Receiver-driven Layered Multicast

Steven McCanne
University of California, Berkeley and
Lawrence Berkeley National Laboratory
mccanne@ee.lbl.gov

Van Jacobson
Network Research Group
Lawrence Berkeley National Laboratory
van@ee.lbl.gov

Martin Vetterli
University of California, Berkeley
martin@eecs.berkeley.edu

Abstract

State of the art, real-time, rate-adaptive, multimedia applications adjust their transmission rate to match the available network capacity. Unfortunately, this source-based rate-adaptation performs poorly in a heterogeneous multicast environment because there is no single target rate — the conflicting bandwidth requirements of all receivers cannot be simultaneously satisfied with one transmission rate. If the burden of rate-adaption is moved from the source to the receivers, heterogeneity is accommodated. One approach to receiver-driven adaptation is to combine a layered source coding algorithm with a layered transmission system. By selectively forwarding subsets of layers at constrained network links, each user receives the best quality signal that the network can deliver. We and others have proposed that selective-forwarding be carried out using multiple IP-Multicast groups where each receiver specifies its level of subscription by joining a subset of the groups. In this paper, we extend the multiple group framework with a rate-adaptation protocol called Receiver-driven Layered Multicast, or RLM. Under RLM, multicast receivers adapt to both the static heterogeneity of link bandwidths as well as dynamic variations in network capacity (i.e., congestion). We describe the RLM protocol and evaluate its performance with a preliminary simulation study that characterizes user-perceived quality by assessing loss rates over multiple time scales. For the configurations we simulated, RLM results in good throughput with transient short-term loss rates on the order of a few percent and long-term loss rates on the order of one percent. Finally, we discuss our implementation of a software-based Internet video codec and its integration with RLM.

ACM SIGCOMM '96, August 1996, Stanford, CA.

1 Introduction

The Internet's heterogeneity and scale make multipoint communication design a difficult problem. For real-time multimedia, we would like to "broadcast" a live signal from any particular sender to an arbitrarily large set of receivers along paths with potentially high variability in bandwidth. The simplest solution to this problem is to distribute a uniform representation of the signal to all interested receivers using IP Multicast [8]. Unfortunately, this is suboptimal — low-capacity regions of the network suffer congestion while high-capacity regions are underutilized.

The problems posed by heterogeneity are not just theoretical, they impact our daily use of Internet remote-conferencing. For example, each week for the past year, U.C. Berkeley has broadcast a seminar over their campus network and onto the Internet. As depicted in Figure 1, a video application is run on a "seminar host" that sources a single-rate signal at 128 kb/s, the nominal rate for video over the Internet Multicast Backbone, or MBone [11]. However, a number of users on the local campus network have high bandwidth connectivity and would prefer to receive higher-rate, higher-quality video. At the other bandwidth extreme, many users have ISDN access and would like to participate from home, but a 128 kb/s video stream overwhelms an ISDN line.

In this open-loop approach, the sender broadcasts at some fixed rate without regard to changing network conditions. A better approach is to adjust the transmission rate to match the available capacity in the network, i.e., to react to congestion. Pioneering research in rate-adaptive video [1, 19, 23] has shown that this is feasible, but unfortunately, in the context of multicast, the notion of network capacity is ill defined. A control scheme that adjusts the rate of a single stream at the source simply cannot meet the conflicting requirements of a set of heterogeneous receivers.

An alternative approach is to combine a layered compression algorithm with a layered transmission scheme [29, 32]. In this approach, a signal is encoded into a number of layers that can be incrementally combined to provide progres-

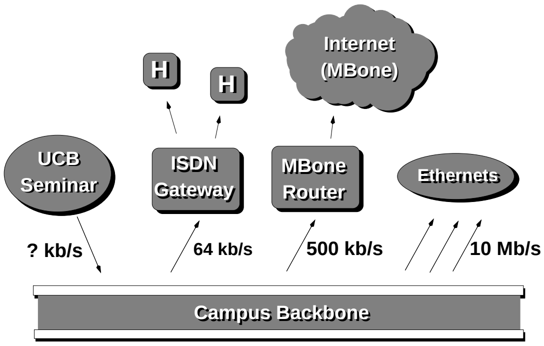

Figure 1: Network heterogeneity

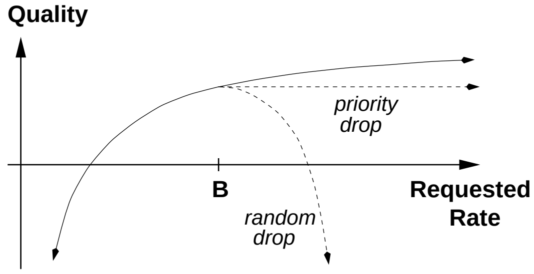

Figure 2: Priority-/Random-drop Tradeoff.

sive refinement. By dropping layers at choke points in the network — i.e., selectively forwarding only the number of layers that any given link can manage — heterogeneity is managed by locally degrading the quality of the transmitted signal.

This framework provides an elegant solution to heterogeneity but a crucial piece is missing. The system must have mechanisms for determining, communicating, and executing the selective forwarding of subflows along all the links in a distribution. While much of the previous work leaves this as an implementation detail, a novel mechanism based on IP Multicast was suggested by Deering [6] and elaborated on and/or independently reported in [4, 9, 20, 26, 33]. In this approach, the different layers of the hierarchical signal are striped across multiple multicast groups and receivers adapt to congestion by adding and dropping layers (i.e., joining and leaving multicast groups). Receivers implicitly define the multicast distribution trees simply by expressing their interest in receiving flows. Thus there is no explicit signaling between the receivers and routers or between the receivers and source.

While this general mechanism has been discussed in the research community, the problem has not been studied in detail, algorithms for adaptation have not been developed, and systems based on these ideas have not yet emerged. This paper addresses some of the open questions related to layered multicast transport through the design and simulation of an experimental network protocol called Receiver-driven Layered Multicast or RLM. In the following section we describe the network model assumed by RLM. Next we provide intuition for RLM and present the protocol in detail. We then explore its performance through simulation. Finally, we discuss the integration of RLM into a comprehensive systems framework, report on related work, and describe our future work.

2 The Network Model

RLM works within the existing IP model and requires no new machinery in the network. We assume:

- only best-effort, multipoint packet delivery, e.g., without guarantees for packet ordering, minimum bandwidth, etc.;
- the delivery efficiency of IP Multicast, i.e., that traffic flows only along links with downstream recipients; and,
- *group-oriented* communication: senders need not know that receivers exist and receivers can dynamically join and leave the communication group in an efficient and timely manner.

These three requirements are sufficient for single source distribution to arbitrary numbers of receivers under RLM. To handle multiple, simultaneous sources, RLM assumes that receivers can specify their group membership on a per-source basis (i.e., a receiver can ask for packets sent to some group but exclude packets from one or more sources)[1].

We refer to a set of end-systems communicating via a common set of layered multicast groups as a *session*. Because the IP Multicast service model does not export any of the routing mechanism, we cannot guarantee that all the groups of a single session follow the same distribution tree. That is, multicast routing can be carried out on a per-group basis and different groups can be routed along different spanning trees. Although RLM is most easily conceptualized in a network where all the groups follow the same route, this is not a requirement.

The relationship among the information contained across the set of groups in a session can either be cumulative or independent. In the cumulative case, each layer provides refinement information to the previous layers and the receiver must subscribe to all groups up to and including the highest group. In the independent case, each layer is independent and the receiver need only subscribe to one group. This latter scheme is often called *simulcast* because the source transmits

[1] Source-based pruning is not part of the current IP Multicast specification but is included in the next version, IGMP-3, which is under review by the IETF.

multiples copies of the same signal simultaneously at different rates (resulting in different qualities). In this paper, we focus on the cumulative model because it makes more effective use of bandwidth but RLM is also compatible with the simulcast model.

