A Simple Loss Differentiation Approach to Layered Multicast

R. Gopalakrishnan, James Griffioen, Gisli Hjálmtýsson, Cormac J. Sreenan, and Su Wen

Networking and Distributed Systems Research
AT&T Labs - Research
Florham Park, NJ 07932

Abstract—**Layered multicast is a promising technique for broadcasting adaptive-quality TV video to heterogeneous receivers. While several layered multicast approaches have been proposed, prior work has identified several problems including significant and persistent instability in video quality, arbitrary unfairness with other sessions, low access link utilization due to conservative bandwidth allocation, and problems with receiver synchronization.**

In this paper we propose a new layered multicast scheme, where we exploit a simple, coarse-grained, two-tier loss differentiation architecture to achieve stable and fair bandwidth allocation for viewers. Despite the simplicity of our loss differentiation model, we show that it achieves most of the benefits of complex and costly priority dropping schemes. In addition, our protocol is receiver-driven and thus retains the incentives to limit bandwidth usage that are not present in existing priority dropping schemes.

Keywords—**multicast, layered video, RLM, priority-dropping**

I. Introduction

The Internet is rapidly becoming the next global network infrastructure, supplanting special-purpose telephony and TV networks. Although originally designed for data transport, IP-based networks are increasingly being used to deliver multimedia services. This introduces new challenges, since media streams require higher levels of service quality and service stability than traditional data transport. Unlike telephone and TV networks, IP-based networks are heterogeneous, with receivers differing in processing capabilities, link capacities, and network connectivities. Consequently, no single fixed bandwidth media stream will be optimal for all receivers. In addition, network load and traffic conditions (losses) can change dramatically, and rapidly. The ability of the Internet Protocol (IP) and its applications to cope with heterogeneity and adapt to changing network conditions has been a key to its success.

Our study is motivated by the desire to transmit medium-scale TV broadcast (few hundreds to tens of thousands of viewers) over IP-based networks to heterogeneous receivers. We are primarily interested in (live) broadcast events of interest to a large audience but not large enough (or co-located enough) to meet the threshold needed to be carried (cost-effectively) by conventional TV networks over satellite or cable to millions of viewers. Example broadcast events include distance learning and training sessions (*e.g.*, televised university classes), special events (*e.g.*, concerts, lectures or sporting events such as a European soccer match), and focused-community events (*e.g.*, large conferences or meetings). Although users are aware of their physical bandwidth limitations, they will still expect TV-like characteristics from their video such as consistent quality. Frequent (observable) quality changes quickly become annoying. Furthermore, we expect multiple IP-based broadcast sessions will occur simultaneously and will compete for the network bandwidth, much like current TV networks broadcast multiple channels at the same time. As a result, it is important that bandwidth is allocated fairly among the multicast sessions.

An appealing approach that accommodates heterogeneous receivers and adapts to congestion is to encode the video stream onto multiple layers and transmit each layer on its own multicast group. The set of layers comprising a video stream constitute a *session*. Receivers of a session subscribe to as many layers as network conditions and receiver capabilities allow. Several schemes for layered multicast have been proposed, including [MJV96], [WSS97], [LPA98], [VRC98], [TPB97].

Layered multicast protocols typically use a *receiver-driven* approach in which the end-systems decide which layers should be delivered. Protocols such as *Receiver-Driven Layered Multicast (RLM)* protocol [MJV96] have been analyzed and evaluated by several researchers [MJV96], [BBS98], [GGHS99], [WSS97], [RKT99]. In our own prior work [GGHS99], we found that receiver-driven schemes such as RLM exhibit significant and persistent *instability* and arbitrary *unfairness* in video quality. Moreover, these problems are aggravated with *scale*.

An alternative approach is to use a *priority dropping* scheme [BBS98] where the network, rather than the receivers, decides which layers should be delivered. When congestion arises, routers in the network drop the least important packets first. An important benefit of priority dropping is stable and fair allocation of bandwidth. Because packet losses are concentrated at the highest layer(s) [1] (given their low priority), the highest layers absorb the majority of transient losses caused by short term congestion. As a result, lower layers are protected from loss so receivers experience stable reception of the most important layers. Network-based priority dropping also results in fair bandwidth allocation if all sessions employ the same layering scheme. [BBS98] developed a "utility model" to compare RLM with priority dropping. They showed that priority dropping maximizes the average "utility" by delivering the most important packets. However, they concluded that the improvement was less than expected and may not be justified given the drawbacks. First, implementing multiple drop priorities in the network adds significant complexity to routers. Second, there are no incentives for receivers to unsubscribe to higher layers. Third, fairness depends on all sessions using the same layering scheme. In short, past work suggests that receiver-driven ap-

[1] In this paper we use the terms *base layers* or *lowest layers* to refer to the most important layers. The *highest layers* are the least important.

proaches are not as bad as expected and the cost/benefit ratio of the alternative, namely priority dropping, does not justify its deployment.

In this paper we analyze the pros and cons of both approaches and then propose a hybrid approach that combines the best features of both. Our new scheme is based on a coarse-grained loss-priority mechanism located in the network that achieves the benefits of priority dropping without the complexity or layer dependency of strict priorities. Our layered multicast transport protocol is a modified version of RLM that uses the simple two-level network priority mechanism to protect lower layers from burst loss, avoid disruptive join-experiments, and improve synchronization among receivers. The use of a coarse-grain (i.e., two-level) drop-preference rather than arbitrary multilevel priorities means our scheme is simple to realize and retains incentives for receivers to unsubscribe from unneeded layers. We show that in spite of its simplicity, our scheme results in a fair and stable bandwidth allocation.

In this paper, we begin our discussion of layer multicast by analyzing receiver-driven approaches. Our analysis uses RLM as an example receiver-driven approach because it has been widely studied, code for RLM is readily available, and it is designed to have good stability properties. Section III examines network-based priority dropping schemes and their benefits/drawbacks. Section IV then introduces our new hybrid approach based on Receiver-Selectable Loss Priorities (RSLP) and Receiver-driven Layered Multicast with Priorities (RLMP). Sections V and VI describe the implementation details of the RSLP and RLMP. Section VII evaluates RLMP and demonstrates its stability and fairness properties. Section VIII summarizes related work on layered multicast and Section IX offers our conclusions.

