Network Support for Multicast Video Distribution*

Samrat Bhattacharjee
Kenneth L. Calvert
Ellen W. Zegura
Networking and Telecommunications Group
College of Computing
Georgia Tech
Atlanta, GA 30332-0280
{bobby,calvert,ewz}@cc.gatech.edu

GIT–CC–98/16

Abstract

Multicast video distribution in a best-effort environment presents challenges to system designers, including heterogeneity in the bandwidth availability on the paths from the sender to the receivers and dynamic behavior in the network and set of receivers over time. Classic approaches to dealing with dynamic conditions involve adaptation at the sender (for unicast) and adaptation driven by the receivers (for multicast). Both approaches have limitations that affect the quality of video received. In this paper, we consider a third option for the location of adaptation, namely: in the network. We demonstrate that a modest amount of state and computation at network routers can yield significant performance gains for multicast video distribution. Our schemes maintain the advantages of receiver-based adaptation, while overcoming the limitation. Since the network applies the adaptation, the time and place for adaptation can better match network conditions. Further, the adaptation can occur more rapidly, without the need for route changes. Finally, the adaptation can occur at finer granularity, providing better quality and more graceful degradation to receivers.

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332–0280

*This work was supported in part by DARPA under contract number N66001-97-C-8512.

1 Introduction

A variety of applications, including distance learning and video conferencing, require delivery of video to a geographically distributed set of receivers. For several reasons, these applications may prefer to use a best-effort service, as opposed to a guaranteed service. The unpredictability of video transmission, and of the set of receivers in a multicast group, makes a reserved service difficult to use. A large set of receivers requires resources over many network links, thus increasing the possibility that a reservation request will be refused.

Three characteristics of multicast video distribution present particular challenges to system designers working in a best-effort environment. First, the paths from the sender to the receivers are likely to be heterogeneous in bandwidth availability. Second, the system will typically exhibit dynamic behavior over time. For example, the bandwidth on a single path may change during the lifetime of the connection and/or the set of receivers in the application may change. Finally, for applications with a large number of receivers, scaling becomes a concern.

1.1 Three Locations for Adaptation

The classic approach to dealing with dynamic network conditions in unicast applications is *sender adaptation* [11]. That is, the sender infers changes in path characteristics and adapts the transmission accordingly. However, sender adaptation has well-known challenges. The first is the time interval required for the sender to detect congestion, adapt to bring losses under control, and have the controlled-loss data propagate to the receiver. The other well-known challenge of sender adaptation is detecting an increase in available bandwidth. The problem arises in traditional best-effort networks because loss is the only mechanism for determining available bandwidth. The sender must detect the easing of congestion by attempting to transmit at a higher rate and monitoring for losses. Multicast compounds the difficulty of sender adaptation. If the sender adapts to the least capable receiver, then all receivers must suffer lowest-common-denominator performance. If the sender relies upon feedback information from receivers (e.g., concerning losses) then the system is prone to feedback implosion for large numbers of receivers. Despite these difficulties, systems have been developed that use sender adaptation for multicast video distribution (e.g., [7]).

The difficulty of sender adaptation for multicast has led more recently to the exploration of *receiver-based adaptation* [14, 16, 13]. Under these schemes, the sender transmits multiple streams suitable for a range of path characteristics. Each receiver joins the stream (or streams) that can be supported by the individual path from the sender. This approach solves the problems of feedback implosion and lowest-common-denominator performance, however other problems persist. The difficulty in detecting an increase in bandwidth remains. To detect a bandwidth increase, the receiver must conduct a "join experiment," attempting to receive at a higher quality and monitoring for losses. Unless these experiments are coordinated, receivers will interfere with one another. To make a change in the received transmission, the receiver must join or leave a multicast group, incurring the delay required for routing changes to propagate. Finally, to avoid complexity at the sender and in the multicast group maintenance, receiver-based schemes typically support coarse granularity in receiver capabilities. That is, the number of distinct bandwidth classes is often quite limited.

In this paper, we consider a third option for the location of adaptation, namely: in the network. We demonstrate that a modest amount of state and computation at network routers can yield significant performance gains for multicast video distribution. Our schemes maintain the advantages of receiver-based adaptation while overcoming the limitations mentioned above. Since the network applies the adaptation, the time and place for adaptation can better match network conditions. Further, the adaptation can occur more rapidly, without the need for route changes. Finally, the adaptation can occur at finer granularity, providing better quality and more graceful degradation to receivers.