Instead of the best-effort, IP Multicast model described above, the universally cited approach to layered packet transmission adds a drop-preference packet discard policy to all the routers in the network. Under drop-preference, when congestion occurs, routers discard less important information (i.e., low-priority packets) before more important information (i.e., high-priority packets). Although this approach provides graceful degradation in the presence of packet loss, we believe it has scaling problems because it rewards poorly-behaved users.

This effect is illustrated in Figure 2, which plots the quality of a received signal vs. the requested bit rate for both priority-drop and random-drop policies. In both cases, the quality of the received signal increases with the requested rate up to the bottleneck capacity B but beyond this, the quality depends on the drop policy. With random-drop, quality degrades because packets are dropped uniformly across all layers, while with priority-drop the quality remains constant because only "enhancement" packets are dropped. The key distinguishing feature of these two curves is their convexity. Because the random-drop curve is strictly convex, it has a unique maximum. Thus we can design a control system that maximizes the quality metric and drives the system toward the stable, uncongested bottleneck rate B. The priority-drop curve has no unique maximum and hence does not admit a control system that optimizes delivered quality by converging to a single, stable operating point. In fact, a greedy or naive user would likely request a rate far above the bottleneck rate B, driving the network into a persistently congested state.

3 The RLM Protocol

Building on the best-effort IP-Multicast network model, we now describe RLM at a high-level to develop intuition for the protocol before discussing the low-level details. To first order, the source takes no active role in the protocol. It simply transmits each layer of its signal on a separate multicast group. The key protocol machinery is run at each receiver, where adaptation is carried out by joining and leaving groups. Conceptually, each receiver runs the following simple control loop:

- on congestion, drop a layer;
- on spare capacity, add a layer.

Under this scheme, a receiver searches for the optimal *level of subscription* much as a TCP source searches for the bottleneck transmission rate with the slow-start congestion avoidance algorithm [21]. The receiver adds layers until congestion occurs and backs off to an operating point below this bottleneck.

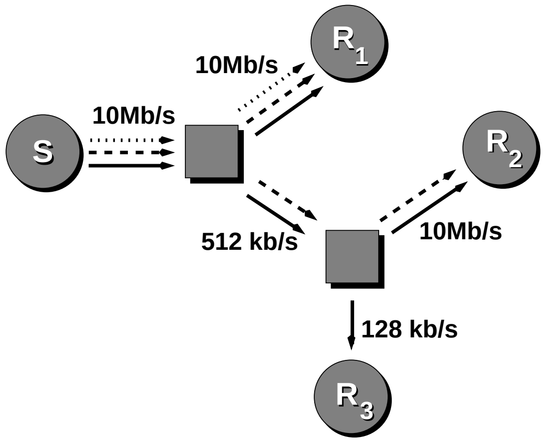

Figure 3: End-to-end adaptation.

Figure 3 illustrates the RLM scheme. Suppose source S is transmitting three layers of video to receivers R_1, R_2, and R_3. Because the S/R_1 path has high capacity, R_1 can successfully subscribe to all three layers and receive the highest quality signal. However, if either R_2 or R_3 try to subscribe to the third layer, the 512 kb/s link becomes congested and packets will be dropped. Both receivers react to this congestion by dropping layer three, prompting the network to prune the unwanted layer from the 512 kb/s link. Finally, because of the limited capacity of the 128 kb/s link, R_3 might have to drop back all the way to a single layer. The effect is that the distribution trees for each layer have been implicitly defined as a side effect of the receiver adaptation.

3.1 Capacity Inference

To drive the adaptation, a receiver must determine if its current level of subscription is too high or low. By definition, the subscription is too high if it causes congestion. This is easy to detect because congestion is expressed explicitly in the data stream through lost packets and degraded quality. On the other hand, when the subscription is too low, there is no equivalent signal — the system continues to operate at its current level of performance. We must rely on some other mechanism to provide this feedback.

One source for this feedback might be to monitor link utilization and explicitly notify end-systems when capacity becomes available. However, this requires new mechanism in the network that renders deployment difficult. The approach we adopt in RLM is to carry out active experiments by spontaneously adding layers at "well chosen" times. We call this spontaneous subscription to the next layer in the hierarchy a *join-experiment*. If a join-experiment causes congestion, the receiver quickly drops the offending layer. If a join-experiment is successful (i.e., no congestion occurs), then

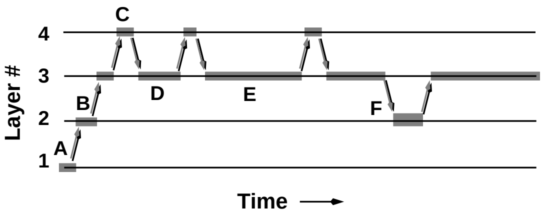

Figure 4: An RLM "sample path"

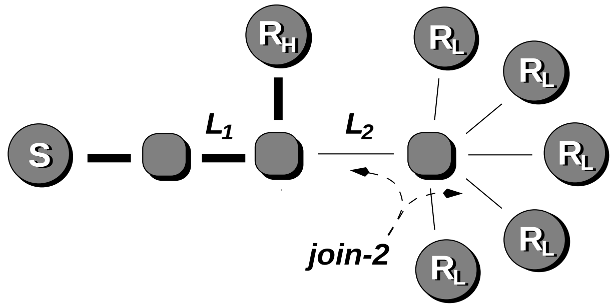

Figure 5: Shared Learning

the receiver is one step closer to the optimal operating point.

3.2 RLM Adaptation

Unfortunately, join-experiments cause transient congestion that can impact the quality of the delivered signal. Therefore, we need to minimize the frequency and duration of join-experiments without impacting the algorithm's convergence rate or its ability to track changing network conditions. This is done through a learning algorithm, where over time, each receiver determines the level of subscription that causes congestion. By doing join-experiments infrequently when they are likely to fail, but readily when they are likely to succeed, we reduce the impact of the experiments. We implement this learning strategy by managing a separate *join-timer* for each level of subscription and applying exponential backoff to problematic layers.

Figure 4 illustrates the exponential backoff strategy from the perspective of a single host receiving up to four layers. Initially, the receiver subscribes to layer 1 and sets a join-timer (A). At this point, the timer duration is short because the layer has not yet proven problematic. Once the join-timer expires, the receiver subscribes to layer 2 and sets another join-timer (B). Again, the timer is short and layer 3 is soon added. The process repeats to layer 4, but at this point, we will assume congestion occurs (C). A queue will then build up and cause packet loss. Once the receiver detects these lost packets, it drops back to layer 3. The layer 3 join-timer is then multiplicatively increased and another timeout is scheduled (D). Again, the process repeats, congestion is encountered, and the join-timer is further increased (E). Later, unrelated transient congestion provokes the receiver to drop down to layer 2 (F). At this point, because the layer 3 join-timer is still short, the layer is quickly reinstated.

In order to properly correlate a join-experiment with its outcome, we must know how long it takes for a local layer change to be fully established in the network and for the resulting impact to be detected back at the receiver. We call this time interval the *detection-time*. If a join-experiment lasts longer than the detection-time without congestion occurring, then we deem the experiment successful. On the other hand, if congestion occurs within the detection-time interval, we assume the experiment failed and increase the join-timer for that layer. Because the detection-time is unknown and highly variable, we estimate it and its variance adaptively. We initialize our estimator (mean and deviation) with a conservative (i.e., large) value, and adapt it using failed join-experiments. That is, when an experiment fails, we update our estimator with the time interval between the start of the experiment and the onset of congestion.

3.3 Scaling RLM

If each receiver carries out the above adaptation algorithm independently, the system scales poorly. As the session membership grows, the aggregate frequency of join-experiments increases; hence, the fraction of time the network is congested due to join-experiments increases. Moreover, measurement noise increases because experiments tend to interfere with each other. For example, if one receiver is conducting an experiment on layer 2 and another begins an experiment on layer 4 that causes congestion, then the first receiver can misinterpret the congestion and mistakenly back off its layer 2 join-timer.