II. RLM

The Receiver-driven Layered Multicast protocol (RLM) [MJV96] is designed to deliver layered video to a heterogeneous set of receivers. In RLM, the source encodes the video signal onto multiple discrete layers where each layer incrementally refines the layer below it. Each layer is transmitted on a separate IP multicast group. To cope with bandwidth heterogeneity and adapt to changing congestion conditions, a receiver selectively subscribes to as many layers as its access bandwidth and network conditions will permit. RLM is designed to exploit existing IP multicast capabilities, and does not require any new mechanisms within the network. Moreover, the RLM protocol is transparent to senders and runs only at receivers.

When RLM detects sustained losses it drops a layer in an attempt to reduce the congestion. To learn of (newly) available bandwidth, RLM periodically conducts a *join-experiment* that probes the network by adding the next layer. If the join-experiment produces congestion, RLM concludes the bandwidth is not available, drops the new layer, and doubles the time before the next join-experiment. Because join and leave operations do not take effect immediately, RLM also maintains a detection timer T_D that estimates the amount of time between the start of a join experiment and the onset of congestion.

RLM is designed to avoid frequent subscription level changes. If a join fails, the receiver becomes more reluctant to add a layer. If a receiver has been stable for some time, it becomes more conservative about dropping a layer. When congestion arises, RLM waits for an interval of $2 \times T_D$ before making a decision to avoid reacting to transient congestion. If the congestion persists to the end of this interval it drops a layer, otherwise it returns to the stable state. These delays are meant to dampen state transitions and prevent thrashing. To scale to large group sizes, RLM also incorporates other optimizations. An example is "shared learning" within a session where the failure of a join experiment conducted by a receiver is inferred by other receivers thus avoiding the need for separate disruptive join experiments.

In [GGHS99] we analyzed the stability and fairness of RLM, and the ability of RLM to effectively utilize access bandwidth. As this study is the primary motivation behind our new protocol, the rest of this section summarizes the main results reported in [GGHS99].

A. Stability of Receiver-driven Layered Multicast

An important metric for evaluating video distribution protocols and end-user experience is the *stability* of the service quality. Since RLM's adaptation mechanism is based on adding/dropping layers, we measure stability as the rate at which the subscription level at a receiver changes. Ideally, we would like to see this rate converge to zero, and at a time scale significantly smaller than the time scale of substantial changes in the total traffic load. While short term congestion is inevitable in a best effort network and can sometimes be masked by the application, persistent fluctuation in the subscription level is evidence that the protocol is unable to converge to the appropriate level.

In [GGHS99] we measured stability of RLM using three different types of traffic sources; constant bit rate (CBR) sources, and two types of variable bit rate (VBR) sources, of medium and high variability denoted VBR-3 and VBR-5 respectively. Details of how these sources are defined is given in section VII. A characteristic of RLM worth noting is that in an effort to enhance stability, receivers that have been at the same level for some period of time will not react quickly to observed congestion.

As expected, the CBR sessions converge rapidly and then remain stable. The same does not hold for the VBR sources. Both VBR-3 and VBR-5 experience many level changes throughout the simulation causing receivers to observe a wide range of video quality. Figure 1 plots the subscription level changes over time for two simultaneous VBR-3 sessions sharing a bottleneck link. [2] The figure shows a period in which the video quality becomes four times better in less than a four minute window. We observe similar quality swings with VBR-5. Another interesting phenomena is how the receiver's quality "flip-flops" over time. This occurs in both the VBR-3 and VBR-5 tests. In the VBR-3 test, receiver 1 initially receives four layers while receiver 2 only receives two layers (e.g. time 400-650). As time progresses the two trade places with receiver 2 receiving four layers and receiver 1 only getting two layers (e.g. time 920-1050). In between (at time 800) the system is running at the "fair" allocation of 3 layers for each receiver. From a stability standpoint, this is a significant protocol deficiency.

[2] To avoid "minor" changes, each point on the subscription level graphs re-

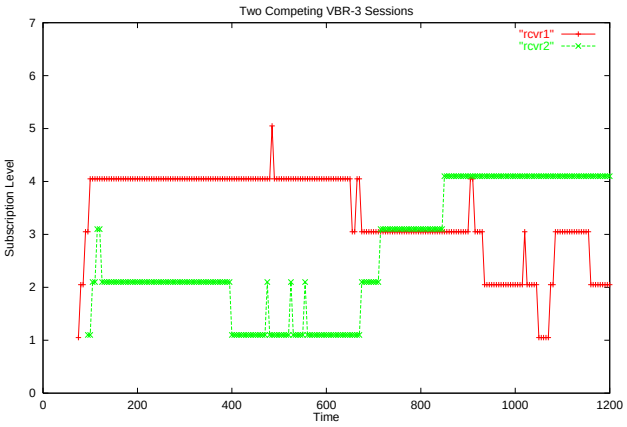

Fig. 1. Subscription level over time for VBR-3 traffic.

If RLM were stable, one would expect to observe an initial series of frequent layer changes followed by longer and longer durations without any layer changes. This is indeed the case for CBR traffic. However, the VBR traffic in Figure 1 shows level changes continue to occur throughout the simulation with (apparently) the same frequency. These persistent level changes are confirmed by Figure 2, which shows the cumulative number of level changes that occur during each 5-second interval for 4, 8, and 16 simultaneous sessions. Each is normalized to show the cumulative number of level changes per receiver. The linear growth of the counting process confirms that the level changes persist.

