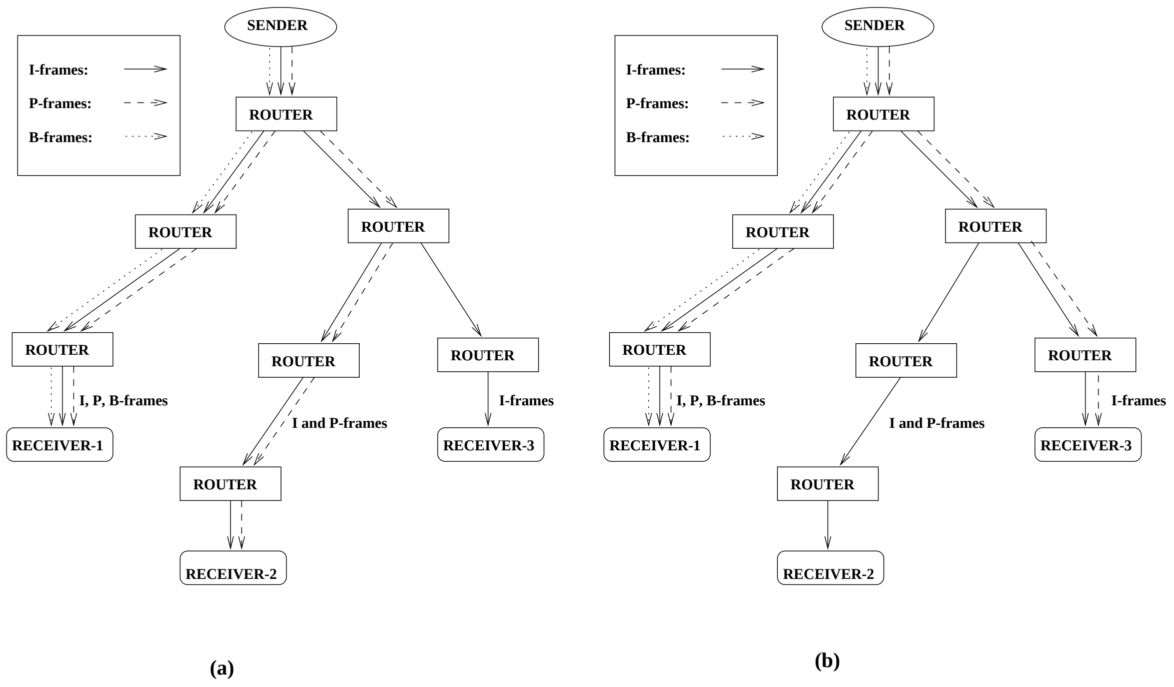


Figure 1: Network-layer View of the Multicast Trees

extending/shrinking the multicast trees (Figure 1(b)). In Figure 1(b), the multicast tree modifications are shown when Receiver-2 decides to quit the multicast group associated with P substream, and Receiver-3 decides to join the multicast group associated with P substream.

3 Discussion of Hierarchical Rate Control

3.1 Approaches: distributed control vs centralized control?

In a purely distributed control scheme each receiver must decide which layer to listen to by itself. The video source does not play any role in the control of the receiver and merely sends layered video streams on multiple multicast groups. Each receiver determines the layers of video it wishes to receive and subscribes to these layers. This technique has the desirable property that actions taken by a receiver are completely decoupled from those of other receivers and senders. However, in a multicast scenario, a pure distributed control scheme will suffer from poor performance since receivers do not coordinate their decisions.

At the other end of the spectrum, a pure centralized control scheme is one where the decision making is performed by an entity which has complete information on receiver capabilities and network topology and congestion conditions. Based on this information, the layers that are to be received can then be computed by this entity and sent to the receiver. It is clear that, if accurate network topology and loading information could be gathered, this technique would lead to the selection of the optimal number of layers for each receiver. However, in the context of large heterogeneous multicast sessions, this type of scheme will scale poorly since it requires receivers to provide constant updates to the centralized controller to track network conditions reliably for the multicast session.

The disadvantages of these approaches has led us to suggest a third approach which attempts to combine the positive aspects of both. Our approach is hierarchical in the sense that the receivers do not give feedback directly to the sender (as in the centralized control) rather they send their feedback to some *agents* in the network. The agents can be either regular receivers or specialized receivers depending on context. The agents are arranged in a hierarchy as described in the next section and are responsible for collecting feedback from receivers and distributing consolidated information back to the receivers. This allows for the information gathering capabilities of the centralized system while allowing distributed decision making capabilities of receivers based on the consolidated information passed on to them by the agents.

3.2 The Hierarchical Approach

Our architecture proposes the use of multiple domains within a multicast group with an Intermediate Agent (IA) in every domain. A domain can be a physical region like a geographical area, or a logical region like a corporate Intranet, or even a logical scope imposed by time-to-live (TTL) field of IP packets. In each subnet within a domain, there is a Subnet Agent (SA). SAs are responsible for collecting the status of their subnets while IAs perform a similar function for the domain. The IA compiles the information from all SAs in its domain and passes it down to the SAs. SAs multicast the information to their corresponding subnets. Thus the *intelligence* of the IA propagates down to the receivers.

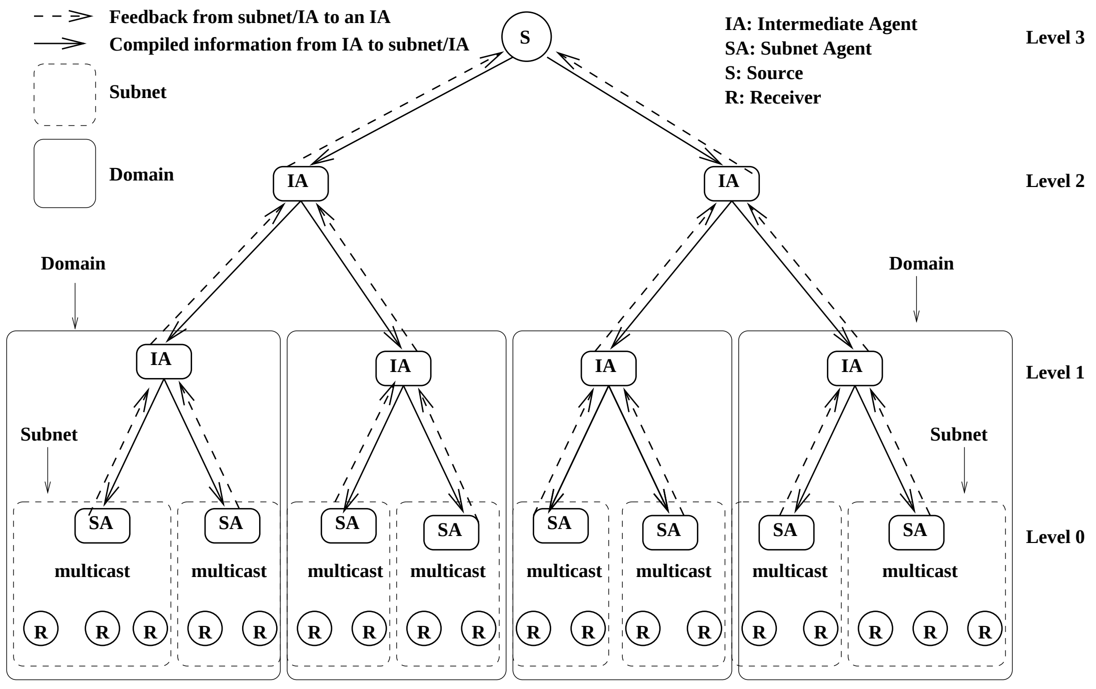


Figure 2: Hierarchical Control Architecture of LVMR

This notion can be easily extended to multiple levels of hierarchy (refer to Figure 2), such that there are IAs at different levels, and an IA at a lower level[3] passes its domain information to its next higher level IA, and so on until it reaches the source. Similarly, there is a flow of compiled information from a higher-level IA to its next lower level IAs (just as the compiled information flows from IA to the SAs in a two-level hierarchy), and so on until it reaches all SAs. It is to be noted here that SAs and IAs are only logically separate from the receivers, but they can be physically co-located (running on the same machine) as the receivers, We are not proposing to have dedicated machines to do the job of an IA and/or SA.

3.3 Shared Learning in Layer-Add Experiments

The key to scalability in layered multicast is for the receivers to make a decision on their own regarding adding or dropping a layer. However, if these decisions are made independent of the results of join/leave experiments done by others, the results can be disastrous. Thus it is fundamental for each receiver to know about the experiments and their results. This is called *shared learning* [MJV96], because receivers share knowledge of what is going on in the network. The mechanism used in the RLM protocol [MJV96] to achieve this goal is *multicasting* the beginning of experiments and the results of experiments to *every* member of the group. We believe this is neither scalable nor efficient, because of the following reasons:

1. Every receiver need not know about every experiment and/or its result. That is just too much state information.

2. Using multicast to distribute control information, such as, beginning of experiments and their results, beyond certain scope is inefficient, because it consumes additional bandwidth, particularly if every receiver need not know about every experiment and/or its results.