We can avoid these problems by scaling down the individual join-experiment rates in proportion to the overall group size. In other words, we can fix the aggregate join-experiment rate independent of session size much as RTCP scales back its control message rate in proportion to the group size [28]. However, reducing the experiment rate in this manner decreases the learning rate. For large groups, the algorithm will take too long to converge.

Our solution is "shared learning": Before a receiver conducts a join-experiment, it notifies the entire group by multicasting a message identifying the experimental layer. Thus all receivers can learn from other receivers' failed join-experiments. For example, Figure 5 shows a topology with a single source, one receiver R_H situated along a high-speed path (denoted by the thickened links) and a set receivers, each labeled R_L, situated at the far end of a low-rate link. Suppose a low-rate receiver decides to conduct a join-experiment on layer 2. It broadcasts a join-2 message to the group and joins the layer 2 multicast group. As a result, link L_2 becomes oversubscribed and congestion results, causing packets to be dropped indiscriminately across both layers.

At this point, all of the R_L receivers detect the congestion and since they know a layer 2 experiment is in progress, they all scale back their layer 2 join-timer. Thus all of the low-bandwidth receivers learn together that layer 2 is problematic. Each receiver need not run individual experiments to discover this on their own.

This learning process is conservative. Receivers make their decisions based on failed experiments not on successful experiments. Moreover, the success/failure decision is based on local observations, not on a global outcome. That is, each receiver decides whether the experiment succeeds based on the network conditions on the path from the source to that receiver, entirely independent of the receiver that instantiated the join-experiment. Hence, a given experiment may succeed for some receivers but fail for others.

Even though the shared learning process enhances the protocol's scalability by reducing convergence time, overlapped experiments can still adversely impact the learning rate. But because receivers explicitly announce the start of each experiment, the probability that an experiment overlaps with another can be substantially reduced by suppressing the start of a new experiment when one is outstanding. For example, if in Figure 5 receiver R_H decides to carry out a join-4 experiment that causes congestion on link L_1, then the low-rate receivers can misinterpret this as a failed join-2 experiment. But because R_H sees the explicit join-2 announcement, it will suppress the join-4 experiment and thereby limit the interference. Note that this exchange of information is merely an optimization. If the announcement packet is lost, the algorithm still works albeit with potentially reduced performance.

Because the shared learning process determines what does *not* work rather than what does work, each receiver can advance its level of subscription only through actual join-experiments. If the suppression algorithm were completely exclusionary, then the convergence time could still be very large because each receiver would have to wait its turn to run an experiment. Instead, we allow experimental overlap if the pending level is the same as or less than the level in progress. This gives newer receivers with lower levels of subscription an opportunity to conduct experiments in the presence of a large population of established receivers at higher levels of subscription. Although this mechanism allows experimental overlap, a receiver that causes an overlap can condition its response accordingly by reacting more conservatively than in the non-overlapped case. The intuition behind this scheme is that high-layer receivers allow low-layer receivers to quickly adapt to their stable level of subscription. As the low-layer receivers adapt, their join-experiment frequency falls off and the high-layer receivers will again find idle periods in which to conduct join-experiments.

This technique for sharing information relies on the fact that the network signals congestion by dropping packets across all layers of the distribution. Under a priority-drop policy, receivers not subscribed to the experimental layer would not see packet loss and would not know the experiment failed. In short, a priority-drop policy interferes with the scalability of RLM.

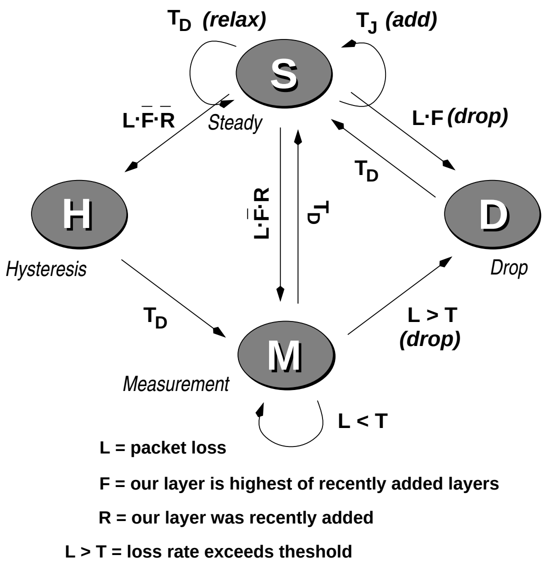

Figure 6: The receiver protocol state machine.

3.4 The RLM State Machine

Figure 6 elaborates the protocol sketched in the previous section. There are four states: steady-state (S), hysteresis state (H), measurement state (M), and drop state (D). Each state transition is labeled with the reason for the transition, either packet loss or a timeout. Actions associated with a transition are indicated in parentheses.

Join-timers (T_J) are randomized to avoid protocol synchronization effects [15], while detection-timers (T_D) are set to a scaled value of the detection-time estimator. The *add* action implies that we subscribe to the next layer in the multicast group hierarchy, while the *drop* action implies that we drop the current layer and multiplicatively increase the join-timer for that layer. The *relax* action implies that we multiplicatively decrease the join-timer for the current layer. There are two types of loss actions: a fast reaction to a single packet loss (indicated by L) and a slower reaction to a sustained loss rate. The loss rate is measured with a short-term estimator and action is taken if the estimator exceeds a configured threshold (indicated by $L > T$).

In the S state, there is always a pending join-timer (unless the receiver is subscribed to all available layers). When the join-timer expires, we broadcast an explicit notification message to the group and add a layer. Upon reception of the join-experiment message, a receiver notes the experiment start time for that layer. In this way, we track the join-experiment activity at each layer and deem an experiment "in progress" if the time since the experiment started is less than

$$k_1 \hat{T}_D + k_2 \hat{\sigma}_D$$

where $\hat{T}_D$ is the detection-time estimator, $\hat{\sigma}_D$ is the detection-time sample mean-deviation, and k_1 and k_2 are design constants. If a lower layer join-experiment is in progress, we ignore the current join-timer and simply schedule a new one.

When loss occurs in the S state, the resulting action depends on the presence of active join-experiments. If there is a join-experiment in progress and our level of subscription corresponds to the highest-level join-experiment in progress, we infer that our join-experiment has failed, drop the offending layer, back off the join-timer, and enter the D state. On the other hand, if we are locally conducting a join-experiment but a concurrent join-experiment is running at a higher layer, then it is likely that the higher layer experiment failed while ours did not but we cannot be certain. Hence, we enter the measurement state M to look for longer term congestion before dropping our layer. Finally, if we were not conducting a join-experiment at all, we transition to the H state.

The H state provides hysteresis to absorb transient congestion periods. This prevents a receiver in steady-state from reacting to join-experiments that are carried out by other receivers in the network or to transient network congestion. Once the detection-timer expires, we assume that any transient join-experiment is finished and transition to the measurement state and back to the S state after another detection time. If on other hand, the congestion is long-term (e.g., because of new offered load), then once we enter the M state, the loss rate estimator ramps up, exceeds the threshold, and forces the current layer to be dropped.

When a layer is dropped in response to congestion, the receiver enters the D state, sets the detection-timer, and ignores losses until the detection-timer expires. This prevents the receiver from (over-)reacting to losses that are unrelated to its current level of subscription. Once the receiver has waited long enough, the incoming packet stream will reflect the new level of subscription and the receiver can take action on the subsequent quality.

3.5 Protocol State Maintenance

In addition to the current state identifier, the receiver control algorithm must maintain the current subscription level, the detection-time estimator, and the join-timers. This state, along with several protocol design constants, is summarized in Table 1.