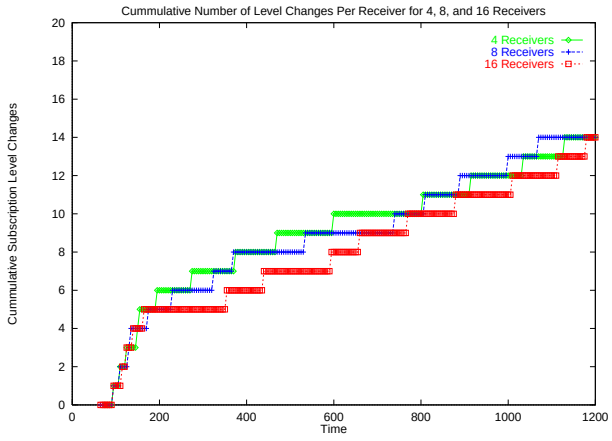

Fig. 2. Stability Comparison: the average number of layer changes per receiver (cumulative) over time for different numbers of sessions.

B. *Fairness among competing RLM sessions*

We define *fair* allocation as equal allocation to all sessions sharing a bottleneck link. Ideally the protocol (RLM) would provide both fair *and* stable video quality. However, these properties are conflicting, so that solving fairness only may lead to gross instability. Conversely, some of the obvious ways to enhance stability of RLM (such as waiting for two detection times before dropping a layer) do so at the expense of fairness. In general, fairness is hard to achieve using RLM since receivers are oblivious of other sessions in progress, of their layering schema and their current state, and cannot distinguishing between transient congestion and long term load changes.

To quantify the unfairness of RLM, we ran multiple RLM sessions sharing a single bottleneck. We find that with CBR traffic, unless the receivers subscribe at virtually the same time, the first one fills the pipe, subscribing to as many of its layers as will fit. Subsequent sessions initially subscribe to and get their base layer but are unable to ramp up beyond that. With VBR traffic, the instability of RLM conspires to remove the persistent unfairness in allocation and removes the sensitivity to initial conditions. However, while the long term average bandwidth of different sessions is almost the same, on shorter timescale the system remains grossly unfair.

flects the average subscription level over a five second interval.

III. Network-based Priority Dropping

Another approach for implementing layered video transmission is to use priority dropping in the network. Each router in the network implements a packet discard policy which, during periods of congestion, drops lower priority packets before higher priority packets. By assigning the highest priority to the base layer and successively lower priorities to each additional layer, losses during short term congestion are confined to the enhancement layers without affecting basic layers. Consequently, priority dropping is very effective at reacting to transient congestion.

In contrast, receiver-driven approaches cannot adapt to congestion at packet time scales since multicast leave latencies are in the order of seconds. In addition, all receivers downstream of the bottleneck must leave the group to stop packets from being forwarded over the congested link. Likewise, probing for newly available bandwidth is a slow time scale operation, since it is timer driven and involves joining a multicast group.

While it may seem that priority dropping has several advantages, it has several significant drawbacks:

- Implementing priority dropping at routers is considerably more complex than existing droptail schemes. Multiple priority levels aggravate this further. Routers need to examine the priority header field and ensure that the lowest priority packet is discarded.
- Priority dropping provides no incentives for receivers to lower their subscription level. Packets priorities are set by the sender; the dropping is done by routers. Without external disincentives (*e.g.*, monetary disincentives) the receivers would remain subscribed to all layers, causing extraneous traffic to be needlessly carried on parts of the network, and extraneous multicast state information to consume space in routers; all yielding little or no value to receivers.
- Priority dropping requires that the network support as many loss priority levels as layers. To ensure fair allocation of resources, it also requires that each session uses the same set of priorities.

In [BBS98], the authors compared the performance of a priority dropping scheme with a uniform dropping scheme and the receiver-based RLM. They concluded that priority dropping can achieve a modest improvement in *utility* over uniform dropping, but show gains of 50% to 100% over RLM in many circumstances.

IV. A new Approach Based on Simple Loss Differentiation

In this section, we introduce a new approach for receiver driven multicast, based on simple loss differentiation. Our approach is motivated by our desire to maintain the simplicity and receiver incentives of RLM while overcoming the instability and unfairness properties. In addition, we are motivated by the effectiveness of priority dropping in reacting to short term congestion. The goal of our design is to achieve stable and fair layer allocation, while avoiding the implementation complexity of traditional priority dropping mechanisms inside the network. Our solution consists of two parts: *Receiver-Selectable Loss Priorities* (RSLP) - simple router mechanisms to support two-level loss priority and priority sensitive multicast joins, and *Receiver-driven Layered Multicast with Priorities* (RLMP) - a new receiver protocol exploiting loss differentiation.

Receiver-Selectable Loss Priorities: We propose a simple two-level drop priority scheme implemented in the (multicast) forwarding path of routers. In this scheme, the network supports two priority levels. As in RLM, the sender encodes video into multiple layers, and transmits them on different multicast groups. A receiver decides which groups (layers) to subscribe to (as it does under RLM), but *in addition it has the option of subscribing to a layer at high priority or low priority*. When congestion occurs, the router attached to the congested link drops packets associated with group(s) mapped as low priority in preference to those of groups(s) mapped as high priority. We refer to this scheme as Receiver-Selectable Loss Priorities (RSLP) since it is the receivers rather than the sender that determines the loss priority of packets (this is in contrast to priority dropping where the sender sets the drop priorities).

RSLP is easier to implement than per layer drop priority schemes, and does not require that all sessions use the same layering scheme. The drop priority can be represented at a router using a single additional boolean (one bit) variable for each outgoing link associated with a multicast group. This can be maintained as part of the multicast forwarding table that is already present at a router. The RSLP protocol is robust to imperfect priority mechanisms, thus allowing simpler implementation at routers. In Section V we describe a simple threshold based FIFO queue implementation, and show in Section VII that it provides sufficient loss differentiation to obtain the desired protocol benefits.

Receiver-driven Layered Multicast with Priorities: To exploit the loss differentiation provided by RSLP we design a new receiver-driven protocol called Receiver-driven Layered Multicast with Priorities (RLMP). The basic idea is to use RSLP's low priority channel for the least important layer, thereby protecting the base layers (transmitted on RSLP's high priority channel) from loss during transient congestion. In addition, RLMP uses an RLM-like approach to find the optimal number of layers that should be received at high priority. The algorithm uses the measured loss rate on its low priority layer(s) to decide when to add a new layer. The *passive measurements* avoid both the delays caused by the timer-based join experiments of RLM and perturbation of network conditions caused by join experiments.