Our solution precisely tries to avoid the above drawbacks of RLM using *intelligent partitioning* of the knowledge base and distributing relevant information to the members in an efficient way. Note that there are several receivers in a multicast group, potentially distributed over a large heterogeneous network, running several experiments, some with success and some with failure. If all these experiments along with their results are compiled into a knowledge base, that would represent the *comprehensive group knowledge base* . In order to partition the comprehensive group knowledge base in an intelligent way, it is necessary to understand:

1. Which receivers get affected when an add-layer experiment is performed?

2. What can be learned from a failed add-layer experiment?

3. Which receivers should learn from a failed add-layer experiment?

The following discussion addresses the above questions.

[3]Assume that SA is the lowest level IA and the source is the highest level IA.

Range	Itself	Others in Subnet	Others in Domain	Others outside Domain
A: $(1 \leq L \leq SL_{max})$	Yes	No	No	No
B: $(SL_{max} < L \leq DL_{max})$	Yes	Yes	Some yes	No
C: $(L > DL_{max})$	Yes	Yes	Yes	Some yes

Table 1: Video levels and who may get affected

3.3.1 Range of video levels and the affected region

Some definitions are in order before going through the discussion in this section. A receiver is said to be in video *level n* (L_n) if it subscribes to n video layers (the base layer and n - 1 enhancement layers). For example, in our prototype system, which uses only three layers for MPEG – I, P, and B, a receiver is in level 2 if it receives I, and P layers and it is in level 3 if it receives I, P, and B layers.

SL_{max} and DL_{max} denote the highest level among the receivers in a subnet or a domain respectively. Also, $SL_{max} \leq DL_{max}$.

Table 1 illustrates three ranges of video levels[4].

Range A: 1$\leq$ L $\leq SL_{max}$. If a receiver joins a layer in this range, it is not going to affect any other receiver, i.e. if it gets "congested", this can only be caused by its own CPU overload, not network congestion. Here the *affected region* is only itself.

Range B: $SL_{max} <$ L $\leq DL_{max}$. If a receiver joins a level in this range, it is definitely going to affect everyone else in the same subnet, and it may possibly affect some nodes outside its subnet, but it will not affect any nodes outside the domain. The potential *affected region* is now the whole domain.

Range C: $DL_{max} <$ L If a receiver joins a level in this range, it is definitely going to affect everyone else in the domain, and it may also affect receivers outside the domain.
This means, each receiver needs to know SL_{max}, SA needs to know SL_{max} and DL_{max}, while IA needs to know DL_{max}.

3.3.2 Failed add-layer experiments

Some new notations will help simplify the discussion in this section.

$CONG(A, l, X)$: *Host A adds layer l causing the network path leading to host X to be congested.*

$L(X)$: *Host X's video reception level.*

There are three key observations which are fundamental to deciding which receivers need to learn from which failed add-layer experiments.

Observation 1 $\exists X, CONG(A, l, X) \Rightarrow CONG(A, l, A)$
Assumption: *in the routers, there is a separate buffer for each of the outgoing link.*

If receiver A adds a layer l and in the process, causes congestion to some other receiver/subnet, then it itself must also be congested. Another way of interpreting the same thing is, if a receiver

[4]To keep the discussion simple, we assume there is a single domain with multiple subnets in it. However, the ideas are equally applicable in a multi-level hierarchy.

is *not* congested after its own add-layer experiment, *no other* receiver/subnet should be congested due to the experiment.

This implies that congestion of a receiver in the network *should not* be correlated with an add-layer experiment if the receiver that performed the experiment did not experience congestion itself.

Observation 2 $CONG(A, l, X) \Rightarrow L(X) \leq L(A)$

If receiver A adds layer l, it cannot bring congestion to nodes that are already receiving l. That is, as a result of a receiver's add-layer experiment, congestion may only be caused at nodes that are at the same level as or at a lower level than the receiver.

This implies that each receiver should only know about those add-layer experiments that involve adding layer(s) higher than its current level.

Observation 3 $CONG(A, l, B) \Rightarrow CONG(B, l, A)$

If as a result of receiver A's adding layer l, some other receiver B gets congested, then receiver A would also be congested if B adds layer l under the same condition.

This implies that if there are receivers which get *mutually* affected due to an add-layer experiment done by any one of them, they should share the information about the same experiments and results.

3.4 Collaborative Layer Drop

If a receiver senses congestion, it may drop the highest layer to reduce the congestion. But this is not always true in multicast. For example, two receivers (receiving 2 layers and 3 layers) on the same subnet both experience congestion caused by a traffic overload on the shared link. Then even if the receiver that is receiving 2 layers drops layer 2, congestion will not be decreased until after layer 3 is dropped by the other receiver.

We propose *Collaborative Layer Drop* in our hierarchical rate control protocol to achieve more efficient and intelligent layer adaptation during congestion in a multicast environment.

1. If a receiver A is congested and finds another receiver B on the same subnet also congested, and $L(A) < L(B)$. Then this A should not drop layer $L(A)$ until after B has dropped layer(s) $L(A) + 1$ to $L(B)$.
2. If the IA finds that a majority of the subnets (including the one(s) receiving layer DL_{max}) are congested, then it can send a mandatory message to get layer DL_{max} dropped so as to decrease or avoid this congestion that is affecting the whole domain.

More details about how to realize the above ideas are presented in Section 4.

3.5 Add-layer Experiment Synchronization

Conducting add layer experiment at deferent layers simultaneously can possibly affect each other. In order to *avoid* such concurrent experiments, a simple scheme can be applied. Let g denote the ID of the current video GOP the receiver has received and n total number of layers. A receiver receiving i layers can only add layer $i+1$ when $mod(g, n-1) = i-1$, where mod is the function of remainder. For example, if there are totally 4 layers, then layer 2 can only be added if $mod(g, 3) = 0$, layered 3 can only be added if $mod(g, 3) = 1$ and layer 3 if $mod(g, 3) = 2$. This way, experiments of adding different layers are conducted at least a GOP number of frames away. Section 5 provides a more detailed study of concurrent add layer experiments for different layers and proposes an intelligent scheme to handle it.

While heterogeneous layer add experiments should be conducted at different time, experiments adding the same layer should be synchronized to happen at the same time. In Section 4 schemes to pass and share the possible waiting time between experiments help to achieve such Synchronization.

4 The Hierarchical Rate Control Protocols

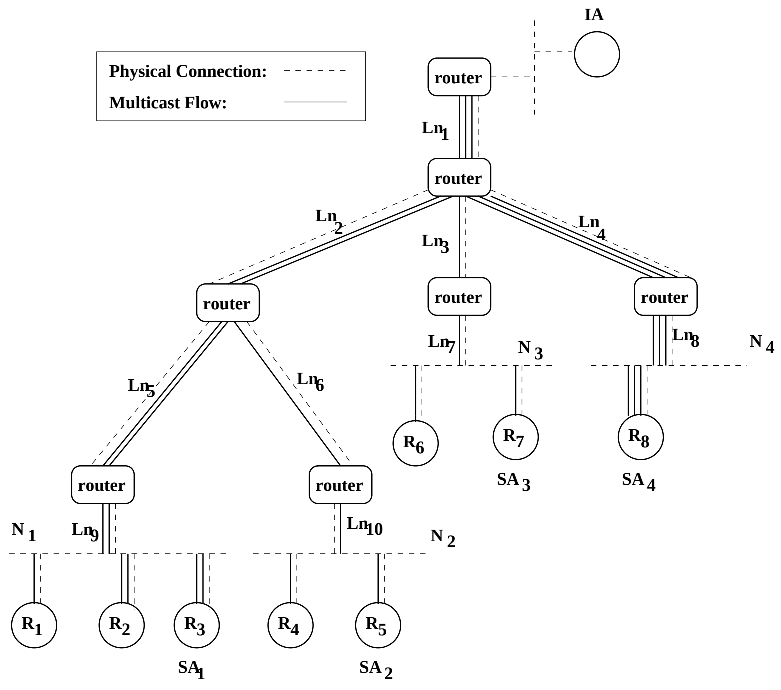


Figure 3: Example Topology and Experiments

In this section, the state transition will be described first, followed by the description and an example of the basic hierarchical rate control protocol. In the next section, the extensions to the basic protocol will be discussed.

Messages exchanged between various entities in the network, particularly between the Subnet Agent (SA) and the Intermediate Agent (IA), are made reliable by using TCP. However, the communication between a receiver and the corresponding SA is made reliable by using a simple timer-based mechanism, in which the receiver retransmits the same message if the expected action does not happen within the time-out period.

For the sake of simplicity, the protocol will be stated using only two levels of hierarchy, meaning there is a single IA, and several SAs, where the SAs exchange information with the IA. However, the basic scheme can be easily extended to multiple levels of hierarchy.