While the subscription level is trivial to maintain, the detection-time estimator and join-timers must be dynamically adapted to reflect changing network conditions. There are two operations performed on join-timers: backoff and relaxation. Call the mean of the join-timer for level-k, $\hat{T}_J^k$. Each timer interval is chosen randomly from a distribution parameterized by $\hat{T}_J^k$. When a join-experiment fails, the join-timer is multiplicatively increased:

$$\hat{T}_J^k \leftarrow \min(\alpha \hat{T}_J^k, T_J^{\max})$$

state	state identifier (S, H, M, D)
N	current level of subscription
$\hat{T}_J^k$	join-timer for level k
$\hat{T}_D$	detection-time sample mean
$\hat{\sigma}_D$	detection-time sample deviation
$T_J^{\min}$	minimum join-timer interval
$T_J^{\max}$	maximum join-timer interval
α	join-timer backoff constant
β	join-timer relaxation constant
k_1, k_2	detection-time estimator scaling term
g_1, g_2	detection-time estimator filter constants

Table 1: RLM State and Parameters

where $\alpha > 1$ is the backoff parameter and $T_J^{\max}$ is the maximum timeout. We clamp the backoff at a maximum to guarantee that a receiver will periodically probe for spare bandwidth. To scale to large session sizes, $T_J^{\max}$ is dynamically adjusted in proportion to the number of receivers. The number of receivers is in turn dynamically estimated through the exchange of session-wide control messages (e.g., as in RTCP [28]). Thus the aggregate join-experiment rate is fixed, independent of the session size, and packet loss induced by join-experiments does not increase with session size.

The join-timer undergoes relaxation in steady-state. The longer a receiver is in steady-state at some level, the more likely it is for that level to be stable. Thus the corresponding join-timer interval should be small. We adapt the join-timer by geometrically decreasing it at detection-timer intervals:

$$\hat{T}_J^k \leftarrow \max(\beta \hat{T}_J^k, T_J^{\min})$$

where $\beta < 1$ is the relaxation constant and $T_J^{\min}$ is the minimum join-timer interval.

While the join-timers are determined algorithmically, the detection-time estimate is derived directly from network measurements. The detection-time reflects the latency between time at which a local action is carried out and the time at which impact of that action is reflected back to the receiver. Note that this delay can be much larger than the time it takes for the network just to instantiate a new flow. If the new aggregate bandwidth exceeds the bottleneck link capacity by only a small amount, a long time may pass before a queue builds up and causes packet loss.

The detection-time estimate is computed by correlating failed join-experiment start times with the onset of congestion. Each time a join-experiment fails, the detection-time estimator is fed the new latency measurement. The measurement, D_i, is passed through first-order low-pass filters with gains g_1, g_2:

$$\begin{aligned} \hat{\sigma}_D &\leftarrow (1 - g_2)\hat{\sigma}_D + g_2 |D_i - \hat{T}_D| \\ \hat{T}_D &\leftarrow (1 - g_1)\hat{T}_D + g_1 D_i \end{aligned}$$

4 Simulations

In this section, we present simulation results of several simple network topologies to explore the scalability of RLM. This work is in an exploratory stage. Our simulations do not prove that RLM is definitively scalable. Rather, they demonstrate that the scaling behavior is consistent with our intuition and show that for simple scenarios, the protocol's performance is good. In a real network, performance will be affected by cross-traffic and competing groups, both of which add noise to the measurement process and introduce interactions that could result in oscillatory behavior. We will assess the impact of such interactions in future work.

We implemented the RLM protocol described above in the LBNL network simulator *ns* [24]. Not only did this implementation serve as a framework for evaluating the protocol's performance, but the simulator provided feedback that was critical to the design process. Ns is an event-driven packet-level simulator controlled and configured via Tcl [27]. Shortest-path routes are computed for the input topology and multicast packets are routed via reverse-path forwarding. A flooding algorithm similar to Dense-mode Protocol Independent Multicast (PIM) [7] handles forwarding and pruning of multicast flows.

Hierarchical sources are modeled as a set of constant-bit rate (CBR) streams with fixed packet sizes. Packets are generated at times defined by the following law:

$$\begin{aligned} T_0 &= 0 \\ T_k &= T_{k-1} + \Delta + N_k, \quad k > 0 \end{aligned}$$

where Δ is a fixed interval chosen to meet the target bit-rate and N_k is zero-mean noise process to model variable coding delays ($\{N_k\}$ is i.i.d. uniform on $[-\Delta/2, \Delta/2]$). Unfortunately, this simple model fails to capture the burstiness of real video streams [18]. Because convergence in RLM relies on matching the layered rates to available capacity, smooth sources are well-behaved and this traffic model is overly optimistic. On the other hand, a bursty source can be smoothed out by applying rate-control through adaptive quantization at the cost of variable quality. A topic for future research is whether RLM is amenable to bursty sources.

Before discussing the simulation results, we define the parameters we varied for the simulations and the metrics we used to evaluate the results. Variable parameters include network topology, link bandwidths and latencies, the number and rate of transmission layers, and the placement of senders and receivers. Fixed parameters include the routing discipline (drop-tail), the router queue size (20 packets), and the packet size (1 KB). In all of our simulations, the link bandwidths are 1.5 Mb/s, the traffic sources are modeled as a six-layer CBR stream at rates 32×2^m kb/s, $m = 0 \ldots 5$, and the start-time of each receiver is randomly chosen uniformly on the interval $[30, 120]$ seconds. The protocol constants from Table 1 have the following values: $\alpha = 2$, $\beta = 2/3$, $k_1 = 1$, $k_2 = 2$, $g_1 = 0.25$, $g_2 = 0.25$, $T_J^{\min} = 5$ sec, $T_J^{\max} = 600$ sec. Each join-timer interval is chosen from $\lambda/2 + X$, where X is a random variable with density

$$f_X(x) = \begin{cases} \lambda e^{-\lambda x}/(1 - e^{-4\lambda^2}) & 0 \le x \le 4\lambda \\ 0 & \text{otherwise} \end{cases}$$

and $\lambda = \hat{T}_J^k$. These protocol parameters were chosen heuristically based on experimentation with and intuition about the protocol. In future work, we plan to present a larger range of configurations and a study of the parametric sensitivity.

4.1 Evaluation Metrics

In our layered multicast transmission scheme, a traditional metric like aggregate throughput is not well defined because each user might receive a different bandwidth and experience different loss rates. Performance not only depends on aggregate metrics like overall loss rate, but also on the stability of the system and the time scales over which events occur. Moreover, we need to separate transient behavior from long-term behavior. For example, an aggregate loss rate can be made arbitrarily good by letting the simulation run arbitrarily long after reaching stability.

To address these issues, we rely on two metrics that (at least to first order) reflect the perceived quality of a real-time, loss-tolerant multimedia stream at the receiver. The first metric is the worst-case loss rate over varying time scales. By considering the short-term loss rates, we can assess the extent of congestion transients. Similarly, by considering long-term loss rates, we can determine how frequently congestion occurs in the steady-state (i.e., by the gap between the long-term and short-term rates).

Our second metric characterizes throughput. In all of the single-source simulations, each receiver eventually reaches the optimal level of subscription. Above this optimum, the network is congested, and below, the network is underutilized. Except for infrequent and brief excursions due to join-experiments, each receiver maintains this level. Accordingly, the throughput can be made arbitrarily close to optimal as described above. Thus we evaluate throughput based on the time it takes the system to converge to the optimal operating point. In an environment where capacity changes dynamically, this rate of convergence characterizes the proximity to optimal throughput. (We ignore the performance loss incurred by a mismatch between the discrete set of possible rates and the exact available bandwidth. In our simulations such mismatch is arbitrary but in practice is difficult to avoid.)

Neither loss rate nor throughput (as measured through convergence time) alone is a comprehensive metric. The system could have a low loss rate with poor throughput (e.g., send nothing), as well as good throughput with high loss rate (e.g., send too much). But taken together, acceptably low loss rates and fast convergence times imply a well-functioning system.