1.2 An Example

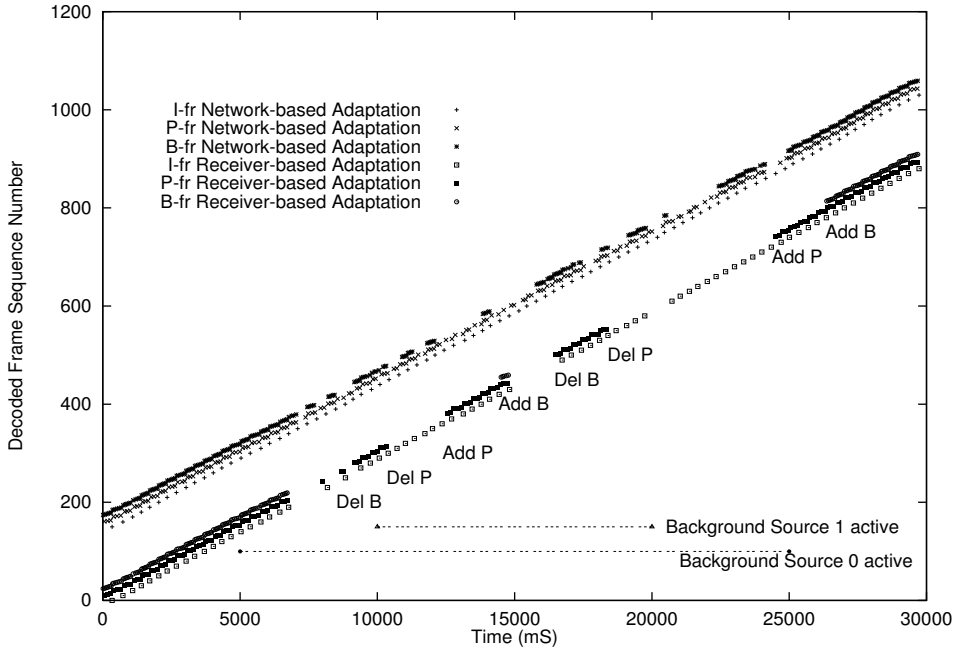

Figure 1: Comparison of Receiver- and Network-based Adaptation

As motivation, we give a preview of the performance improvement possible using network-based adaptation as compared to receiver-based adaptation. The exact details of the mechanisms are discussed later in the paper; our point here is to highlight the broad trends in behavior.

Figure 1 shows the results of a simulation that sends MPEG video to a receiver over a link that is congested from time 5000 ms to time 25000 ms by one background source, and further congested from time 10000 ms to time 20000 ms by a second background source. Two types of adaptation are shown, receiver-based layering (lower points) and network-based adaptation (upper points). Each point indicates that a frame was received and decodable; the x-axis gives the time the frame was received and the y-axis gives the offset frame sequence number [1]. MPEG video has three types of frames (I, P and B); each type is shown with a different set of

[1] The sequence numbers are offset so that the data can be better viewed.

points. The details of the two transmission schemes will be presented later. The basic idea is that in the receiver-based scheme, receivers join and leave multicast groups (flows) according to their perception of current congestion conditions.

This figure has three key features. First, the number of frames received by the network-based adaptation scheme is considerably higher than the receiver-based adaptation scheme (630 frames versus 420 frames). Second, the receiver-based adaptation scheme is slow to react to changes in the congestion. For example, although the second background source goes away at time 20000 ms, the P-stream is not added until time 25000 ms. Under network-based adaptation, the receiver gets P-frames and even some B-frames during the time interval (20000,25000). Third, the receiver-based scheme occasionally experiences periods of such severe overload that no frames are received and decodable. This happens at approximately time 7500 and again at time 15000.

1.3 Roadmap

This paper is organized as follows. In the next section, we describe our network-based adaptation approach, with emphasis on the state and computation requirements. We then describe a common receiver-based adaptation approach based on MPEG layering that is used as a comparison for our network-based approach. In Section 4 we provide details on the simulation environment and metrics used to compare our schemes to traditional receiver-based adaptation. Section 5 contains the results of a set of experiments that demonstrate when network-based adaptation offers significant advantages. After examining the advantages of network-based adaptation, we explicitly consider some of the common concerns about adding functionality to the network. (See Section 6.) We discuss related work in Section 7, and conclude in Section 8.