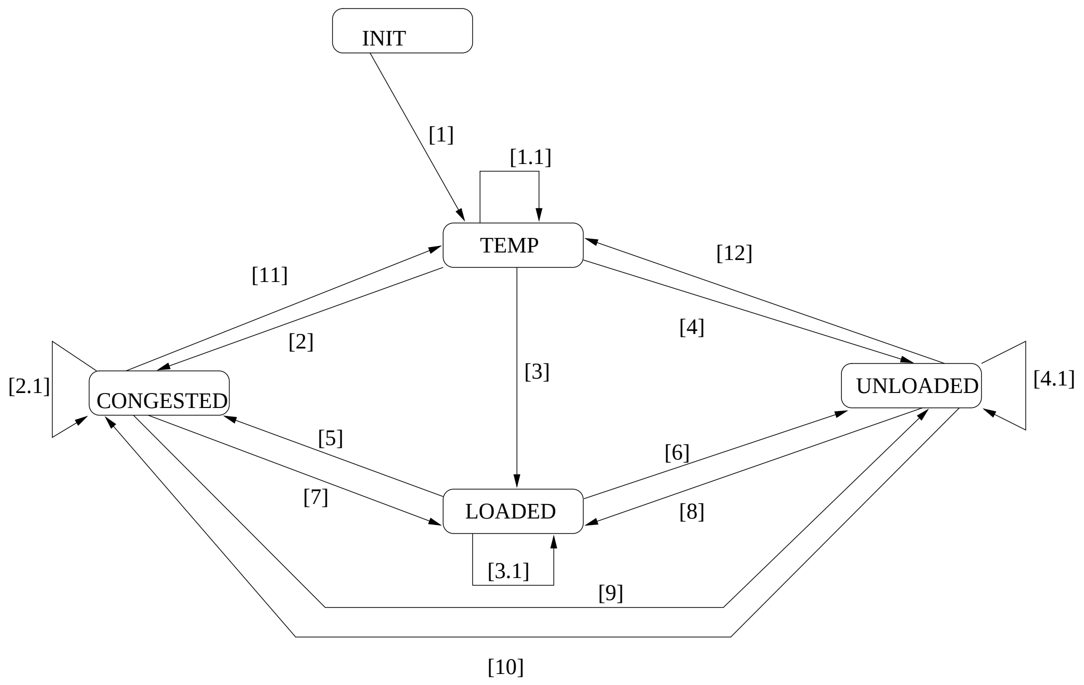


Figure 4: State Transition Diagram

An example topology given in Figure 3 will be used for the illustration. In this figure, there are four subnets N_1, N_2, N_3, and N_4, each with at least one receiver. The receivers are $R_1$to R_8. $Ln_1$to Ln_{10} are links connecting routers to routers or to the corresponding subnets. We assume that $SL^{(1)}_{max} = 2$, $SL^{(2)}_{max} = 1$, $SL^{(3)}_{max} = 1$, and $SL^{(4)}_{max} = 3$. Note that the number of multicast flows in each subnet is indicated by the number of solid lines.

4.1 The State Transition

A simplified discussion of the state transition diagram for the basic protocol can be found in [LPPA97]. Each transition in Figure 4 is associated with (1) a *condition* clause, (2) a *send* operation, (3) a *receive* operation, and (4) an *action* clause. The "condition" clause indicates that the "condition" must be satisfied in order for the transition to take place, the "send" operation indicates what message(s) are sent during the transition, the "receive" operation indicates what messages are received during the transition, and "action" clause indicates what action is taken by the protocol entity during the transition. In Table 3, we have used three "conditions:" (1) *Congestion-condition*, (2) *Load-condition* and (3) *Unload-condition* to assist state transitions. As stated in [LPPA97], congestion can be detected by various factors, such as, packet loss rate exceeding a certain threshold, or percentage of video frames missing deadlines exceeding a threshold. Similarly, if packet loss rate is below a certain threshold, and the percentage of video frames arriving late is below a given threshold, the network is in *Unload-condition.* Any network condition that cannot be categorized either by "Congestion" or "Unload" is denoted by "Load". Periodically, each receiver does some simple statistics on the packet loss ratio and/or frame arrival timing in the past period and matches the result to one of the three conditions. Then state transition takes

Parameter	Description
δ	Time counter of how long a receiver has been continuously in certain state
T_t	Time a receiver needs to spend in the TEMP state before transition to other states.
T_c^i	Time a receiver needs to remain in the CONGESTED state before dropping layer i.
T_u^i	Time a receiver needs to remain in the UNLOADED state before adding layer$i+1$.
r	Packet loss rate during a period (usually a GOP)

Constant	Description
$T_{c,0}$	The default value for T_c^i
$T_{c,add}$	The value for T_c^i right after a layer-add experiment
$T_{u,min}, T_{u,max}$	The range for T_u^i. $T_{u,min}$ is also the default value
R_c, R_u	*Unload-condition:* $r \le R_u$. *Congestion-condition:* $r \ge R_c$
α	Factor to increase T_u^i if adding layer $i+1$ fails.

Table 2: Parameters and Constants in the State Transition and Protocol

Transition	Description of Transition
[1]	*Send:* LAYER_REQ *to* subnet *Receive:* LAYER_ACK(L = SL_{max}) *from* SA *Action:* Join layers 1 to SL_{max}
[1.1]	*Condition:* $\delta < T_t$AND NOT Congestion-condition
[2]	*Condition:* Congestion-condition
[4.1]	*Condition:* $\delta < T_c^i$ AND Congestion-condition
[3]	*Condition:* $\delta \ge T_t$ AND Load-condition
[3.1]	*Condition:* Load-condition
[4]	*Condition:* $\delta \ge T_t$ AND Unload-condition
[4.1]	*Condition:* $\delta < T_u^i$ AND Unload-condition
[5]	*Condition:* Congestion-condition
[6]	*Condition:* Unload-condition
[7]	*Condition:* Load-condition
[8]	*Condition:* Load-condition
[9]	*Condition:* Unload-condition
[10]	*Condition:* Congestion-condition
[11]	*Condition:* Congestion-condition AND $\delta \ge T_c^i$ *Action:* drop layer i
[12]	*Condition:* Unload-condition AND $\delta \ge T_u^i$ *Action:* add layer i+1

Table 3: Receiver's State Transition Table

place after the statistics. All the timing parameters and constants are in the unit of such "period", and in our prototype system and simulation, this period is set to the time span of a GOP (Group of Picture) of MPEG frames.

The parameters involved in the state transitions are described in Table 2. T_u^i and T_c^i are two very important parameters and their adaptation reflects the key ideas in hierarchical rate control protocol.

A receiver needs to remain in the $UNLOADED$ state for T_u^i time before adding layer $i+1$. Initialy, for all the layers, T_u^i is set to $T_{u,min}$ and should always stay within $T_{u,max}$. When a layer-adding experiment fails, T_u is exponentially expanded ($T_u^i = T_u^i * \alpha$) so as to avoid oscillation caused by another add-layer experiment that happens too soon. With hierarchical rate control schemes, other receivers that could also possibly cause congestion when adding layer $i+1$, can "learn" from this failure by setting their T_u^i to the same prolonged value. This learning procedure is realized by some message exchanging mechanisms described in the next subsection. When an add-layer experiment succeeds, the T_u is reduced ($T_u^i = T_u^i/\alpha$) and for next layer, it is set as $T_u^{i+1} = T_u^i$.

A receiver needs to remain in the $CONGESTED$ state for time T_c^i before dropping layer i. T_c^i is set to the default value of $T_{c,0}$ except in two cases: (1) Right after the experiment of adding layer i, T_c^i is set to a smaller value, $T_{c,add}$, such that if the newly added layer i brings too high a load to the network, the receiver will detect it quickly and drop layer i. (2) If a receiver (receiving i layers) is in the $CONGESTED$ state and finds another receiver (receiving more than i layers) on the same subnet also congested, then it extends $T_c^i = T_c^i + 1$. The intuition is that if a receiver finds that on the same subnet, another receiver receiving higher layer(s) is also congested, then possibly the congestion is caused by the higher layer(s). So this receiver should wait longer before dropping its own layer.

4.2 The Protocol

There are six fundamental operations performed by a receiver in hierarchical rate control, namely, *join-session*, *leave-session*, *add-layer*, *drop-layer*, *send-cong-msg* and *proc-cong-msg*. Each of these six operations are described below. Refer to Figure 4 and Table 3 as the operations are being described. Note that **leave-session** messages and the operations by IA/SA have not been shown in the state diagram.

1. **join-session**: A new receiver goes through the *join* procedure when it is in the *INIT* state.

   (a) A new receiver R announces to its subnet that it wants to join the multicast session. The announcement is made by multicasting (with a TTL of 1) a LAYER_REQ message to its subnet (label [1] in Figure 4).

   (b) If there is already an SA, the SA multicasts a LAYER_ACK (SL_{max}, T_u^i) message to its subnet after receiving LAYER_REQ.

   (c) When R receives LAYER_ACK(i, T) it subscribes to i layers and set $T_u^i = T$. If R does not hear a LAYER_ACK message within a certain time, it assumes that there is no current SA, so itself becomes the SA and joins the base layer.

It is possible that two or more receivers may become SAs on the same subnet, and the control functionalities of this protocol still work in this situation, although some extra information distribution and processing may be involved. A simple scheme can be applied to eliminate the extra SA(s) during the later control message exchanges.

2. **leave-session:**

   (a) If a receiver that is not an SA decides to leave the session, it multicasts a message to inform the SA about its leave and then quits.

   (b) Otherwise, it needs to choose another receiver to be the SA (if there are still other receivers on the same subnet). It multicasts a SA_LEAVE message to the group and whoever responds the first will be chosen as the next SA. If it does not hear any response within certain time, it will resend the message and if again no response, then it assumes that there is no other receiver on the subnet.

3. **send-cong-msg:**

   A receiver R announces to its subnet when it is in the *CONGESTED* state. This announcement is done by multicasting (with a TTL of 1) a CONG (i) message, where i is the receiver's video reception level.