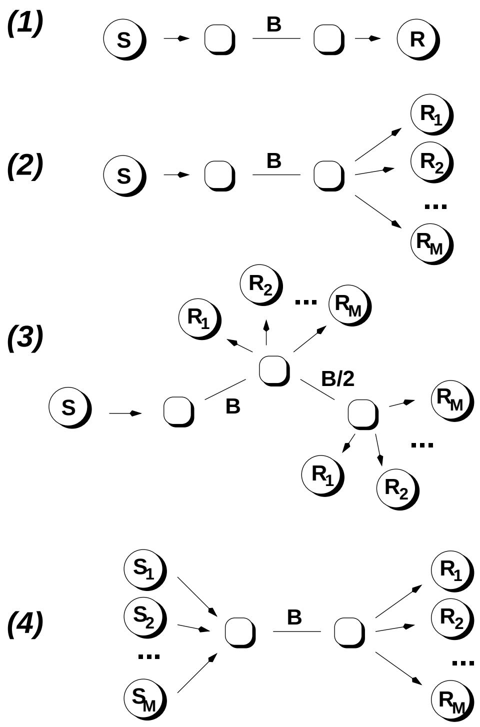

Figure 7: Simulation Topologies.

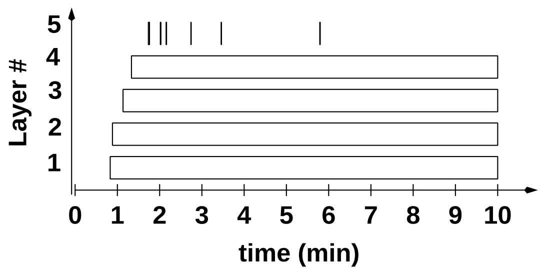

Figure 8: Simple sample path.

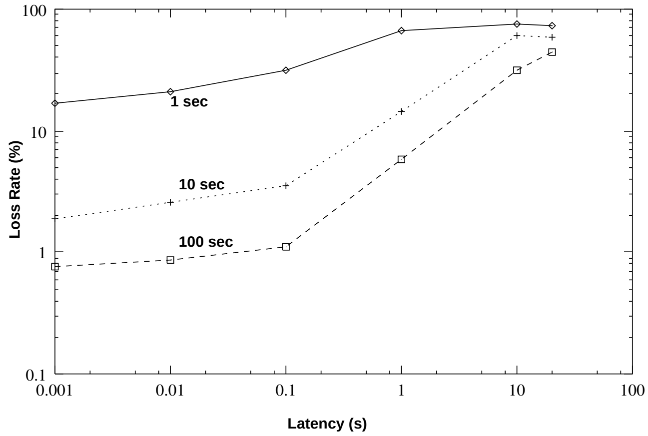

Figure 9: Latency Scalability.

4.2 Experiments

We have simulated RLM in a large number of topologies and configurations. Here we present a subset of the simulations that explores the scalability of RLM in simple environments. The four topologies are illustrated in Figure 7. Topology (1) consists of a single source and receiver separated by a bottleneck link. By analyzing the performance as we vary the latency on the bottleneck link , we explore the protocol's delay scalability.

Topology (2) extends topology (1) with multiple receivers. Here, we explore the scalability of the algorithm with respect to session size. As the size increases, we expect the join-experiment frequency during transients to increase and would like to assess the impact of this on the packet loss characteristic. Also, in large sessions join-experiments inevitably interfere with each other that lead to misinterpretation of the optimal capacity.

Topology (3) explores the performance in the presence of bandwidth heterogeneity by considering two sets of receivers. The first set is connected at the bottleneck rate B while the second set is connected at rate $B/2$. In this scenario, the receivers downstream of the lower speed link must be robust against the high-bandwidth join-experiments from the other set of receivers.

Finally, topology (4) considers the superposition of a large number of independent sessions.

4.3 Results

In this section, we present the results of simulations on the four topologies described above.

Latency Scalability. In the first experiment, we placed a hierarchical CBR source at S in topology (1), ran RLM at R, and fixed the link delay at 10 ms. The simulation was run for 10 (simulated) minutes. In this case, the behavior is predictable. The receiver ramps up to the number of layers supported by the link, then conducts join-experiments at progressively larger intervals until the maximum interval is reached. The duration of the join-experiment is roughly twice the link latency plus the queue build-up time; the impact of packet loss is proportional to the duration of the join-experiment, and thus proportional to the link latency.

This behavior is confirmed in Figure 8, which shows the level of subscription as it evolves over time for this simulation. Note that the receiver reaches the optimal layer subscription in about half a minute and at that point conducts join-experiments at progressively larger time intervals. Each join-experiment lasts less than a second.

To explore the delay sensitivity, we varied the link delay in topology (1) from 1 ms to 20 seconds and computed the worst-case loss rate over different time scales. For each re-

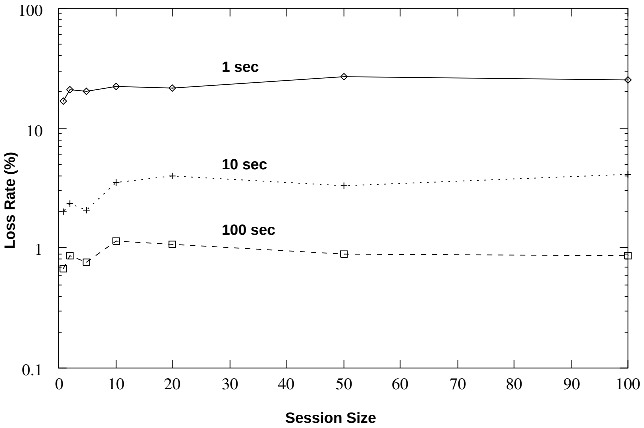

Figure 10: Session-size Scalability.

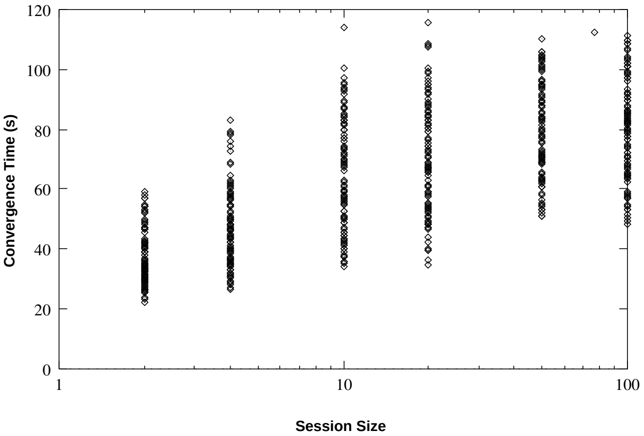

Figure 11: Rate of Convergence.

ceiver, we slide a measurement window (1, 10, or 100 seconds) over the arrival packet process. Within the window, we compute the fraction of lost to total packets and we take the maximum over all time offsets. As the latency increases, we expect the performance to decrease since it takes longer to learn that loss is occurring, prolonging congestion periods. Figure 9 plots the maximum loss rate for a given measurement window versus link latency. For the large measurement window (100 sec) and low delays ($\leq$ 1 sec), the worst-case loss rates are under 1%. On the other hand, the short-term worst-case loss rate (window of 1 sec) ranges from 10 to 20% even for latencies below 100 ms. Finally, each curve has a knee that occurs roughly where measurement window size is twice the link latency. We expect this behavior because the join-experiment congestion period will last at least twice the latency, so the loss rate will be maximal for this size of measurement window.

Session Scalability. In the next experiment, we varied the session size as illustrated in topology (2). Again, we fixed the

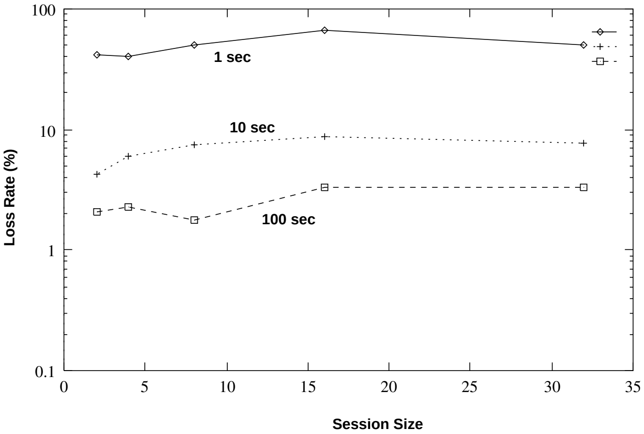

Figure 12: Bandwidth Heterogeneity.

link delays to 10 ms and ran each simulation for 10 minutes. Figure 10 shows the results. For each time scale, we plotted the maximum loss rate against the number of receivers. Because this configuration has multiple receivers, we compute the maximum loss rate by taking the maximum across the worst-case loss rates for each receiver (each computed as the supremum over the sliding window). The graph shows that the worst-case loss rates are essentially independent of the session size. And even for the largest sessions the long-term loss rate is only about 1%.