An important feature of RLMP is that receivers retain the incentives present in RLM to prune back groups that exceed their bottleneck capacity. If a receiver maps all groups as high priority, it will experience losses across all layers including the base layers. To avoid this, a receiver maps as high priority only those groups that can fit in its bottleneck link without causing losses during bursts. In addition, it maps at most one additional layer (group) as low priority to "monitor" network congestion without compromising its base layers and to get whatever it can when the source is less bursty. The use of a low priority "buffer layer" enhances stability for VBR sources since short term burstiness only affects this layer and therefore the high priority layers experience fewer losses. Thus the scheme is effective in controlling congestion at packet time scales.

V. Network support for RSLP

This section presents the network mechanisms needed to implement RSLP. The objective is to use a receiver-selectable two-priority architecture to approximate a full-priority dropping architecture with much less complexity than a full-priority architecture.

Unlike a full-priority scheme where the flow priorities are set by the sender and do not change as the packet traverses the network, our proposed two-priority algorithm allows packet priorities to change at multicast branch points. In addition, the priorities at the branch-points are determined by the downstream receivers rather than by the sender.

RSLP requires that routers implement a two-priority dropping scheme and a multicast join/leave protocol that signals the priority at which a receiver wishes to subscribe to a multicast group. The next section describes the requirements in terms of router data structures and processing. Section V-B presents the protocol used by receivers to signal their desired subscription levels.

A. Router Data Structures and Processing

Our objective is to minimize the router overhead needed to implement RSLP. In particular, we want to keep the fastpath processing as efficient as possible. Although more complex implementations are possible, we have found our simple implementation provides sufficient drop differentiation for RLMP.

Each router maintains a single FIFO queue per outgoing link. The router reserves part of the queue, represented by a threshold τ, for high priority packets. The remainder of the queue can be used for high or low priority packets. When the queue usage is less than the threshold τ, all packets are added to the queue, regardless of their priority. If the queue usage exceeds τ, low priority packets are discarded on arrival. Only high-priority packets are allowed to enter the queue. When the queue usage reaches its capacity, all packets are discarded upon arrival. The complete discard algorithm *IsRoom* is shown in Figure 3. If *IsRoom()* returns FALSE, the packet is dropped, otherwise the packet is enqueued.

Packets are enqueued at the head of the queue, and dequeued for transmission from the tail of the queue. Therefore the queue is a strict FIFO queue. The advantage to this implementation of the hi-low priority queue is that we never take a packet off the queue once is has been placed on the queue. As a result, the queue can be implemented as a *hardware* FIFO queue in high-speed routers. It also has the nice characteristic that it is a single queue, so packets will be sent out of the queue in the

```
IsRoom(PacketPriority P) {
    if ( CurUsage < Threshold ) or
            (( P == High) and (CurUsage < Q_max)) {
        CurUsage++;
        return TRUE;
    }
    return FALSE;
}
```

Fig. 3. Routine to determines whether there is room on the outgoing link's queue.

```
ForwardPacket(struct packet p) {
    for ol in (routerentry[p.S, p.G].out_list) {
        if ( IsRoom(ol.getPriority()) ) {
            cp = p.clone();
            cp.setPriority(ol.getPriority());
            enque cp on ol.Q;
        }
    }
    discard p;
}
```

Fig. 4. The router forwarding routine.

order they arrived. In a Diffserv like architecture with multiple queues having different forwarding priorities, it is possible that packets on low priority queues could be delayed for a significant amount of time. Our single queue approach does not have this problem. However, since we never remove a low priority packet in favor of a high priority packet, we do not see the same benefits as if using an implementation that steals buffers away from low priority packets and gives them to high priority packets. The reward is implementation simplicity.

We now turn to forwarding state maintained at routers. In the following we focus our discussion on non-shared multicast trees, since the modifications required in the case of shared multicast trees (such as those used by CBT and its variants) are simpler. Each router maintains an array of router entries where each entry in the array contains the following information:

$$(S, G, (OL_1, P_1), (OL_2, P_2), ..., (OL_n, P_n))$$

where

S is the source address,
G is the multicast destination address,
OL_i is a structure with pointers to the outgoing link's queue
P_i is a pointer to the current priority setting for this multicast group on this outgoing link.

We refer to the list of (OL_i, P_i) pairs as the *outgoing link list* or simply the *out_list*.

All packets destined for the same multicast group over an outgoing link have the same priority value which is the maximum priority of all the downstream routers/receivers. The maximum priority is established via the join/leave signaling protocol described in section V-B.

Figure 4 shows the forwarding algorithm which consults the routing table to decide the outgoing links and send priorities. When an incoming packet arrives, the outgoing links on which to send the packet are looked up in the forwarding table (which also tells the priority at which the packet should be sent on each link). For each outgoing link, we try to enqueue the packet on the outgoing link (i.e., call *IsRoom()*). If it can be enqueued, it is appended to the queue associated with the outgoing link.

When the packet is placed in the outgoing buffer, the TOS (Type Of Service) field in the header (e.g., the IP TOS header field) is set to either high priority or low priority so that the receiver of the packet can tell whether the packet was sent at high or low priority. Note this TOS field only represents the service the packet observed over the last hop.

B. Signaling Protocol

The signaling protocol is used to establish the multicast tree and to set the queuing priorities for packets at the multicast branch points. The signaling protocol we propose could be implemented as a minor extension to almost any multicast routing protocol. In particular, we need to extend existing protocols by adding a new field to join messages called the *priority* field.

When a receiver wants to join a multicast group, it issues a *join* operation specifying the *multicast group* and the *priority* at which it wishes to receive packets. Note that join requests can be sent at any time. In fact we assume that the signaling protocol will periodically resend join messages to refresh the control state in the router. When the router receives a join message from and existing child (at the same priority level), it simply refreshes its state. In addition to refresh messages, a receiver may decide to change its priority by sending a new join message for the same group but specifying a new priority (which may be higher or lower than the previous priority). Leave messages work as they would in the normal multicast routing protocol.