4. **proc-cong-msg:**

   (a) If within a certain time interval, more than one CONG(i) message, where $i = SL_{max}$, is received by the SA, it informs IA with CONG(i) message showing that its subnet is congested. Note that this kind of congestion may result from some add-layer experiments (described later) going on in another subnet or network load increase.

   (b) When the IA gets a congestion message, it keeps the message for some time. Please refer to the descriptions for *add-layer* and *drop-layer* for further operations.

5. **add-layer:** If a receiver is receiving $i - 1$ layers and has been in the *UNLOADED* state for certain time ($\delta \geq T_u^{i-1}$), it attempts to do an add experiment for layer i.

   (a) If no congestion is observed at this node during certain time T_t after the experiment, or if $i \leq SL_{max}$, the $add - layer$ operation is done.

   (b) Otherwise the receiver drops layer i, extends $T_u^{i-1} = T_u^{i-1} * \alpha$, and multicasts a FAIL($i, T_u^{i-1}$) message on the subnet.Then SA informs IA about the failure of the experiment by the same message.

   (c) If IA receives a CONG(x) message from an SA and a FAIL(i, T) within a specified period of time, and $x < i$, it forwards this FAIL message to the SA, and then SA multicasts it to the subnet. The receivers in the subnet updates their T_u^{i-1} accordingly. If the receiver is currently receiving $i - 1$ layers and in the *UNLOADED* state, then $\delta = 0$. (The state time counter is reset to 0 so that from now on this receiver will also have to wait for at least T_u^{i-1} before trying to add layer i. This also helps to achieve add layer experiment synchronization.)

6. **drop-layer:**

If a receiver senses congestion, it may try to reduce its rate by dropping one or more enhancement layers. But this is not always the right approach. For example, as in figure 3, if congestion happens on Ln_5, then although R_1 senses the congestion, R_1 should not drop any layer, and R_2 and R_3 should both drop their highest layer.

(a) If a receiver (at level i) is in the $CONGESTED$ state and hears a CONG(x), and $x > l$, then it extends $T_c^i = T_c^i + 1$.

(b) If a majority of SAs report congestion within a short time, the IA requests the highest video layer in the domain to be dropped to decrease the congestion. A DROP(DL_{max}) message is sent to those SAs that reported CONG (i) within certain time and $i = DL_{max}$. The SAs would multicast the DROP message to the subnet.

(c) After a receiver (receiving i layers) has been in the $CONGESTED$ state for certain time ($\delta \geq T_c^i$), or after it hears a DROP(i) message, it drops the highest layer i.

4.3 Examples

Suppose, in Figure 3, R_1 is at L_1 (level 1), and it is ready to add layer 2. We refer to this experiment as $L_1 \rightarrow L_2$ experiment. Since $SL_{max}^{(1)} = 2$, R_1 can safely do the experiment without affecting anybody. However, if R_2, which is at L_2, wants to add layer 3 which is beyond $SL_{max}^{(1)}$, it multicasts ADD(3) message on the subnet (N_1). and SA_1 will inform IA about the experiment. Subnet agent of the affected subnet (SA_2 in this example) sends a CONG(1) message to the IA indicating congestion in subnet N_2. Also SA_1 sends a FAIL(3) message to the IA indicating congestion in subnet N_1, and hence failure of the experiment in its subnet. IA correlates congestion in subnet N_2 with the $L_2 \rightarrow L_3$ experiment done by R_2, and passes the information down to SA_2 with FAIL_($3, T_u$) message. Note that this will prevent a $L_2 \rightarrow L_3$ experiment in N_2 for at least a time period T_u.

5 Protocol Extensions: Concurrent Experiments and Virtual Subnet

5.1 Potential problems with concurrent experiments

There are some potential problems when more than one experiment is being conducted simultaneously in the same domain. If multiple experiments are all being run at the same level, or at a level below SL_{max}, there is no problem. However, there might be problems if the experiments are at different levels. There are actually two problems:

1. *Mutual confusion:* Suppose there are two receivers A and B at levels L_A and L_B respectively (say, $L_B < L_A$). If both A and B do an add-layer experiment simultaneously, and both get congested, then neither A nor B can figure out if it was due to its own experiment or it was due to someone else's experiment that it got congested.

2. *Third-Party confusion:* Suppose there are three receivers A (at level L_A), B (at level $L_B < L_A$), and C (at level $L_C \leq L_B$). If both A and B do an add-layer experiment, and as a result of which C gets congested (in addition to A and B being congested), C cannot figure out which experiment caused the congestion.

5.2 Extensions of the basic protocol

To apply the following extension, an extra step is involved in the *add – layer* operation: before trying to add the layer, the receiver needs to first inform the SA with message ADD_ACK and waits until after getting an ADD_REQ message that allows this experiment to take place.
SA - IA coordination: Before doing an experiment, the corresponding SA informs the IA, and if the IA receives more than one request, it can allow only one experiment to be conducted. In fact, all the IA needs to do is to inform the SAs about the selected experiment, and the time one must wait before requesting another experiment. In order to incorporate this mechanism into our basic protocol, step (b) of **add-layer** operation in section 4.2 needs to be modified. Right after the SA receives an ADD_REQ($L_i = SL_{max} + 1$) message, it sends an ADD_REQ($L_i = SL_{max} + 1$) message to the IA. If IA receives more than one ADD_REQ within a time duration Δ, it checks its knowledge base, and based on that, allows one or more concurrent experiments by sending ADD_ACK($L_i = SL_{max} + 1$, Δ_i) to the corresponding SAs. SAs then multicast ADD_ACK($L_i = SL_{max} + 1$, Δ_i) to their corresponding subnets exactly as in the basic protocol. The interaction between an SA and an IA is captured in Figure 2 by the feedback line from SA to IA and the compiled information line from IA to SA. Thus the IA can play an important role as a coordinator between experiments. IA can either:

1. resolve conflict between multiple experiments by choosing one of many possible experiments at any instant of time or
2. allow multiple experiments at the same time.

We have shown how an IA can choose one of many conflicting experiments at any instant of time. However, choosing one among several experiments is not necessarily the best thing to do in many situations. For example, suppose R_2 wants to do $L_2 \rightarrow L_3$ experiment, while R_6 wants to do $L_1 \rightarrow L_2$ experiment. In the current scheme, IA will allow R_6 to do its experiment and prevent R_2. However, these two experiments are mutually independent, because R_2's experiment will not change the load on link Ln_1, and hence will not affect subnet N_3. Similarly, R_6's experiment will also not affect subnet N_1. Thus these two experiments can be conducted simultaneously.

The above example illustrates that in order to make an intelligent decision, IA needs to maintain some information regarding which subnets are mutually dependent and which are mutually independent. Also, the IA needs to maintain the maximum level DL_{max} in its domain. DL_{max} is simply the maximum of $SL^{(i)}_{max}$ for all i in its domain. Note that the IA cannot schedule an experiment at the same time with an experiment that is an attempt to join beyond DL_{max}. For example, the IA can schedule both R_2's $L_2 \rightarrow L_3$ experiment and R_6's $L_1 \rightarrow L_2$ experiment at the same time, because none of these experiments try to go beyond DL_{max} which is 3. However, if $SL^{(4)}_{max}$ were 2 meaning that $DL_{max} = 2$, then R_2's $L_2 \rightarrow L_3$ experiment cannot be scheduled with R_6's $L_1 \rightarrow L_2$ experiment.

Knowledge-base compilation at an IA: Given that the IA needs to maintain some information for making intelligent decision, the question is how does it compile that knowledge base. The IA builds the knowledge base over time based on the feedback from the SAs of the various subnets in its domain after each experiment. For example, when R_8 does a $L_3 \rightarrow L_4$ join experiment, and link Ln_1 gets overloaded, all the subnets get congested. Thus IA correlates $L_3 \rightarrow L_4$ experiment with subnets N_1, N_2, N_3, and N_4, and creates an entry in its knowledge base to that effect. We introduce the term *virtual subnet* to indicate a subnet which connects all the affected physical subnets in a virtual plane. That is, if any host connected to the virtual subnet does a $L_3 \rightarrow L_4$ experiment, all the hosts on the virtual subnet will be affected. Note that the IA forms the virtual subnets based on the CONG(L_i) and FAIL $(L = SL_{max} + 1)$ messages that it receives in step (e) of the **add-layer** operation in section 4.1. The knowledge base will be further refined as lower level add-layer experiments are performed. For example, suppose R_2 does $L_2 \rightarrow L_3$ experiment, and subnet N_2 gets affected (because of overloading of link Ln_2), IA will correlate subnets N_1 and N_2 with the $L_2 \rightarrow L_3$ experiment; as if subnets N_1 and N_2 belong to the same virtual subnet for $L_2 \rightarrow L_3$ experiment. Thus, over time, the knowledge base at IA will have two entries:

1. Experiment: $L_2 \rightarrow L_3$; Virtual Subnet: N_1, N_2.
2. Experiment: $L_3 \rightarrow L_4$; Virtual Subnet: N_1, N_2, N_3, N_4.