In this second experiment we also explored how the session size of topology (2) impacts the rate of convergence of each receiver to its optimal level of subscription. Figure 11 is a scatter plot collected over a number of simulation runs. Each point represents the time it took a receiver to reach and maintain its optimal level of subscription (aside from infrequent join-experiments). There is a linear trend in the log plot suggesting logarithmic dependence between convergence time and session size. As the number of receivers grows, we expect longer convergence times since a large number of receivers will suppress join experiments at higher layers. However, because information is shared on each failed join-experiment, receivers rapidly learn the state of the network.

Bandwidth Heterogeneity. Figure 12 illustrates that the algorithm works well even in the presence of large sets of receivers with different bandwidth constraints. The worst-case loss rates are comparable though somewhat higher than the homogeneous cases. The dependence on session size is more notable on short-term time scales because the larger session size increases the probability of colliding join-experiments. Thus, receivers that are genuinely responsible for congestion will transition through the M state before dropping the offending layer. Hence, short-term congestion periods can last longer at larger session sizes. However, the impact of this increase is limited by the detection time estimator, and hence does not increase without bound with the session size.

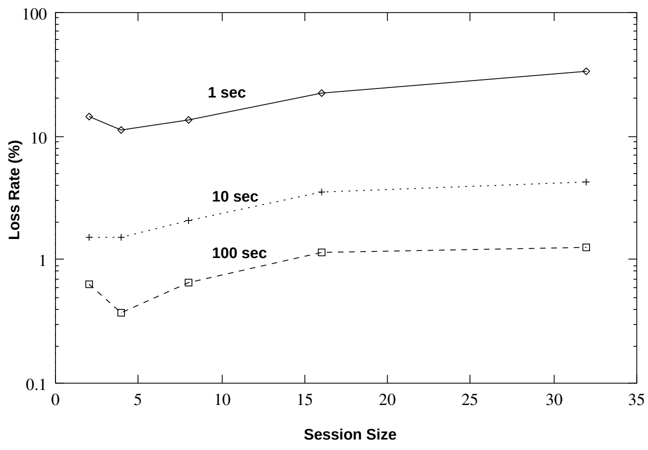

Figure 13: Superposition.

Superposition. Topology (4) explores the performance of RLM when some number of independent single-source/single-receiver sessions share a common link. We ran several simulations and varied the number of source/receiver pairs. The bottleneck link bandwidth was scaled in proportion to the number of pairs and the router queue limit scaled to twice the bandwidth-delay product. Although each simulation converged to an aggregate link utilization close to one, the bandwidth allocation to each pair was often unfair (though no pair was ever starved of bandwidth, since a high-bandwidth session is more likely to experience loss during a join-experiment). Figure 13 illustrates the worst-case loss rate performance, which is consistent with our other simulations. Long-term loss rates are under 1% while medium-term rates are a few percent.

5 Network Implications

Although this paper focuses on the transmission mechanism for layered signals, RLM is only one component of an overall system for multimedia communication. In this section, we discuss some of the implications that RLM has on other components in a comprehensive system for layered multicast transmission.

Receiver-consensus An important requirement of RLM is that all users cooperate. The level of traffic on any given link requires consensus among all of the participants downstream from that link. If just one user in a large group defects and joins all the layers, then nothing can be done to counteract the resulting congestion. Of course, if everyone is running RLM, this will not happen. On the other hand, given the way multicast membership is managed and how RLM might be implemented, more subtle failure modes are possible. For example, a user might temporarily "suspend" an RLM-based application at an inopportune time, causing the end-host to be oversubscribed and unable to react to the resulting congestion. This problem could be solved with the appropriate system fixes (e.g., by deactivating multicast group membership for suspended applications), but complicates deployment.

Group Maintenance. Our simulations show that the performance of RLM depends critically on the join/leave latencies. Once the receiver leaves a group, the network must suppress the flow in a timely manner because congestion persists as long as the network continues to deliver the offending layer. Similarly, to allow receivers to correlate join-experiments with resulting congestion periods, the network must instantiate a new flow expediently. In the case of IP Multicast, the Internet Group Management Protocol (IGMP) [12] carries out both of these operations on reasonable time scales. When a receiver joins a new group, the host immediately informs the next-hop router, which in turn, immediately propagates a *graft* message up the multicast distribution tree in order to instantiate the new group. If the flow already exists, the graft is suppressed. The leave-case is more complicated because the next-hop router must determine when all the hosts on a subnet have left the group. To do this, when a host drops a group, it broadcasts a "leave group" message on the subnet and the router responds by briefly accelerating its normal membership query algorithm. Upon quickly determining that no members remain, the router sends a *prune* message up the distribution tree to suppress the group.

Fairness. In a network with arbitrary numbers of senders each transmitting to an arbitrary number of receivers, each receiver should individually adjust its number of layers so that the aggregate system performance is "good". When there is only a single source sending some number of receivers, "good" is well-defined: each receiver should receive the maximum number of layers that the network can deliver. But when there are multiple sources, "good" is ill-defined because it depends on the relative importance of the users within and across sessions. In short, an aggregate performance metric depends on how group "fairness" is defined.

Rather than tackle the problem of defining fairness, we have placed our initial focus on the design of RLM in *isolation*, that is, when a single source sends to multiple receiver without interfering traffic. RLM alone *does not* provide fairness. In general it is not possible to achieve a "fair" allocation of bandwidth without some additional machinery in the network, even if all the end-nodes cooperate [22]. But, if machinery for fairness is added to the network, RLM should work effectively in concert with it.

Similar circumstance surrounds the design of TCP. TCP congestion control works well in isolation but in aggregation can be unfair [13]. As an optimization, network mechanism can be introduced to make TCP perform better: Random Early Detection (RED) [14] gateways or Fair Queuing (FQ) [10] routers minimize the interaction between connections to improve fairness. Similarly, we can design RLM to behave relatively well in a loosely controlled, drop-tail, best-effort network, and as an optimization add RED or FQ to the network (or to parts of the network) to improve aggregate performance.

All of our simulation results assume that routers drop packets on arrival when their queue is full. This widely deployed drop-tail discard policy is unfortunate because it delays the warning signal from the receivers until well after congestion has occurred. RED gateways, on the other hand, react to incipient congestion by discarding packets at the onset of congestion (i.e., when the rate of change of the queue size exceeds a threshold) [14]. RED's early reaction to congestion interacts nicely with RLM because it allows receivers to react to congestion before the bottleneck link becomes fully saturated. We have run simulations using RED gateways in place of drop-tail gateways and the loss rate performance indeed improves.

6 The Application

To complement the layered transmission system provided by RLM, we have developed a layered source coder adapted for this environment [26]. Our goal is to design, build, and evaluate all of the components that contribute to a scalable video transmission system, ensuring that the pieces of the design interact well with each other. To this end, our system is based on Clark and Tennenhouse's Application Level Framing (ALF) protocol architecture [5]. While ALF says that an application's semantics should be reflected in the design of its network protocol, we further believe that the "network's semantics" should also be reflected in the application design. For example, instead of designing an "optimal" framing protocol for a compressed-video bitstream emanating from a black box (i.e., reflecting application semantics in the protocol), we claim the compression format itself should be tailored to its environment (i.e., reflecting network constraints in the application design).