2 Network-based Adaptation

The basic operation of network-based adaptation is as follows: the sender transmits a full rate stream on a single multicast group, to which all receivers subscribe. The routers in the multicast tree have information (established *a priori* by out-of-band mechanisms and/or carried in-band as part of data packets) about how to intelligently reduce the rate of the stream in the face of congestion on an outgoing link. For routers in the tree that do copying, the rate adaptation occurs after copying the multicast packet. Thus each outgoing link is treated separately and each receiver obtains performance based on the path from the sender, not influenced by other receivers or parts of the multicast tree. The adaptation occurs when dictated by congested conditions, thus an increase in available bandwidth can immediately be utilized and a decrease in available bandwidth immediately triggers controlled reduction of the rate.

This basic approach can be adapted in various ways. If the sender has information on the maximum rate that any receiver can receive (e.g., based on static information or infrequent feedback), that can be used to determine the "full rate" of the sending stream. Our techniques do not require that all routers to have the ability to intelligently reduce rate. The other routers can apply a standard rate reduction technique such as dropping any packet that arrives at a full output queue ("tail drop"). (For best performance, the intelligent routers would be

strategically placed at boundaries between bandwidth-rich and bandwidth-poor areas of the network.) If the application makes use of multiple streams from different senders, the routers can be provided with information about how to intelligently reduce the rate of the application, potentially by discriminating across senders.

Network-based adaptation raises a number of questions:

- How is congestion detected?
- What techniques are used to reduce the rate of the stream?
- What are the state and computational requirements of the rate reduction techniques?
- How do the routers obtain the information needed to apply rate-reduction techniques?

Triggers for initiating congestion control have been studied elsewhere in the literature, both for end-to-end mechanisms as in TCP [11, 2] and for intra-network mechanisms such as ATM ABR [8]. This issue is orthogonal to the other components of our scheme, thus we consider a simple trigger that initiates congestion when the arriving packet will not fit in the output queue.

Next we provide detail on the reduction techniques, the state and computation requirements, and the methods used to get reduction information to routers.

2.1 Reduction Techniques

The success of network-based adaptation depends on the ability of the reduction techniques to preserve —as much as possible— the "quality" of the stream, as evaluated at the receiver. Good reduction techniques, therefore, will depend on the particular application and information coding scheme. We focus on MPEG-encoded video, though our reduction techniques are based on general principles that could be applied to other video coding schemes or to applications other than video. We have previously considered network-based congestion control for unicast MPEG video [4]; some of the mechanisms used for unicast are similar to what we describe here.

MPEG encodes video as a sequence of *frames*, of three types. An I-frame is encoded independently of all other frames. A P-frame may be encoded relative to another I- or P-frame. A B-frame may be encoded relative to two other I- or P-frames. A *Group of Pictures* (GOP) consists of a single I-frame followed by structured order of P- and B-frames. The video is encoded as a sequence of GOPs, and the structure of the GOPs may change within a stream.

The exact dependencies between frames will depend on the particular video being encoded. For our experiments, we model the dependencies in the following way. First, we assume a GOP structure encoded as the 10-frame sequence "IBBPBBPBBP" and transmitted as "IPBBPBBPBB" [2]. We assume that each P-frame is encoded using the I-frame that starts the GOP; we assume that each B-frame is encoded using both the preceding P-frame in the transmission sequence and the I-frame that starts the GOP. These assumptions are consistent with the encoding rules for MPEG.

Initially, we consider a frame to be *decodable* only if the receiver gets all packets from the frame **and** all packets from all frames that this frame depends upon. For example, decoding a

[2] MPEG transmission typically occurs in a different order than the playout order to facilitate decoding.

P-frame requires receiving all packets in the P-frame and all packets in the dependent I-frame. In Section 5.3 we experiment with "smarter" decoders that are able to make use of partially received frames.

Our reduction technique exploits two features of MPEG: the application-layer frame units and the decoding dependencies across frames. Our "GOP-level discard" technique makes use of three mechanisms. The first two mechanisms involve discarding packets that are already in the output queue.