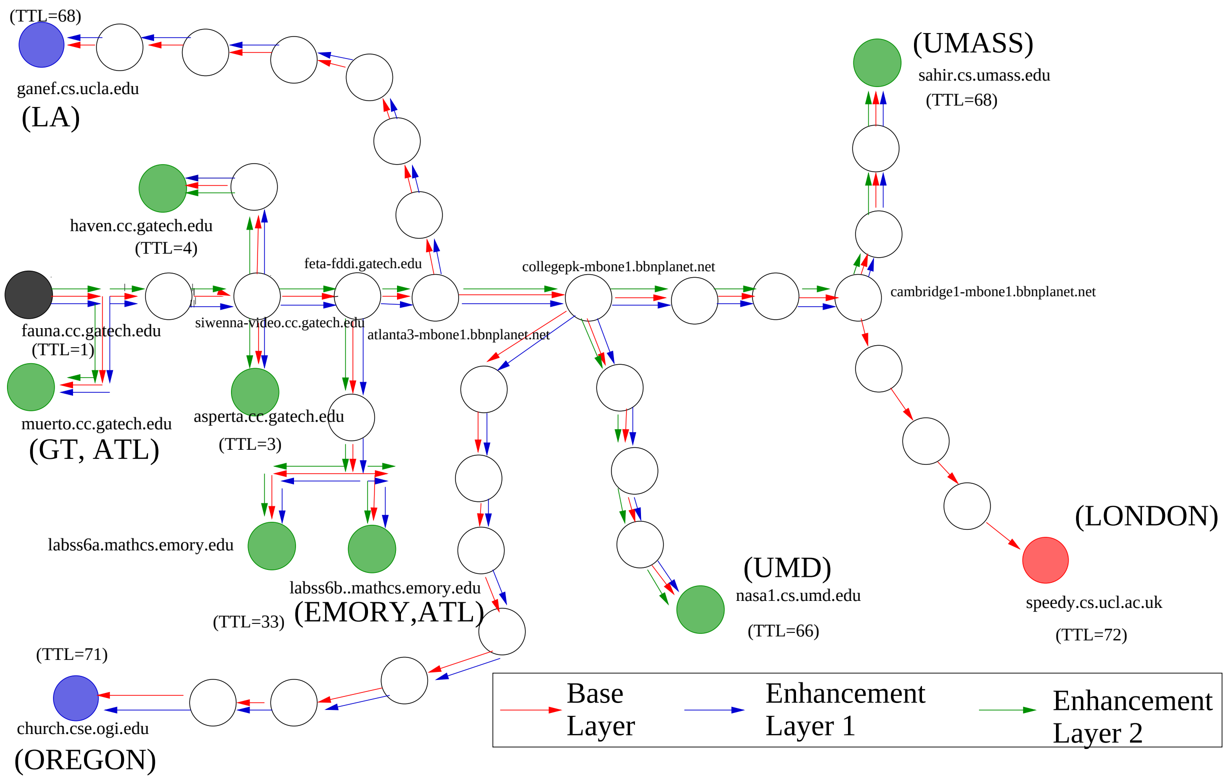


Figure 5: Topology of the Mbone Test and a Sample Video Layer Distribution

Using the knowledge base for making decisions: The knowledge base is used by the IA when deciding which experiments can be allowed to run concurrently in its domain. For example, based on the two entries in the knowledge base shown above, IA can schedule the following two experiments simultaneously:

(1) $L_2 \rightarrow L_3$ experiment by R_3 in subnet N_1 and

(2) $L_1 \rightarrow L_2$ experiment by R_7 in subnet N_3.

This is possible because experiment (1) will not affect N_3 as is clear from the knowledge base.

However, IA will not schedule the following two experiments at the same time:

(1) $L_2 \rightarrow L_3$ experiment by R_2 in subnet N_1 and

(2) $L_1 \rightarrow L_2$ experiment by R_4 in subnet N_2.

This is true because experiment (1) will affect N_2 as is obvious from the knowledge base.

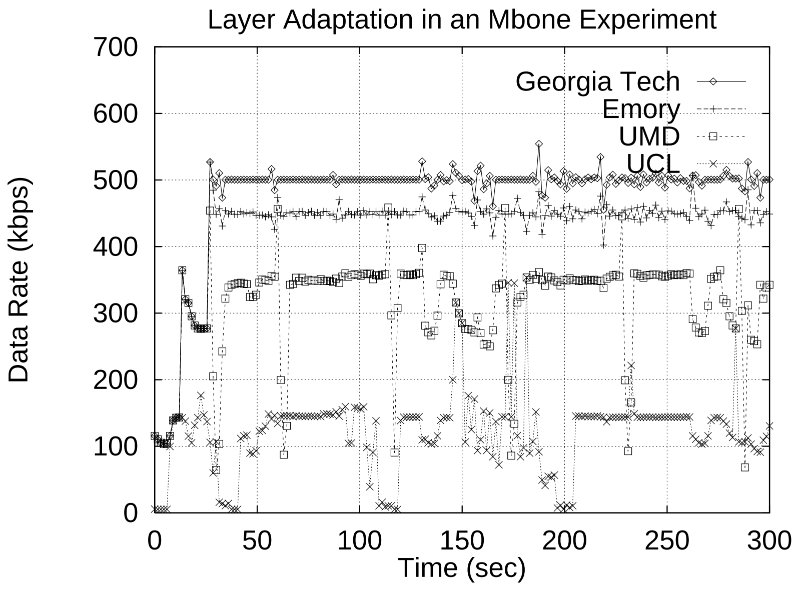


Figure 6: Video Layers in One Mbone Experiment

6 System Prototype Implementation and Experiments

We developed a prototype system for LVMR with IP multicast over the MBone, distributing MPEG II streams. The system is presented in more details in an earlier paper [LPPA97], which also addresses the retransmission-based error recovery schemes and the basic rate control.

Currently, the major features of the hierarchical rate control protocols are implemented in the system. Due to the limitation of the available testbed, it is hard to get the topology and network load desired to carry out experiments to test certain scenarios in our protocol. We will present an example of MBone experiments in this section. In the next section, we will show our simulation work to further test the protocols.

A sample topology of our experiments is shown in Figure 5. The video server is located in Georgia Tech, and receivers are spread over the continental US (Georgia Tech, Emory, UMass, UMD, OGI and UCLA) and Europe (UCL). This provides very heterogeneous network conditions and receiving capacity.

Figure 6 shows the data rates and video layers some of the receivers received during one Mbone experiment. The Mbone video was sent out from Georgia Tech with frame rate of 8 f/s. All the receivers started to join only the base layer and after a certain time, all receivers except the one in UCL joined the first enhancement layer, and later the second enhancement layer. The Georgia Tech and Emory receivers both ended up receiving all three layers, but the node in Georgia Tech observed almost no loss while the receiver in Emory had some constant loss around 10%. The

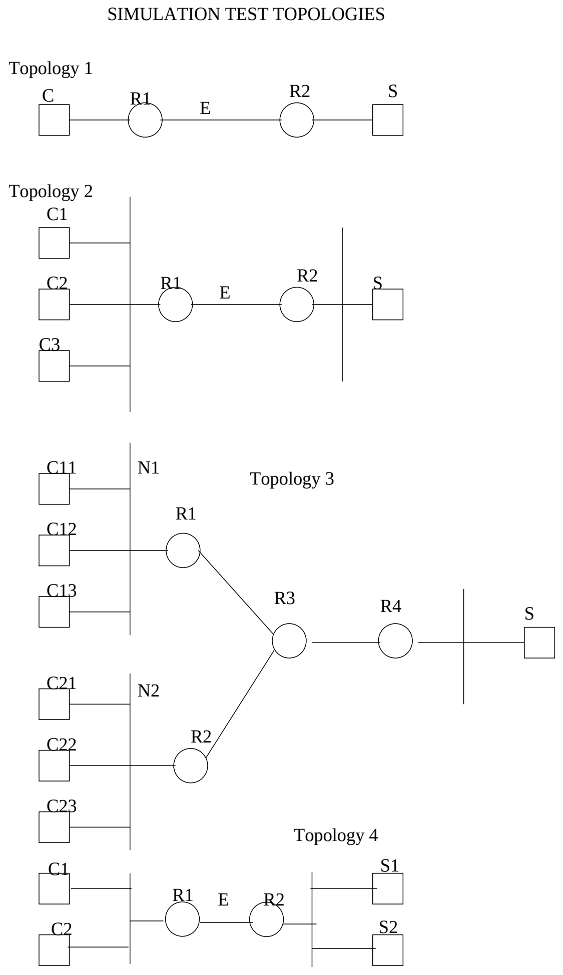


Figure 7: Topologies of Simulation

machine in UMD received 2 layers most of the time although it launched add-layer experiment from time to time. The receiver in UCL suffered a loss of around 20% and it could only join the base layer.

7 Simulation

We simulated the hierarchical rate control protocol with a modified *ns* [MF] version 1.4. We used 36000 frames of an MPEG II trace to produce 4 layers with the following layering scheme: given the GOP pattern of $IB_1B_2PB_3B_4PB_5B_6PB_7B_8$, I frames make the base layer, P frames the first enhancement layer, B_1, B_3, B_5 and B_7 the second enhancement layer and the rest of the B frames the third enhancement layer.

Figure 7 lists some of the topologies we used in the simulation tests. Unless otherwise specified, each link is a duplex link with bandwidth of 10Mb and delay of 0. Video frames are sent out in packets of 1k bytes at a speed of 24 frames per second or 2 GOPs per second. In the tables and plots in this section, *data rate* is the averaged data rate over every half second.