The ALF model is embodied in the co-design of RLM and our layered codec. The characteristic of the RLM communications environment substantially influences the design of our layered codec, while conversely, the layered compression model drives the design of RLM. Rather than design the sub-components in isolation, we design them jointly to complement each other and thereby produce an application with high performance not only over the network but also through the end-system and ultimately to the user.

Two key features of our layered coder are its resilience to packet loss and its low complexity. These characteristics make it especially well suited for scalable video transmission over the Internet. First, the scheme is robust to the RLM join-experiments (and the background transient congestion common in the Internet) since transient periods of congestion are gracefully accommodated through its loss resilience. Moreover, because join-experiments are announced to the group, the source can dynamically modify its coding algorithm to trade bandwidth for loss resilience. Second, the algorithm's low complexity admits an efficient software implementation that can be readily distributed to many users in the Internet.

Because RLM relies only on mechanisms that are already widely deployed in the Internet, we can field our system by building it into an application. We are currently implementing RLM and our layered codec in the UCB/LBNL video conferencing tool *vic* [25]. Vic's network transport is based on the Real-time Transport Protocol (RTP) [28], an application level protocol for multimedia transport standardized by the Internet Engineering Task Force. Although RTP has proven to be a solid foundation for interoperable real-time audio/video applications, it was designed without any explicit notion of a layered signal representation. In joint work, we have extended RTP for layered stream delivery [31] and are currently implementing our proposed changes in vic.

Since the RLM protocol processing is not in the "fast path", run-time performance is not critical. In fact, our prototype is implemented almost entirely in the interpreted language Tcl [27]. Vic's C++ packet processing code performs upcalls into Tcl when loss is detected. Tcl runs the adaptation algorithm and manipulates IP Multicast group membership via downcalls back to C++.

7 Related Work

The idea that the rate of an information source can be adjusted by degrading reconstruction quality was born in rate-distortion theory first developed by Shannon [30]. The rate-distortion framework forms the bedrock of traditional video codec design, where codec parameters (i.e., compression quality) are dynamically adjusted to match the transmission rate of a CBR communications channel. Gilge and Gusella [19] applied the CBR coding model to packet networks by viewing the network as the codec smoothing buffer. They proposed an end-to-end design that uses explicit feedback from the receiver to throttle the video transmission rate at the source.

Kanakia et al. [23] build on Gilge and Gusella's model with an architecture where the feedback signal is derived directly from the network. The bottleneck switch or router along the transmission path communicates its queuing delay back to the source. A controller uses this information to adjust the output rate of the source coder, allowing the source to react to queue buildup before packet loss occurs.

These source-based rate-adaptation schemes are poorly matched to multicast environments. QMTP [34] and the IVS congestion control scheme [1] adapt by soliciting feedback from the receivers in a scalable fashion, but these schemes do not cope well with bandwidth heterogeneity. Either low-capacity regions of the distribution are overwhelmed or high-capacity regions are underutilized.

Shacham proposed a scheme based on layered transmission and compression to solve the heterogeneity problem [29]. He focused on computing fixed, optimal routes for a given traffic mix and on error control procedures for coping with loss rather than reacting to it.

Taubman and Zakhor [32] have developed a layered video compression algorithm that performs on par with the best non-layered schemes. Their focus is on the compression technology rather than the network, and their network model is based on signaling and packet discard policies that are not widely deployed.

The "Discrete Scaling" mechanism in the Heidelberg Transport System (HeiTS) [9] uses a receiver-oriented scheme for adapting to delivered bandwidth. Here, receivers open and close ST-II [3] multicast connections to adapt to bandwidth. The authors do not discuss adaptation algorithms or report any implementation results.

Deering first suggested that the IP Multicast be used as a layered transmission system where layers are individually mapped onto multicast groups [6]. Both Chaddha and Gupta [4] and Bolot and Turletti [33] describe this architecture but do not present an adaptation algorithm or implementation. Brown et al. have implemented a multi-resolution extension to the CU-SeeMe video conferencing system where IP Multicast receivers can subscribe to either a 160x120 or a 320x240 stream by joining either one or two multicast groups [2]. Receivers drop down to the 160x120 resolution when they detect high packet loss rates.

Concurrent with our work, Hoffman and Speer have built a similar system based on the layered multicast architecture [20]. They use multiple frame rates of JPEG video to generate a temporal hierarchy and employ two techniques for adaptation. Their first technique is a negotiation algorithm run by each receiver that obtains the highest available quality of service explicitly from the network (e.g., using RSVP [35]). Their second approach uses layered multicast with an aggressive adaptation scheme where a new receiver subscribes to all the layers in the distribution and drops layers until the quality of the delivered stream is adequate.

8 Future Work

RLM is the first comprehensive instance of a receiver-driven multicast adaptation algorithm and we have just scratched the surface of this problem. While we have evaluated RLM in terms of packet loss rates, the ultimate evaluation metric is the level of quality perceived by the user. We will soon carry out qualitative performance measurements both in a controlled environment as well as by fielding an implementation in the Internet. The litmus test will be whether or not the user community adopts the RLM and the layered codec as the preferred configuration.

We also plan to experiment with algorithms that dynamically adjust the bit-rate allocation of the different compression layers. Our compression scheme produces an *embedded code*, which has the property that any prefix of the compressed bitstream remains a valid representation at a lower quality. In other words, a given video frame can be successively refined at a very fine granularity. Using this property, we can partition the bit-rate arbitrarily among layers and vary this allocation dynamically, from frame to frame or slowly over time. As an optimization, we can use scalable, low-rate feedback from the receivers (e.g., as provided by RTCP [28]) to tailor the rate allocation to the environment. For example, if the entire session is connected at high-rate, but one user is connected at ISDN rate, we could produce a two-layer stream rather than a higher-complexity multi-layer stream.

In an integrated services network, a receiver could explicitly negotiate with the network to determine the appropriate number of layers [20], with or without consideration of a pricing structure. In this case, RLM adaptation is not necessary. On the other hand, if the granularity of resource management were not as fine-grained, then RLM adaptation within an integrated services environment might still make sense. For example, Class Based Queuing (CBQ) [16] could be used to provide an "adaptive-rate video" traffic class with some specified bandwidth. Then within this CBQ class, video sessions could contend for the aggregate class bandwidth using RLM. This approach has the desirable side effect that RLM is shielded from interactions with other protocols.

The RLM framework could be combined with the Scalable Reliable Multicast (SRM) protocol [17] in the LBNL whiteboard, *wb*, to optimize the latency of rate-controlled transmissions. Because SRM uses a token-bucket rate-controller, it has the same limitations that single-layer video has in heterogeneous environments. On the other hand, several token-buckets with a range of rates could be used in tandem with multiple multicast groups and RLM. SRM would *simulcast* new data across all of the token-buckets to trade off bandwidth for latency. By spacing the rates exponentially, the overhead of the simulcast is minimized.

Our simulations explored interactions only among different instances of RLM. We plan to explore interactions with other bandwidth-adaptive protocols like TCP. Similarly, we are studying the interactions among multiple RLM sessions in the context of different scheduling disciplines. For example, fair-queuing (FQ) routers drop packets from sessions that use more than their proportion of bandwidth. Thus, if the FQ allocation granularity is the set of layered multicast groups, then RLM should converge to a fair operating point. Likewise, because RED gateways drop packets from connections with probability proportional to their bandwidth consumption, the system should converge to approximate fairness.

Finally, we intend to improve our modeling and analysis of the problem. We are developing a model for layered signal sources (based on our codec work) that expresses the dependencies between packets in different layers. This will allow us to develop better loss metrics since losses in the core layers tend to impact higher layers. We are also investigating a tractable analytic model of the protocol actions on simple topologies to characterize convergence and loss probabilities as a function of scale.

9 Summary

We have proposed a framework for the transmission of layered signals over heterogeneous networks using receiver-driven adaptation. We evaluated the performance of RLM through simulation and showed that it exhibits reasonable loss and convergence rates under several scaling scenarios. While many existing solutions are either network-oriented or compression-oriented, our focus is on the complete systems design. We described our work on a low-complexity, error-resilient layered source coder, which when combined with RLM, provides a comprehensive solution for scalable multicast video transmission in heterogeneous networks.