- **Discard-Other.** When a packet arrives and the queue is full, packets belonging to this flow but of lower priority[3] are discarded in order to fit the current packet in the output queue. Further, if a packet from a particular frame is discarded, all packets that depend on the discarded packet are discarded. For example, if a packet belonging to an I-frame encounters congestion (full queue), then a packet belonging to a P-frame may be discarded to accommodate the "I packet". Further, the deletion of a single "P-packet" will cause all the packets in that P-frame along with all packets in B-frames that depended on the discarded P-packet to be discarded as well. In our implementation, B-frames are always discarded before P-frames. Thus, a P-frame is discarded if and only if an I-frame is congested and the output queue does not contain any more B-frames.

- **Discard-Own.** When a packet arrives and it must be discarded, then all packets already enqueued and that belong to the current frame are discarded as well. For example, if a packet belonging to a B-frame encounters a full queue, it must be discarded. The Discard-Own mechanism is used to discard all the other packets that belonged to the current frame and were already enqueued in the output queue.

- **Discard-On-Arrival.** The third mechanism involves discarding arriving packets before they enter the output queue. If a packet is discarded (due to congestion or other discards described above), the node state is set such that all subsequent packets that depend on the discarded packet are discarded upon arrival. For example, if a packet from an I-frame is discarded, all subsequent dependent packets (in this case, all packets until the next I-frame) are denied entry to the output queue. Similarly, if a packet from a P-frame is discarded, all arriving B-frames that depend on this P-frame are discarded on arrival, etc. Note that for this mechanism, only a few bits of state need to be set (and reset), and access to the output queue is not required.

2.2 State and Computation Requirements

All three mechanisms described above can be implemented with a small amount of per-flow state and modest per-packet-arrival computation. The Discard-Other mechanism must be able to (1) determine if enough lower priority packets are in the queue to make room for the incoming packet, and (2) remove specific packets in the queue. The first task can be accomplished by keeping per-flow state that records the cumulative size of B-packets and the cumulative size of P-packets that are currently in the queue. This state is updated whenever a packet for this flow enters or departs the queue. The second task requires an interface to the queue that supports deletion based on a pattern match. If the streams are properly labeled, the pattern match can be extremely efficiently implemented (as is the case in our

[3] The priority ordering is strict, with I-packets higher than P-packets higher than B-packets.

implementation [4] and simulations). These deletions can clearly be implemented in time linear in the length of the queue; more efficient implementations are possible (by keeping a separate queue per frame type). The Discard-Own mechanism can either be implemented similar to the Discard-Other (deletion based on pattern-match), or be implemented by keeping a one frame buffer in which complete frames are reassembled before entry to the output queue. The reassembly buffer makes it easier to implement Discard-Own and guarantees that frame fragments are not transmitted; however, it also adds a reassembly latency at each node. In our case, we do not implement the one frame reassembly buffer.

The incoming discard mechanism must keep state recording the frame type and frame sequence number of the packets of each frame type that were discarded. A simple finite state machine is used to decide whether to drop an incoming packet, using the information on the packet that was dropped and the contents of the incoming packet header. The decision consists of a small number of comparisons.

In Section 6.1 we examine the performance of the GOP-level discard scheme with some, but not all, of the three mechanisms enabled.

2.3 Router Information

As indicated above, the reduction techniques make use of a combination of information carried in packets (e.g., frame type and frame sequence number) and information resident at routers. The information resident at a router might be installed off-line and out-of-band (e.g., by a routing manufacturer) to provide a value-added service to MPEG video. Alternatively, the router might support a programmable interface that allows some form of on-line installation of state and computation. Such interfaces are being developed under various Active Networking projects [1, 15].

3 Receiver-based Adaptation

We compare our network-based adaptation scheme to a receiver-based adaptation scheme based on the main ideas in the Receiver-driven Layered Multicast approach [14]. Specifically, the sender transmits three streams —each on a different multicast group— corresponding to the I-frames, P-frames and B-frames of the MPEG video. The I-frames form the base layer; the P-frames form the first enhancement layer and the B-frames form the second enhancement layer. A receiver initially joins all three groups. As described in more detail below, a receiver uses periodic monitoring of the received streams to decide whether to add a layer (by joining a multicast group) or delete a layer (by leaving a multicast group). Each receiver acts independently.

The receivers join and leave multicast groups based on monitoring the quality of the received video stream. During each monitoring interval, the receiver measures two quantities: (1) the number of bytes received that could be decoded and (2) the number of bytes received. At the end of each interval, the receiver takes the ratio of the first quantity to the second. If the result is greater than or equal to a fixed threshold $hiWater$, then the lowest layer that is not currently being received is added. Conversely, if the result is less than a fixed threshold $loWater$, then the highest layer that is currently being received is deleted. However, our receivers never drop the lowest layer (the I-frames). In our experiments, we set the value of

the monitoring interval to 6 GOPs, or every two seconds. We experiment with different values for the thresholds; most of our experiments use values of 0.85 and 0.90.