Bandwidth	Stable	Stable Data Rate(kbps)			Stable Loss Ratio		
(kbps)	Level	25kbyte	50kbyte	100kbyte	25kbyte	50kbyte	100kbyte
400	1	374.78	379.94	390.12	0.050	0.039	0.027
500	1	396.87	408.19	418.86	0.010	0.009	0.014
600	1	399.39	411.74	429.17	0.008	0.007	0.009
700	1	410.28	421.59	449.98	0.007	0.006	0.014
800	1	414.76	466.63	499.09	0.008	0.015	0.021
900	2	887.23	894.66	900.00	0.111	0.103	0.098
1000	2	965.82	973.0	9981.72	0.040	0.033	0.025
1100	3	1100.07	1100.06	1100.08	0.076	0.076	0.077
1200	3	1184.48	1191.33	1200.56	0.010	0.012	0.009
1300	4	1300.19	1298.01	1301.45	0.128	0.124	0.127
1400	4	1400.13	1400.56	1401.42	0.064	0.064	0.064
1500	4	1500.03	1500.52	1501.37	0.001	0.001	0.000

Table 4: Stable Level, Data Rate and Loss Ratio vs Network Bandwidth

The state transition parameter default values in the simulation tests unless otherwise specified are (all the time parameters are in the unit of GOP, i.e. half second) $T_t = 8, T_{c,0} = 4, T_{c,add} = 1, T_{u,min} = 16, T_{u,max} = 64, \alpha = 2, R_c = 0.03$, and $R_u = 0.12$.

The *stable level* of a receiver is the highest number of video layers it can join and keep receiving for a considerably long time. In the next set of experiments, we look at how bandwidth and buffer size affect the stable level, and the data rate and loss ratio at the stable level.

7.1 Topology 1: The Basic Layer Adaptation.

In this set of tests, there are one receiver C and one sender S connected by two routers R1 and R2. Let E denote the unidirectional link between R_2 and R_1, The following two parameters are changed to see how they affect layer adaptation: (2) the bandwidth of E, and (2) the buffer space in router R_2 that keeps packets going to link E (we will call it buffer space of E). Background traffic can also be added to link E using an *ns* constant bit rate traffic generator.

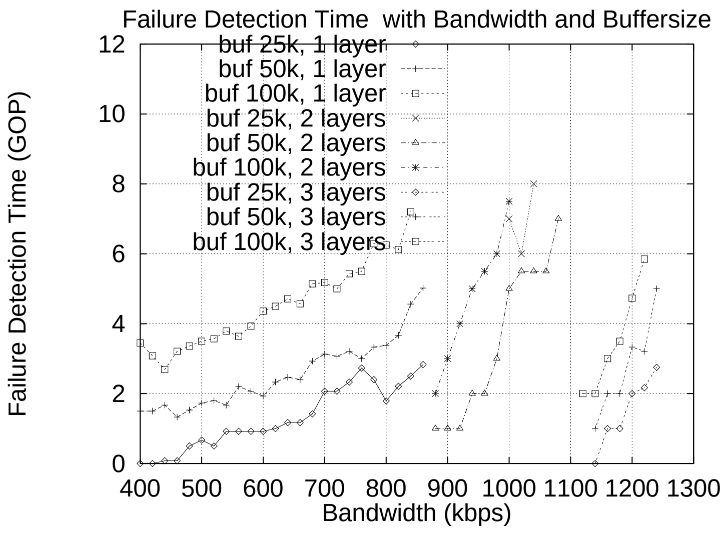
1. Bandwidth, buffer size and stable video level.

   Table 4 shows what stable video level can be achieved with a given bandwidth and buffer size at link E. The data rate and loss ratio are their averaged values when the receiver is receiving stable layers and 36000 frames are transmitted in each test. The higher the link bandwidth, the higher the stable video level, and when the link can provide bandwidth of 1.5mb, all 4 layers can be subscribed and received with virtually no loss. At the same video level, a higher bandwidth link results in higher video reception data rate at the receiver and lower loss. With the same bandwidth, a larger buffer usually leads to higher data rate and lower loss ratio at the receiver, but the difference is not significant.

2. Bandwidth, buffer size and failed add-layer experiments.

Bandwidth	Stable	Congestion Data Rate(kbps)			Congestion Loss Ratio		
(kbps)	Level	25kbyte	50kbyte	100kbyte	25kbyte	50kbyte	100kbyte
400	1	NA	180.29	172.32	NA	0.557	0.620
500	1	218.88	200.18	177.38	0.527	0.523	0.552
600	1	223.30	229.54	193.97	0.529	0.434	0.521
700	1	269.04	213.20	217.71	0.372	0.475	0.464
800	1	286.07	256.40	320.80	0.359	0.412	0.369
900	2	NA	902.42	880.24	NA	0.207	0.215
1000	2	NA	823.23	819.58	NA	0.260	0.244
1100	3	NA	NA	NA	NA	NA	NA
1200	3	927.36	938.60	918.72	0.262	0.238	0.242

Table 5: Add Layer Failure: Data Rate and Loss Ratio vs Network Bandwidth



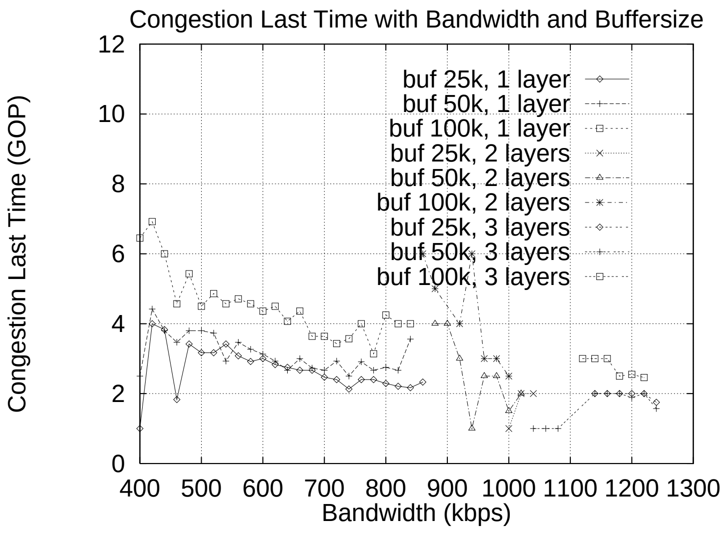


Figure 8: Failure detection time and congestion last time.

When a receiver's stable level is not the highest video layer, from time to time, this receiver possibly conducts *add-layer* experiments that turn out to be failures. Although it has bad side effect, this kind of experiment is necessary so as to dynamically determine the time to join more layers when extra network bandwidth becomes available.

A failed experiment brings congestion to the network and results in decreased video data rate and increased packet loss for a short period of time. To study the effect of a failed add-layer experiment, we measure the following 4 parameters for each failure: (1) the averaged data rate during the congestion, (2) the averaged loss ratio during the congestion, (3) t_{dect} failure detection time: the time between a layer is added and the congestion is detected, and (4) t_{last} congestion last time: the time between dropping the added layer and congestion going away.

Table 5 shows how bandwidth and buffer size affect data rate and loss ratio during the failed add-layer experiments. At the same video level, a receiver linked to a higher bandwidth link experiences less loss and remains at higher data rate during the experiment. For example, with the same 100 kbytes buffer size and the same video level 1, failed experiments with a 400kbps link result in an average packet loss of 0.62, while the loss ratio is only 0.46 with a 700kbps link.

Figure 8 illustrates the change of the failure detection time t_{dect} and congestion last time t_{last}, given different bandwidth and buffer sizes. When receiving the same level of video layers, the receivers linked through a higher bandwidth link or a link with a larger buffer, usually has a longer failure detection time. At the same video level, a higher bandwidth link brings shorter congestion last time and a larger link buffer results in longer congestion last time.