Upon receiving the first join request for a group, a router adds the requesting node and the requested priority (i.e., (OL, P)) to its routing table. It then forwards the join request, with the same join priority, to all its "parents" for the multicast group (i.e., sources that it knows about).[3] Subsequently, when the router receives another join request (at the same priority level) from a different node, it simply updates its out_list and need not forward the join message as it has already joined. Using this mechanism, the join-priority is forwarded toward the sender, establishing the desired priority dropping at each node in the network.

The above join processing essentially mimics that of existing join protocols. Where priority joins differ from conventional multicast joins is in a child's ability (receiver or router) to specify a priority level that is possibly different than that of other children. A child can also *rejoin* the group at a different priority level (higher or lower than before). When a router receives a high priority join request, it simply records the request if it is already joined at high priority. If it is not currently joined at high-priority, it records the request in the routing table and forwards the join request to ensure it receives the packets at high-priority. In other words, as soon as one child requests high-priority, the router must also request it of its parent. When the router receives

[3] For some multicast routing protocols such as DVMRP, the join priority cannot be forwarded until the parent becomes known (e.g., after the first multicast message is flooded to the group).

a low priority join request, it simply adds the child to the routing table if it is already joined at low priority. However, if the router is joined at high priority, it will check to see if there are any remaining high priority children. If no high priority children remain, the router formulates a low priority join message and forwards it to its parents. If high priority children remain, nothing needs to be done. Finally, note that if the outgoing link is a shared access link, the priority for the link is set to the highest priority requested.

VI. Receiver-driven Layered Multicast with Priorities

To take advantage of the RSLP architecture, the layered multicast protocol must assign priorities to layers (i.e., *high* or *low* priority). Unlike priority-based layered approaches where the sender assigns priorities to each layer and every receiver subscribes to all layers, RSLP receivers set the priorities and must judiciously assign the priorities to avoid over-subscribing the high-priority channel. Consequently, the receiver's objective is to find the optimal number of layers that should be received at high and low priority.

The basic idea behind RLMP is to subscribe to N layers, where N is determined from the current high-priority and low-priority loss rates. At any point in time the lowest layers, layers 1 to $N-1$, will be subscribed to at high-priority, while the highest layer, layer N, will be subscribed to at low priority. To ensure that RLMP always subscribes to exactly one layer at low-priority and all others at high-priority, *adding* a layer involves:
1. joining layer $N+1$ at low priority, and
2. re-joining layer N at high priority

while *dropping* a layer involves:
1. leaving layer N, and
2. re-joining layer $N-1$ at low priority

Subscribing to the N^{th} layer at low priority serves two purposes (in addition to delivering bits to the application):
1. When packet bursts (congestion) occurs, the majority (if not all) packet loss will be concentrated at the highest (and thus least important) layer. In RLM, packet loss is equally distributed across all layers.
2. The low-priority layer provides a non-intrusive way to monitor the unreserved bandwidth, eliminating the need for disruptive join-experiments.

As a result, the loss rate on the low-priority layer can be used to accurately determine when sufficient bandwidth is available to change a layer to high-priority. One can think of subscribing to a low priority layer as a bandwidth reservation request. If the loss rate is sufficiently low after subscribing, the receiver assumes the "reservation" was granted and confirms the reservation by moving the layer to high-priority. The low-priority layer effectively eliminates the guess-work of RLM's join-"experiments". At the same time, the loss rate of the high-priority layers can be used to determine whether there is adequate bandwidth for the current $N-1$ layers.

The first key to RLMP's add/drop algorithm is the use of two loss-rate thresholds. The first threshold, H_{drop}, is similar in purpose to the drop threshold in RLM. It determines when the available bandwidth is insufficient to carry the current layers. When the high-priority channel loss rate exceeds H_{drop}, the low-priority layer is dropped and the highest high-priority layer changed to low-priority to reduce the session's bandwidth consumption. The second threshold, L_{add}, determines when the low-priority loss-rate has become so low that adding a new layer is warranted. Low (or no) loss on the low-priority layer indicates that sufficient bandwidth is now available to carry the low-priority layer with minimal loss (i.e., the "reservation" request succeeded). Because the low priority channel is primarily intended for layers that cannot be carried in their entirety, the current low-priority layer is changed to high-priority because it can be carried in its entirety and a new layer is added at low priority.

The second key to RLMP's add/drop mechanism is the way the loss rates are computed before being compared to the H_{drop} and L_{add} thresholds. Since RLMP's objective is to protect low-layers from short-term burst losses, we use relatively long-term average loss rates to trigger layer changes. We achieve this by computing the long-term loss rate using an exponentially weighted average, with smoothing parameter α:

$$S_{n+1} = \alpha S_n + (1-\alpha) X_{n+1}$$

where X_n is the n^{th} loss-rate observation, and S_n is the estimated long-term loss-rate. The parameter α controls how quickly the estimated loss rate reacts to changes in X_n. Larger values of α provide better stability. Note that losses are observed and maintained separately for the low priority channel and the high priority channel, and use different smoothing factors.

The advantage of a longer-term loss rate estimator is that it is less likely to react to sudden changes in loss rates, but instead looks for longer-term, significant and sustained changes in network load. On the other hand, it has the potential of creating a system that is slow to react. In a conventional (non-priority) network, slow reaction times can be catastrophic because sustained congestion causes loss across all layers, severely degrading the reception of the base layers. However, in RSLP congestion is absorbed by the low-priority layers, protecting the high-priority layers from loss and ensure a consistent base-level quality.

A long-term loss rate estimator has several advantages:
1. It is not susceptible to minor errors in the loss rate measurement.
2. The loss rate estimator curve (i.e., S_n plotted over time) changes slowly, causing it to slowly approach the threshold rather than jumping back and forth across the threshold (as the instantaneous loss rate X_n would do).
3. All receivers are more likely to see the same loss rate curve.