We give some benefit to the receiver-based adaptation scheme by modeling all multicast group joins and leaves as instantaneous events. That is, when a receiver decides to join or leave a group, no time elapses for the updates to the corresponding routing tables.

4 Experimental Environment

Our results were obtained using a locally developed discrete-event simulator —ANSWER— that has specifically been developed for evaluating active networks. In this section, we detail the source models used for both video and data sources, the experimental topology, and the metrics we use to evaluate the performance of each adaptation scheme.

4.1 Source Models

The video sources in the simulations generate three distinct video layers, corresponding to layered MPEG video with 10 frames per GOP, at the rate of 30 frames per second. The I-, P-, B- layers generate 3, 9, 18 frames per second each, respectively. The GOP structure, and the frame sizes are shown in Table 1. The maximum size of a packet is 576 bytes, and larger frames are segmented over several packets. Note that the B-frames account for 60% of all frames, but only 13% of the bandwidth, while the I-frames only constitute 10% of the total frames, while consume more than half the bandwidth. Our analysis and the analysis of others [10, 9, 6] indicate that this is a reasonable model for frame sizes for MPEG.

For layered video, each layer is transmitted over a separate multicast group, and the destinations may choose to receive only a subset of all the layers. For the network-based adaptation, all the packets generated by the video sources are transmitted over a single multicast group. All video receivers have a re-assembly buffer of size 2 GOPs, which is used to reassemble the packets to reconstruct the original GOP structure.

Frame Type	Rate (fps)	Max. (bytes)	Mean (bytes)	Min. (bytes)	Generation Distribution	Rate (Kbps)
I	3	17000	16000	15000	Uniform	385
P	9	17000	2100	512	Exponential (trunc.)	150
B	18	17000	550	256	Exponential (trunc.)	80

Table 1: Details of Simulated MPEG Streams

The background traffic generators produce packets of size 576 bytes, with packet inter-arrival obtained using a truncated exponential distribution. The mean of the exponential distribution is chosen such that the background traffic has the desired long-term average rate. For some of our experiments (Section 6.2), the background traffic sources use an exponential-backoff algorithm (multiplicative decrease, additive increase similar to TCP) under congestion.

4.2 Topology

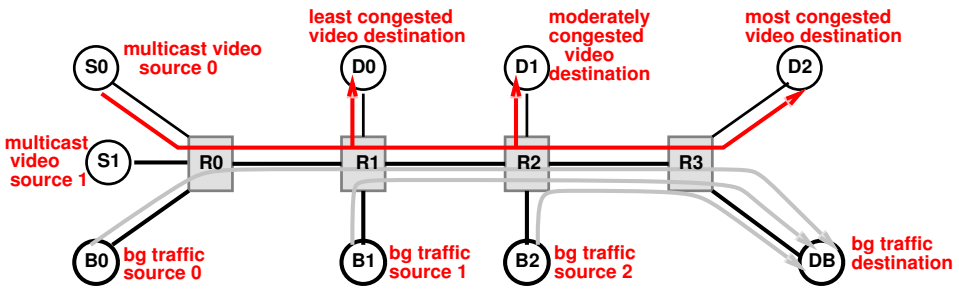

Figure 2: Experimental Topology

The experiments were simulated on the topology shown in Figure 2. There are four routers in the topology: nodes R0 through R3. Node S0 is the source of multicast video to destinations D0, D1, and D2. For multi-source experiments, node S1 also transmits multicast video to the same three destinations. The multicast video is congested at various links between the routers by the background traffic sources (nodes B0, B1, B2). Note that the traffic from node B0 congests all the video destinations, while the traffic from node B1 only affects video traffic to nodes D1 and D2 only. Further, the background traffic from node B2 only affects the most congested video destination : node D2. All background traffic is destined to the background destination BD.

4.3 Evaluation Metrics

We use the following metrics to evaluate the quality of the received streams.

- **Fraction of frames decoded.** We report the number of decodable frames over the total number of frames sent by the sender. This "end-user centric" metric takes into account that after the decode process, the I-, P-, or B-frames each account for the same time duration of playback (i.e., 1/30-th of a second). Thus, the type of the decoded frame, from the end-user's point of view, is irrelevant. Of course, in order for a P- or a B-frame to be decoded, all the frames they depend on must also be received and decoded.