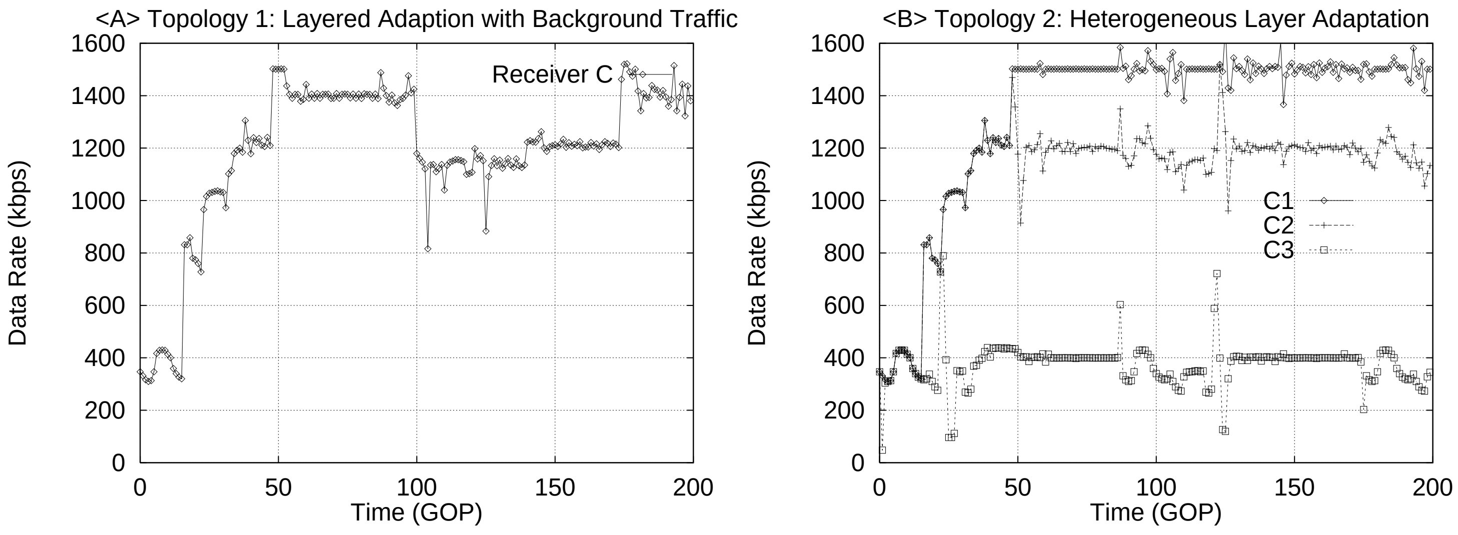


Figure 9: Layer Adaptation in Topologies 1 and 2

3. Congestion and Layer Drop

   In the simulation system, to generate background traffic for each link, a traffic generating node can be attached to the router at the start point of the link. For example, in topology 1, to add background traffic to link E, an extra node is added to router R_2 and it sends packets to router R_1. Thus the background traffic shares the same queue in R_2 as the video traffic from S to C.

   Figure 9A shows the video layer adaptation when network load is increased. Constant bit rate traffic with packet size of 1 kbytes and inter-packet time 0.03 second is generated on link E at time 100 GOP. The receiver gets a very high loss ratio of 0.3 and transitions to the $CONGESTED$ state and remains there for 4 GOPs, then it drops layer 4. Receiving only 3 layers, the loss rate is dropped to almost 0. The receiver conducts an add-layer experiment at GOP 125 but fails. At GOP 160, the background traffic is terminated and at GOP 170, the receiver conducts another add-layer experiment and joins layer 4.

7.2 Topology 2: Heterogeneity and Collaborative Layer Drop

In topology 2, there are 3 receivers on the same subnet and they have heterogeneous video processing power due to the difference in their CPUs, current machine load, and so on. To simplify the problem, the receiver's video processing power is identified with a processing speed (in kbps) and a buffer (in kbps) to absorb the jitter. Receivers C_1, C_2 and C_3 have processing speed of 1500 kbps, 1200 kbps and 400 kbps with the same buffer size of 50 kbytes.

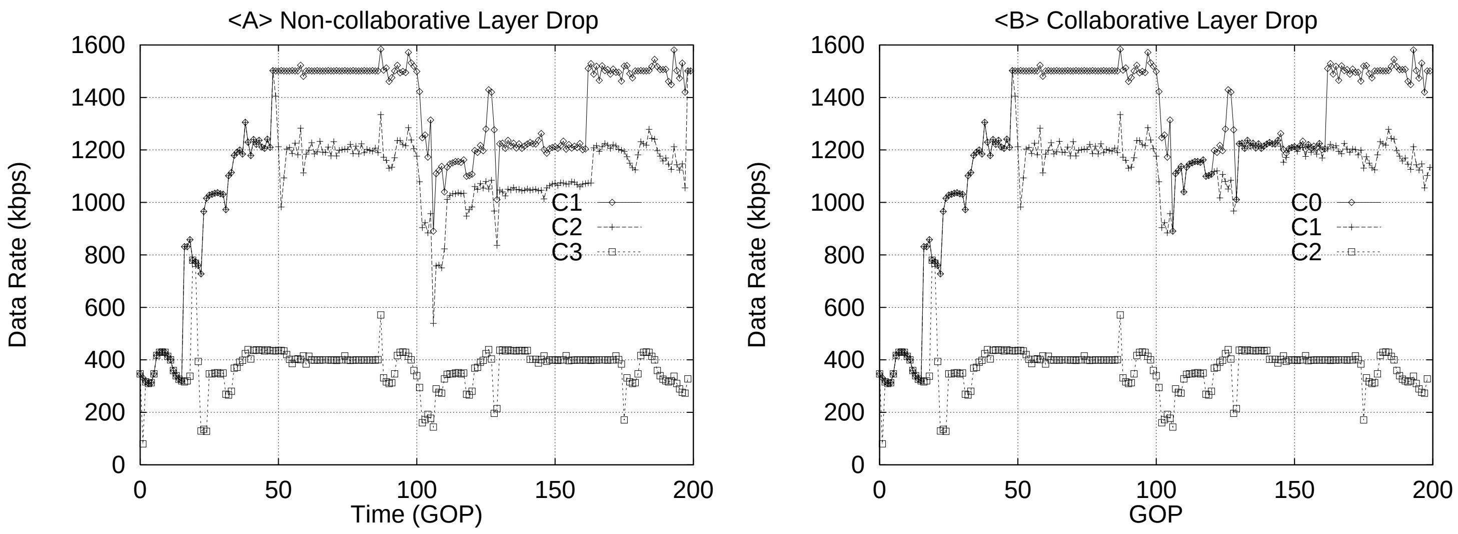


Figure 10: Layer Drop: Non-collaborative and Collaborative

- Layer adaptation in heterogeneous environment.

  Figure 9B illustrates how the heterogeneous receivers adapt to the stable video levels. They all start with 1 layer and after the initial adaptation, C_1 receives all 4 layers, C_2 receives 3 layers and C_1 receives 1 layer.

- Collaborative layer drop.

  Figure 10 shows how the receivers adapt their layers when the shared link between R_2 and R_1 is congested. The congestion is caused by constant bit background traffic (1-kbyte packet every 0.03 sec) on E between time 100 GOP and 160 GOP.

  When the background traffic is added to the shared link, all three nodes observe increased loss ratio and transition to the $CONGESTED$ state. In Figure 10A, C_2 drops layer 3 and C_3drops layer 4, and they get recovered when the side traffic ends. While in Figure 10B, *collaborative layer drop* is achieved by letting C_2 increase its T_c^3 when it hears the CONG messages from C_1. Thus, when C_1 drops layer 4, C_2 still remains in 3 layers, and then the congestion goes away after layer 4 is dropped. So C_2 remains receiving 3 layers instead of dropping to 2 and thus avoids the unnecessary quality degradation and layer fluctuation.

7.3 Topology 3: Shared Learning and Add-layer Synchronization

Topology 3 includes 2 subnets N_1 and N_2 each with 3 receivers, and the IA is set to be on receiver C_{22}. The video processing speed on C_{11} and C_{21} is 2.0mbps and 400kbps on other receivers. C_{21} joins the video session late at time 20 GOP, and it adds layers up to 3 at time 54 GOP, while by then C_{11} receives 3 layers and has already conducted a failed add-layer experiment to join layer 4.

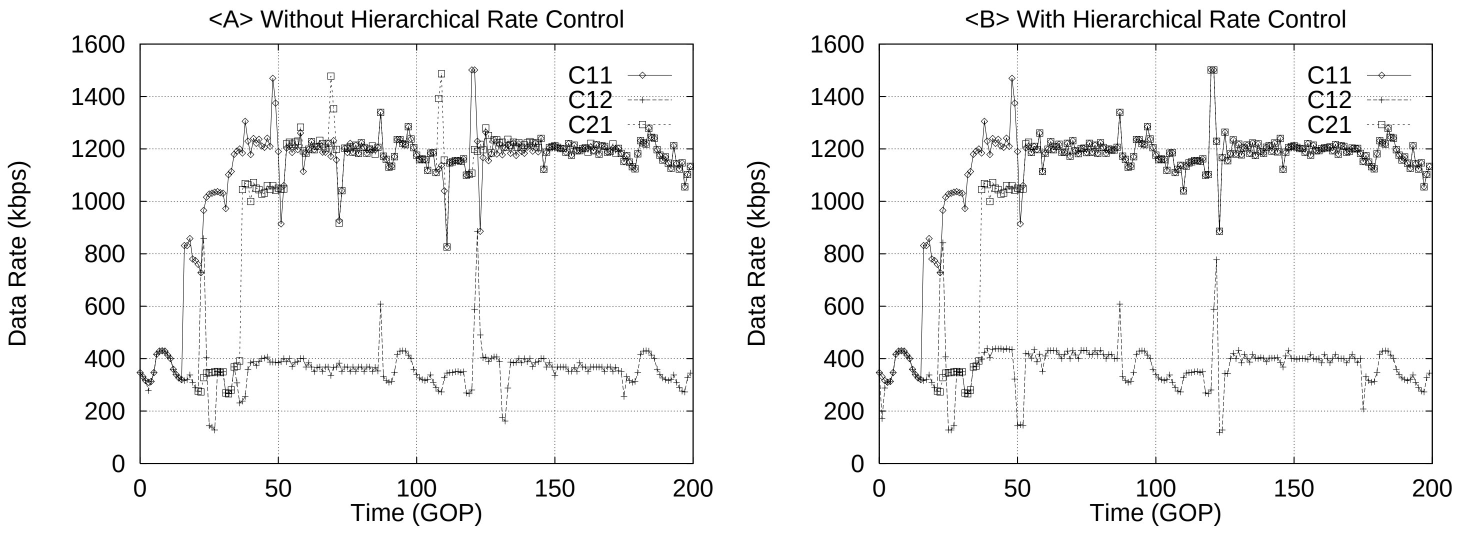


Figure 11: Add Layer Experiments

In Figure 11B, with shared learning message exchanging in hierarchical rate control, the failure of the first experiment by C_{11} is passed to the IA, the SA on net N_2 and then to C_{21}. So C_{21} gets extended T_u^3 and its add layer experiment is synchronized with that of C_{11}. While in Figure 11A where no hierarchical rate control is applied, C_{21} would try ani add-layer experiment soon after joining layer 3, and again at time 120 GOP, while both of the experiments are failed and they should be avoided as shown in Figure 11B.