The fact that receivers "share loss rate knowledge" (point 3 above) can be used to improve fairness between competing sessions. In particular, if we set L_{add} using a graduated scale based on the current subscription level (or more generally on the current receive bit rate), we can ensure that sessions with few layers add a new layer before sessions with many layers. To ensure this, we use the following function to set the L_{add} threshold (which can, alternatively, be specified using received bit rates instead of layers):

$$L_{add} = 1 - (l\Delta + \rho_{min})$$

where l is the current subscription level, Δ is the desired separation between L_{add} levels, and ρ_{min} is the minimum through-

put that must be present in order to add a layer. In our tests, we set $\Delta = .09$ and $\rho_{min} = .52$. Note that $l \geq 2$ resulting in $L_{add} \geq 70\%$. When new bandwidth becomes available, receivers with few layers cross their L_{add} threshold before receivers with many layers, ultimately bringing them into a fair state.

VII. Evaluation

We simulated the performance of both RLMP and RLM using the NS simulator [Tea]. Our evaluation is based on the RLM code used in [MJV96] and the simulation model described in [BBS98]. The simulation topology used in our experiments is shown in Figure 5. In each experiment there are n concurrent sessions, each having a single source and a single receiver. Each source sends out on a dedicated 10 Mb/s link to the router at the head of the bottleneck link. The receivers are immediately downstream of the bottleneck link. The capacity of the bottleneck link is $S * 500$ Kb/s, where S is the number of concurrent sessions. For RLM experiments, routers use FIFO scheduling with a tail drop policy. For RLMP experiments, routers use RSLP's two-priority drop policy. Each source transmits a layered video session consisting of 6 layers. The base layer sent at a rate of 32 Kb/s, with the rate doubling with each subsequent layer. Thus the total bandwidth of 4 layers was 480 Kb/s, sufficient to accommodate 4 layers per source if allocated fairly.

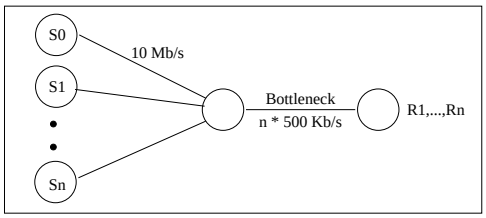

Fig. 5. Simulation Topology

We used the CBR/VBR source model described in [BBS98]. In it, the base layer generates traffic over 1 second intervals. In each interval n packets are transmitted, where n is chosen independently from the following random distribution: $n = 1$ with probability $1 - 1/P$, and $n = PA + 1 - P$ with probability $1/P$. A is the average number of packets per interval and is chosen to be four 1KB packets in our experiments. The n packets are transmitted in a single burst, starting at a random time (uniformly distributed) within the interval. For each higher layer l the interval is broken into 2^l subintervals, and n packets are sent in one burst at a random time in each of these subintervals correlated across layers. P represents the burstiness of the traffic source and gives an indication of the peak-to-mean ratio. For $P = 1$ the above model produces CBR traffic. We also report results for $P = 3$ and $P = 5$. Others have shown that peak-to-mean ratios of 2 to 10 are common for VBR traffic [RT99].

In all our tests, we used an α_{loprio} of 0.9 and an α_{hiprio} of 0.8, although we found that the results are not very sensitive to these settings as long as they tended toward long-term averages. For both RLM and RLMP we used a drop threshold loss rate of 25%. RLMP's add threshold loss rate was based on the current subscription level as described in VI. We measured the loss rate using a 250 ms interval, but have used smaller and larger intervals and found that specific value does not significantly affect the results. In all experiments, routers were configured with 60 packet buffers of which 60% was reserved for the high-priority traffic (i.e., base-layers).

A. Stability and Fairness

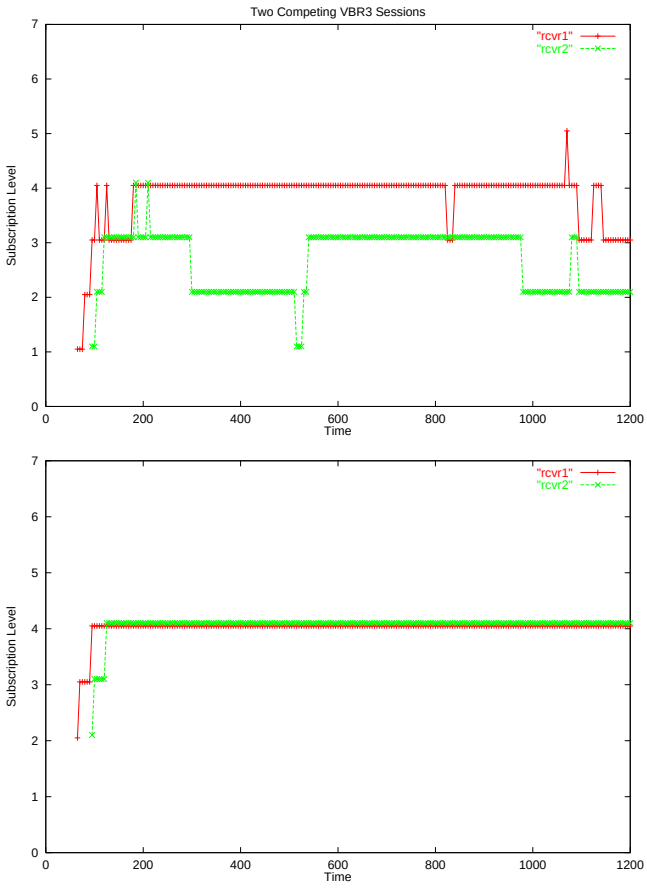

Fig. 6. RLM (top) vs. RLMP (bottom) stability for two VBR-3 sessions.

The graphs presented in this section show a single example run to illustrate the service quality as viewed by a receiver over time. Although behavior varies from run to run, the graphs shown represent a "typical" run. We performed many simulations to verify that the examples shown here are representative. Our simulation system is publicly available (at www.research.att.com/ gisli/iptv) so the interested reader can verify our results.

Figure 6 and Figure 8 show the subscription levels of competing VBR sessions using RLM and our new RLMP. Figure 6 plots the subscription level of two competing VBR-3; sessions using RLMP and RLM. Figure 8 the subscription level of 16 competing VBR-5 sessions. To make the graphs more readable, each session's subscription level is plotted with a small offset; so any value between N and $N + 1$ refers to a subscription level of N.