- **Fraction of data received.** We also note the usual metric of fraction of data received over the amount sent. In case of hierarchically encoded media, this metric may not correlate well with fraction of data that is actually useful.

- **Useless data received.** As we have noted, some data may be received that form partial frames, or frames that cannot be decoded. We report the fraction of data received but is useless at the receiver. This is a "network centric" metric as it accounts for wasted bandwidth on the network links.

- **Detailed Results.** We present details of particular runs in the form of decoded frame sequence numbers versus time and decoded frames per second over time.

For all of the reported results (except details of particular runs), each reported data point is a mean of 10 separate runs of length 30 seconds each. The cyclic order of frames in MPEG account for the (relatively) quick convergence and tight confidence intervals shown in the results.

5 Results

The results that follow fall in two broad categories:

- Results that show performance at all three video destinations (D0, D1, D2). For each destination, the results show the effect of network-based adaptation and of receiver-based adaptation. For these results, the plots have three distinct "regions" corresponding to the performance of each destination, commensurate with the bandwidth on the path from the source to the receiver.
- To preserve legibility and space, for a subset of results, we only present the performance at the moderately congested receiver (D1). In these cases, the performance at the other receivers, in general, follow the same trends.

5.1 Single Video Source

In this series of experiments, we analyze the effect of network- versus receiver-based adaptation for a single video source. Unless otherwise stated, the background sources B0, B1, B2 (in Figure 2) have average rates of 350, 100 and 150 Kbps respectively, and the capacity of all links is fixed at 1000 Kbps. Before examining any results, we analyze the expected behavior at the receivers based on the long-term average rates of the traffic. The background traffic only accounts for 350 Kbps on the link between routers R0 and R1. Since the video rate is 615 Kbps, we note that destination D0 should be able to receive all the video frames. The background traffic from source B1 further congests the link between routers R1 and R2, and the residual capacity of 550 Kbps is only sufficient for destination D1 to receive the I- and the P-frames. Similarly, we expect destination D2 to receive only the I-frames, since the residual capacity on the link between routers R2 and R3 is only 400 Kbps. Of course, this is a simple analysis based upon average frame sizes and data rates at each video layer. Short-term increases in rate will cause router buffers to fill and receivers to experience performance that differs from these expectations. It is precisely these fluctuations in load that we expect the adaptation mechanisms to deal with.

5.1.1 Parameters for Receiver-based Adaptation

Before we present the results for the single video source experiment, we have to choose values for the different parameters for the receiver-based adaptation scheme. In Figures 3 and 4 we investigate the effect of changing the values of the high and low watermark parameters in the receiver-based scheme. Figure 3 shows the fraction of transmitted frames decoded at each receiver for three values of the high watermark (HW=0.8, HW=0.9, HW=0.99) as the value of the low watermark ranges from (0 ... HW). Note that when the low watermark value is 0, the receivers never drop a layer.

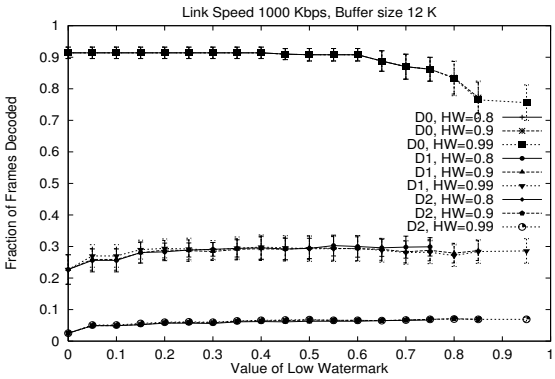

Figure 3: Single video source, varying high, low watermark

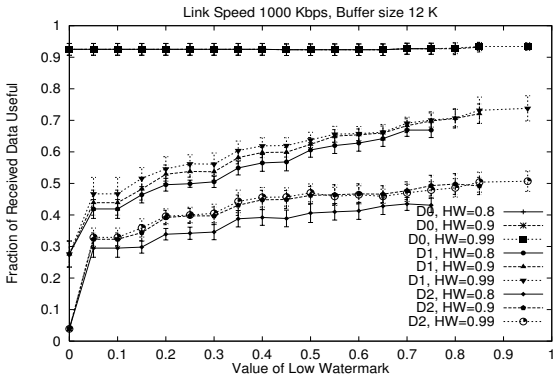

Figure 4: Single video source, varying high, low watermark

We see in Figure 3 that for all three receivers, the value of the high watermark does not make an appreciable difference with respect to the number of decodable frames received. However, as the low watermark increases, the uncongested receiver (D0) drops layers more readily and performs worse. The congested receivers (D1, D2), however, perform marginally better as the low watermark increases.