10 Acknowledgments

This work benefited from several thought-provoking discussions with Sally Floyd. Sally proposed the model where RLM is used within a CBQ class allocation. Elan Amir, Hari Balakrishnan, Sugih Jamin, Deana McCanne, Vern Paxson, Scott Shenker, and the anonymous reviewers provided thoughtful comments on drafts of this paper. Lixia Zhang inflicted an early version of this paper on her seminar at UCLA, which generated constructive feedback. Finally, Steve Deering participated in several early discussions of this work.

Support for this work was provided by the the Director, Office of Energy Research, Scientific Computing Staff, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. Equipment grants and support were provided by Sun Microsystems, Digital Equipment Corporation, and Silicon Graphics Inc.

References

[1] BOLOT, J.-C., TURLETTI, T., AND WAKEMAN, I. Scalable feedback control for multicast video distribution in the Internet. In *Proceedings of SIGCOMM '94* (University College London, London, U.K., Sept. 1994), ACM.

[2] BROWN, T., SAZZAD, S., SCHROEDER, C., CANTRELL, P., AND GIBSON, J. Packet video for heterogeneous networks using CU-SeeMe. In *Proceedings of the IEEE International Conference on Image Processing* (Lausanne, Switzerland, Sept. 1996).

[3] CASNER, S., LYNN, J., PARK, P., SCHRODER, K., AND TOPOLCIC, C. *Experimental Internet Stream Protocol, version 2 (ST-II).* ARPANET Working Group Requests for Comment, DDN Network Information Center, SRI International, Menlo Park, CA, Oct. 1990. RFC-1190.

[4] CHADDHA, N., AND GUPTA, A. A frame-work for live multicast of video streams over the Internet. In *Proceedings of the IEEE International Conference on Image Processing* (Lausanne, Switzerland, Sept. 1996).

[5] CLARK, D. D., AND TENNENHOUSE, D. L. Architectural considerations for a new generation of protocols. In *Proceedings of SIGCOMM '90* (Philadelphia, PA, Sept. 1990), ACM.

[6] DEERING, S. Internet multicast routing: State of the art and open research issues, Oct. 1993. Multimedia Integrated Conferencing for Europe (MICE) Seminar at the Swedish Institute of Computer Science, Stockholm.

[7] DEERING, S., ESTRIN, D., FARINACCI, D., JACOBSON, V., GUNG LIU, C., AND WEI, L. An architecture for wide-area multicast routing. In *Proceedings of SIGCOMM '94* (University College London, London, U.K., Sept. 1994), ACM.

[8] DEERING, S. E. *Multicast Routing in a Datagram Internetwork.* PhD thesis, Stanford University, Dec. 1991.

[9] DELGROSSI, L., HALSTRICK, C., HEHMANN, D., HERRTWICH, R. G., KRONE, O., SANDVOSS, J., AND VOGT, C. Media scaling for audiovisual communication with the Heidelberg transport system. In *Proceedings of ACM Multimedia '93* (Aug. 1993), ACM, pp. 99–104.

[10] DEMERS, A., KESHAV, S., AND SHENKER, S. Analysis and simulation of a fair queueing algorithm. *Internetworking: Research and Experience 1* (1990), 3–26.

[11] ERIKSSON, H. Mbone: The multicast backbone. *Communications of the ACM 37*, 8 (1994), 54–60.

[12] FENNER, W. *Internet Group Management Protocol, Version 2.* Internet Engineering Task Force, Inter-Domain Multicast Routing Working Group, Feb. 1996. Internet Draft expires 8/31/96.

[13] FLOYD, S., AND JACOBSON, V. On traffic phase effects in packet-switched gateways. *Internetworking: Research and Experience 3*, 3 (Sept. 1992), 115–156.

[14] FLOYD, S., AND JACOBSON, V. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking 1*, 4 (Aug. 1993), 397–413.

[15] FLOYD, S., AND JACOBSON, V. The synchronization of periodic routing messages. In *Proceedings of SIGCOMM '93* (San Francisco, CA, Sept. 1993), ACM, pp. 33–44.

[16] FLOYD, S., AND JACOBSON, V. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking 3*, 4 (Aug. 1995), 365–386.

[17] FLOYD, S., JACOBSON, V., MCCANNE, S., LIU, C.-G., AND ZHANG, L. A reliable multicast framework for light-weight sessions and application level framing. In *Proceedings of SIGCOMM '95* (Boston, MA, Sept. 1995), ACM.

[18] GARRETT, M. W., AND WILLINGER, W. Analysis, modeling and generation of self-similar VBR video traffic. In *Proceedings of SIGCOMM '94* (University College London, London, U.K., Sept. 1994), ACM.

[19] GILGE, M., AND GUSELLA, R. Motion video coding for packet-switching networks—an integrated approach. In *Proceedings of the SPIE Conference on Visual Communications and Image Processing* (Boston, MA, Nov. 1991), ACM.

[20] HOFFMAN, D., AND SPEER, M. Hierarchical video distribution over Internet-style networks. In *Proceedings of the IEEE International Conference on Image Processing* (Lausanne, Switzerland, Sept. 1996).

[21] JACOBSON, V. Congestion avoidance and control. In *Proceedings of SIGCOMM '88* (Stanford, CA, Aug. 1988).

[22] JAFFE, J. M. Bottleneck flow control. *IEEE Transactions on Communications 29*, 7 (July 1981), 954–962.

[23] KANAKIA, H., MISHRA, P. P., AND REIBMAN, A. An adaptive congestion control scheme for real-time packet video transport. In *Proceedings of SIGCOMM '93* (San Francisco, CA, Sept. 1993), ACM, pp. 20–31.

[24] MCCANNE, S., AND FLOYD, S. *The LBNL Network Simulator*. Lawrence Berkeley Laboratory. Software on-line[2].

[25] MCCANNE, S., AND JACOBSON, V. *vic:* a flexible framework for packet video. In *Proceedings of ACM Multimedia '95* (Nov. 1995), ACM.

[26] MCCANNE, S., AND VETTERLI, M. Joint source/channel coding for multicast packet video. In *Proceedings of the IEEE International Conference on Image Processing* (Washington, DC, Oct. 1995).

[27] OUSTERHOUT, J. K. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

[28] SCHULZRINNE, H., CASNER, S., FREDERICK, R., AND JACOBSON, V. *RTP: A Transport Protocol for Real-Time Applications*. Internet Engineering Task Force, Audio-Video Transport Working Group, Jan. 1996. RFC-1889.

[29] SHACHAM, N. Multipoint communication by hierarchically encoded data. In *Proceedings IEEE Infocom '92* (1992), pp. 2107–2114.

[30] SHANNON, C. E. A mathematical theory of communication. *Bell Systems Technical Journal 27* (1948), 379–423.

[31] SPEER, M. F., AND MCCANNE, S. *RTP usage with Layered Multimedia Streams*. Internet Engineering Task Force, Audio-Video Transport Working Group, Mar. 1996. Internet Draft expires 9/1/96.

[32] TAUBMAN, D., AND ZAKHOR, A. Multi-rate 3-D subband coding of video. *IEEE Transactions on Image Processing 3*, 5 (Sept. 1994), 572–588.

[33] TURLETTI, T., AND BOLOT, J.-C. Issues with multicast video distribution in heterogeneous packet networks. In *Proceedings of the Sixth International Workshop on Packet Video* (Portland, OR, Sept. 1994).

[34] YAVATKAR, R., AND MANOJ, L. Optimistic strategies for large-scale dissemination of multimedia information. In *Proceedings of ACM Multimedia '93* (Aug. 1993), ACM, pp. 1–8.

[35] ZHANG, L., DEERING, S., ESTRIN, D., SHENKER, S., AND ZAPPALA, D. RSVP: A new resource reservation protocol. *IEEE Network 7* (Sept. 1993), 8–18.

[2] http://www-nrg.ee.lbl.gov/ns/