RLMP's stability is evident from these graphs. After the initial ramp-up, the RLMP sessions lock-in on a subscription level

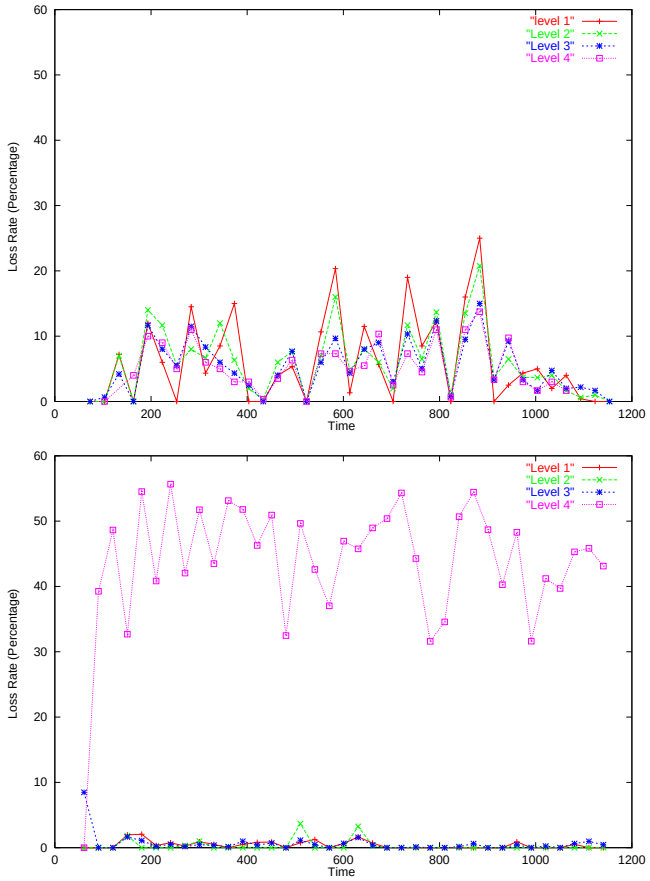

Fig. 7. RLM (top) vs. RLMP (bottom) loss rates per layer for session 1 in Figure 6 (plotted using averages over 30 second intervals).

and exploit the low priority of the highest layer to "ride-out" the transient bursts. This is in stark contrast to RLM. Using RLM, the variability of the traffic causes RLM to change the subscription level, resulting in persistent pattern of fluctuating level changes. The effectiveness of RLMP to successfully protect the lower layers from loss, and concentrate the losses on the low priority layer is illustrated in Figure 7. While effective, the imperfectness of our priority scheme results in some losses at lower layers, leaving a disincentive for the sessions to become too greedy and simply subscribe to all layers of the session.

RLMP achieves fair allocation among the competing sessions. In Figure 6 the two sessions converge to the equal allocation of four layers each. Figure 8 depicts 16 VBR-5 sessions competing for bandwidth. Out of the 16 sessions, 14 subscribe to four layers, while two sessions subscribe to five layers. Although ideally all of the sessions would subscribe to exactly the same number of layers, subscription level differences of one (due to roundoffs to integral number of layers) is acceptable and unavoidable. We have run similar tests at other VBR settings using 2, 4, 8, and 16 competing sessions and observed similar results to those shown here.

The shared knowledge approach to obtain fairness works well as long as bandwidth continues to become available. Assuming sessions are constantly coming and going, the approach works well with both CBR and VBR traffic. In addition, all our tests showed that the "idle" periods in VBR traffic are sufficient to cause receivers with few layers to reach their add threshold and bring the system into a reasonably fair allocation even if sessions are not coming and going. CBR traffic behaves differently. If the sessions start at roughly the same time, they end up sharing the bandwidth fairly (which was also true of RLM), and if sessions come and go over time, CBR also reaches a fair allocation. However, in certain pathological cases, it is possible for one CBR session to prevent another from achieving its fair share. We are currently investigating mechanisms similar to those used in [WSS97] to handle this case.

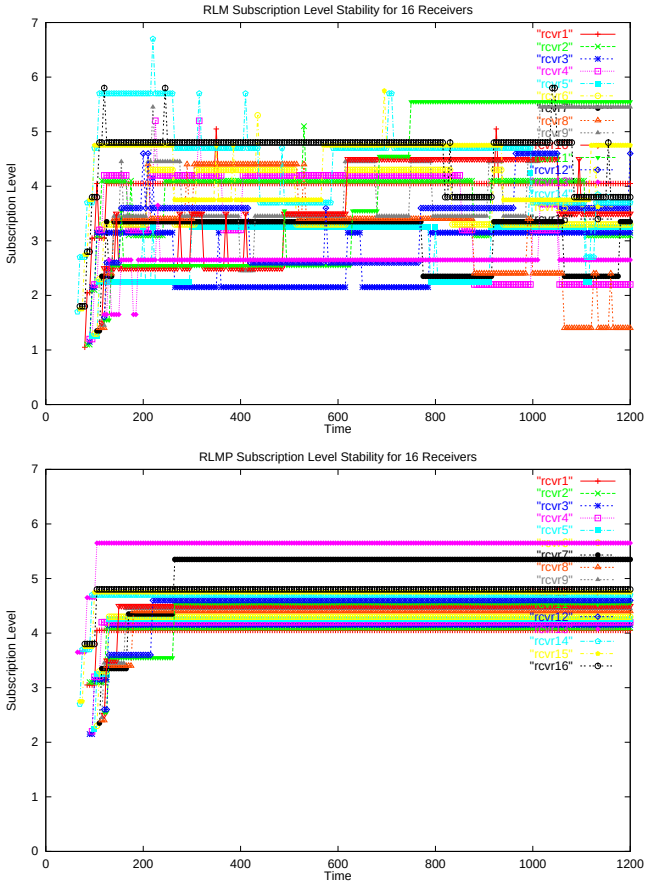

Fig. 8. RLM (top) vs. RLMP (bottom) fairness/stability for 16 competing VBR-5 sessions.