7.4 Topology 4: Multiple Video Sessions

In this topology, there are two receivers, C_1 and C_2, and two senders S_1 and S_2. Two video sessions are set up, Session 1 from S_1 to C_1 and Session 2 from S_2 to C_2, each running with different sections of the video trace. Link E from R_2 to R_1 is shared by both sessions.

In Figure 12, the bandwidth of the link E is set to 1.5mb and Session 1 starts first. Since E provides enough bandwidth for this session, C_1 joins all 4 layers. At time 100 GOP, Session 2 starts and C_2 joins layer 1, receiving less than half of the packets. C_1 gets congested and drops layer 4 soon after Session 1 starts and then remains stable with 3 layers. C_2 keeps receiving only 1 layer and the average loss ratio gets around 0.2. The results shows that when a new video session and an ongoing session compete for bandwidth, the ongoing session tends to drop an enhancement layer and the new session can obtain the bandwidth for the base layer. But the new session will not further obtain bandwidth to achieve the same video reception level as the ongoing session.

8 Concluding Remarks

The layered video multicast with retransmission (LVMR) is a protocol designed to operate over a best-effort packet-switched network with heterogeneous multicast receivers – heterogeneity arising from either limited computing resources of the hosts or from dynamically changing bandwidth of the network interconnecting them. LVMR has two novel features: 1) the use of retransmissions in

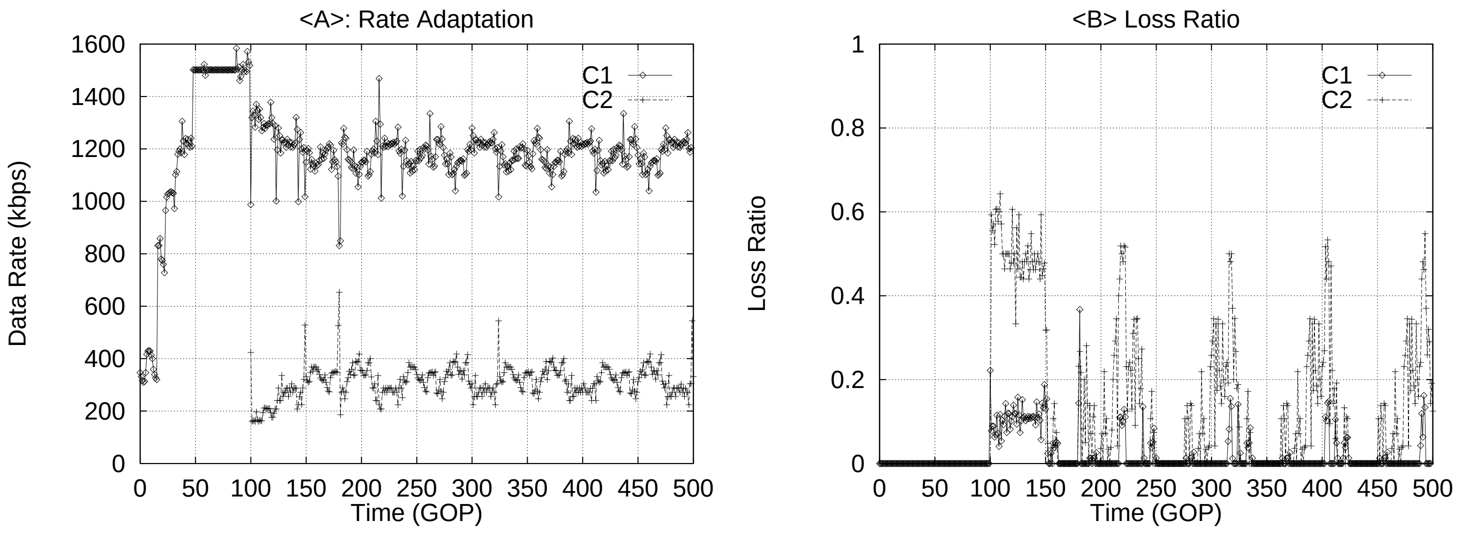


Figure 12: Multiple Session Experiment

a layered environment and 2) the use of hierarchical control to manage the adding and dropping of video layers by receivers. Our previous work [LPPA97] has considered the first aspect while this paper focusses on the second. We describe the insight behind the mechanisms we propose for our protocol and then describe in some detail the operation of the basic protocol and some proposed extensions. Our experience with an implementation and its use over the Mbone are also described. In addition, we present the results from a simulation of the hierarchical control aspects of the protocol. While hierarchical control does have some additional overhead compared to a purely receiver-driven approach, we believe the overhead is negligible both in terms of complexity and in terms of additional load on the network. In addition, our simulation results show significant performance improvement at a minimal cost and that may well justify the additional overhead in a real video multicast system.

There are several things that need to be done in future to evolve LVMR as a complete solution for real-time video distribution over the Internet. First of all, the hierarchical rate control protocol needs to be fully implemented and tested on the Internet. There is no substitute for real measurements. The complete hierarchical solution including a dynamic choice of IAs needs to be investigated further. Other possibilities include exploring the role of sender in LVMR in terms of dynamically adjusting layer quality, and combining or splitting layers for congestion avoidance and for accommodating heterogeneity.

References

[BFC93] T. Ballardie, P. Francis and J. Crowcroft,"Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing," *Proceedings of ACM SIGCOMM '93*, September 1993.

[BFGH+95] R. Bettati and D. Ferrari and A. Gupta and W. Heffner and W. Howe and M. Moran and Q. Nguyen and R. Yavatkar Connection Establishment for Multi-Party Real-Time

Communication, *NOSSDAV 1995* pages 255–266.

[BT94] Jean-Chrysotme Bolot and Thierry Turletti. A Rate Control Mechanism For Packet Video in the Internet. In*IEEE Infocom-94 Sumposium*, Vol 3, pages 1216-1223.

[BTW94] J-C. Bolot, T. Turtelli, and I. Wakeman,"Scalable Feedback Control for Multicast Video Distribution in the Internet," *Proceedings of ACM SIGCOMM '94*, Pages 58-67, September 1994.

[CAL96] S.Y. Cheung, M.H. Ammar, and X. Li, "On the Use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution," *Proceedings of IEEE INFOCOM '96*, Pages 553-560, March 1996.

[D88] S.E. Deering, "Multicast Routing in Inter Networks and Extended LANs," *ACM Computer Communications Review*, Vol. 19, No. 4, 1988, Pages 55-64.

[D89] S.E. Deering, " RFC-1112: Host Extension for IP Multicasting," August, 1989.

[DC90] S.E. Deering and D. R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Transactions on Computer Systems*, Vol.8, No.2, Pages 85-110, May 1990.

[DEF+94] S. E. Deering, D. Estrin, D. Farinacci, V. Jacobson, C-G Liu and L. Wei, "An Architecture for Wide-Area Multicast Routing," *Proceedings of ACM SIGCOMM '94*, Pages 126-135, October 1994.

[FBZ94] Domenico Ferrari,Anindo Banerjea and Hui Zhang. Network Support For Multi- media. *Computer Networks and ISDN Systems,* Vol.20, 1994, Pages 1267–1280.

[KMR93] Hemant Kanakia, Partho P. Mishra and Amy Reibman. An Adaptive Congestion Control Scheme for Real-Time Packet Video Transport. In*ACM Sigcomm-93* Symposium, pages 20-31, September 1993.

[LA96] Xue Li and Mostafa H. Ammar. Bandwidth Control for Replicated-Stream Multicast Video Distribution. *HPDC 96*

[LPPA97] Xue Li, Sanjoy Paul, Pramod Pancha and Mostafa Ammar, Layered Video "Multicast with Retransmission (LVMR): Evaluation of Error Recovery", *Proceedings of NOSSDAV 97*, St Louis, May 1997

[MF] S. McCanne, and S. Floyd. *The LBNL Network Simulator*, Lawrence Berkeley Laboratory. Software on-line: http://www-nrg-ee.lbl.gov/ns.

[MJV96] S. McCanne,V. Jacobson and Martin Vetterli, "Receiver-Driven Layered Multicast," *Proceedings of ACM SIGCOMM '96*, October 1996.

[Sha92] Nachum Shacham. Multipoint Communication by Hierarchically Encoded Data. *Infocom 92.*Pages 2107 - 2114.

[ZDE+93] L. Zhang, S. Deering, D. Estrin, S. Shenker and D. Zapppala, "RSVP: A New Resource ReSerVation Protocol," *IEEE Network Magazine*, September 1993.