B. *Protocol complexity*

The complexity of our new protocol is low. At receivers the additional space requirement consists only of a second set of statistics, one set for each loss priority. Specifically, our *ns* implementation stores two double variables, one for the exponentially weighted loss average and another one for the current loss measurement. In comparison to the other state information associated with each layer, this additional state is negligible. The processing time at the receiver compares favorably with RLM.

At branch points in the multicast tree, routers must record an additional boolean variable (i.e., a bit to record high or low drop priority) with each output port for each group. The processing complexity at routers is increased in two (minor) ways. First, during the lookup-operation for multicast packets, the drop priority must also be retrieved from the routing table entry. Second,

before enqueuing the the datagram on the outgoing queue, the router must verify that the queue occupancy has not exceeded the low priority admit level. Although this overhead is in the router's fast-path, this complexity is less than or comparable to what is being discussed in the differentiated services working group at the IETF, and has negligible impact on forwarding performance. While our protocol would benefit from a better priority implementation, it is explicitly designed to be robust to coarse implementation of priorities.

VIII. Related Work

In addition to the work on RLM, there are several other protocols for layered video transmission. ThinStreams[WSS97] addresses issues of fairness for multiple sessions consisting of "thin" equal-sized layers. The algorithm requires that receivers calculate join and leave thresholds based on their current level of subscription. The idea is that receivers subscribing to a larger number of layers will surrender them more quickly than a receiver with a smaller subscription set. Instability is not addressed, and may be significant given the join/leave overheads of thin layers.

The issue of fairness between RLM and TCP traffic has also been studied [VRC98], [TPB97]. The idea in [VRC98] is for receivers to use a join/leave strategy for congestion control which mimics the behavior of TCP. This relies on making appropriate choices of layer bandwidths and the time delay between trying to increase subscription. In [TPB97], each receiver tries to determine the share of bandwidth that an equivalent TCP connection would use, and then makes join/leave decisions in order to match that value for the multicast session. TCP's objectives differ significantly from the objectives one would design for a video transmission protocol. Consequently, making the layered multicast scheme behave like TCP can help it "get along", but may wreak havoc on the "visual experience". The focus in both studies is to provide fairness between all TCP and all RLM traffic, rather than between individual RLM sessions. Although these solutions interact better with TCP, they will experience the same saw-tooth behavior experienced by TCP flows. In fact, the use of multiplicative decrease will result in more sudden and drastic level changes than RLM. Also, because these protocols behave like TCP, they will offer "fairness" similar to that of TCP, which is known to produce arbitrary unfairness. Other enhancements over RLM such as synchronized joins between receivers in the same session may offer substantial improvements over RLM. However, since our study used a single receiver per session, these enhancements will not improve on the RLM stability and fairness results presented here.

Recently, [RKT99] and [JZA99] address the issue of fairness between receivers of a single layered multicast session. [RKT99] identifies a set of fairness properties and shows that they can be achieved through sender coordination of joins. [JZA99] proposes the use two multicast groups, a low-bitrate base group, and an additional variable bitrate group. All receivers are expected to be capable of receiving the base group with minimal losses. The bitrate of the variable group is set by the source in response to messages from the receivers reporting losses. Unlike these proposals, our algorithm does not require any receiver-sender interaction, and thus has more desirable properties in terms of scaling and responsiveness.

IX. Conclusions

We have presented the Receiver-Selectable Loss Priorities mechanism that provides simple loss differentiation, and the associated Receiver-driven Layered Multicast with Priorities algorithm for supporting layered multicast. The main motivation was to address the lack of stability and fairness observed in layered multicast protocols such as RLM. The rationale behind RSLP is to obtain the benefits of priority dropping while using a low complexity, coarse grain loss priority mechanism. In addition, RSLP retains the incentives of the receiver driven multicast model to subscribe to only those layers that can be effectively delivered to receivers. RLMP utilizes the RSLP mechanism to drive the adding and dropping of layers to achieve stability and fairness. It uses the loss rates measured on the high and low priority layers to determine its optimal subscription level. Our simulation results show that RLMP protects the base layers during short term congestion thus providing stability. It also provides fairness across competing sessions.

References

[BBS98] S. Bajaj, L. Breslau, and S. Shenker. Uniform versus priority dropping for layered video. In *Proceedings of the SIGCOMM '98 Conference*, pages 131–143, September 1998.

[GGHS99] R. Gopalakrishnan, J. Griffioen, G. Hjalmtysson, and C. Sreenan. Stability and Fairness Issues in Layered Multicast. In *Proceedings of the NOSSDAV '99*, June 1999.

[JZA99] T. Jiang, E. W. Zegura, and M. Ammar. Inter-receiver fair multicast communication over the internet. In *Proceedings of the 9th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 103–114, June 1999.

[LPA98] X. Li, S. Paul, and M. Ammar. Layered Video Multicast with Retansmissions (LVRM): Evlauation of Hierarchical Rate Control. In *Proceedings of the INFOCOMM '98 Conference*, 1998.

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

[RKT99] D. Rubenstein, J. Kurose, and D. Towsley. The impact of multicast layering on network fairness. In *To appear in Proceedings of ACM SIGCOMM*, September 1999.

[RT99] Jennifer Rexford and Don Towsley. Smoothing variable-bit-rate video in an internetwork. *IEEE/ACM Transactions on Networking*, pages 202–215, April 1999.

[Tea] The MASH Research Team. The ns network simulator. `http://www-mash.cs.berkeley.edu/ns`.

[TPB97] T. Turletti, S. F. Parisis, and J-C. Bolot. Experiments with a layered transmission scheme over the Internet. Technical Report 3296, INRIA Sophia Antipolis, France, November 1997.

[VRC98] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like congestion control for layered multicast data transfer. In *Proceedings of the INFOCOM '98 Conference*, pages 996–1003, March 1998.

[WSS97] L. Wu, R. Sharma, and B. Smith. Thinstreams: An Architecture for Multicast Layered Video. In *Proceedings of NOSSDAV '97*, 1997.