The reason for the variable sensitivity across receivers is shown in Figure 4, which gives the fraction of received data that is useful (i.e., is part of a decodable frame) for the same values of the high and low watermarks. In this case, neither the high nor the low watermark affects the useful data received at the uncongested receiver. However, for the congested receivers, the values of both the high and the low watermarks make a difference. More data is wasted for lower values of the high watermark, since the destination is more likely to add a layer even when the path may be congested. The fraction of received data that is useful steadily increases as the value of the low watermark is increased. As will be shown shown in detailed results (Figures 7 and 8), the effect of receiving data that cannot be used to decode frames is particularly severe for congested receivers.

In summary, a higher value for the high watermark tends to be preferred. However, the choice of low watermark is more complicated. Congested receivers prefer a higher value of the low watermark, while uncongested receivers prefer a lower value. For our experiments, we choose a high watermark of 0.90 and a low watermark of 0.85.

5.1.2 Performance of Network- and Receiver-based Adaptation

Figure 5 shows the fraction of frames decoded at each receiver (D0, D1, D2) for network- and receiver-based adaptation when the link speed is fixed at 1000 Kbps. The network-based adaptation curves are labeled "GL" for GOP-level discard; the receiver-based adaptation curves are labeled "Layering". Further, we also evaluated the case in which both the network adapts using the GOP-level discard scheme, and the receivers adapt by adding and deleting layers. This hybrid scheme (not shown) performs within 5% of GL in all cases.

For the congested receivers (D1 and D2), the network-based adaptation consistently delivers 2-3 *times* more decodable frames than layering. In fact, the performance of the most

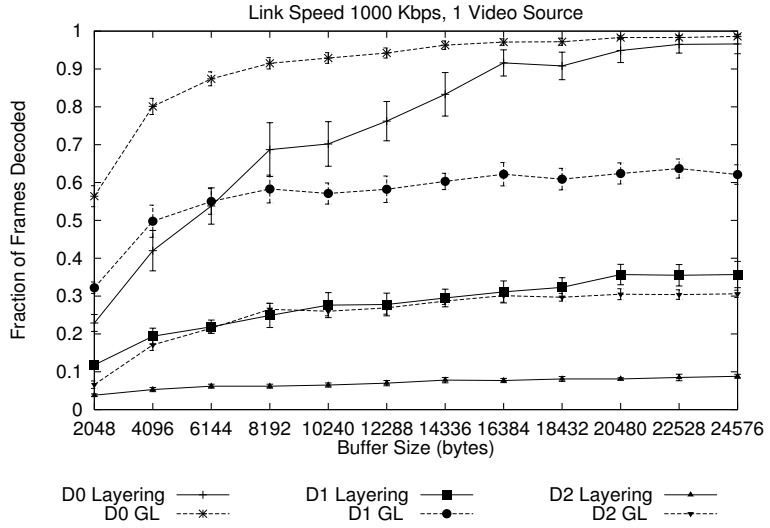

Figure 5: Single video source, varying buffer size, 1000 Kbps Link

congested receiver (D2) in case of the network-based scheme is comparable to the performance of the moderately congested receiver (D1) under layering. We note that none of the schemes perform well with a very small amount of buffering at the output queues (less than 4KB). In case of the least congested receiver (D0), the network-based scheme has far better performance at smaller buffer sizes. This is because the receiver-based adaptation "over-reacts" to transient congestion and drops the B-layer, which accounts for 60% of all the frames. As the buffer sizes are increased, the network is able to absorb the transient bursts, and the performance of receiver D0 under receiver-based adaptation approaches 100% goodput. However, increasing buffers do not help in case of long-lived congestion as is seen in the case of the congested receivers (D1 and D2).

In other experiments (not shown), we increase the link speed to 1100 Kbps. This causes the long-term average load on the links to be about 90%. Under these conditions, receivers D0 and D1 both perform well using receiver-based adaptation, receiving more than 90% of all frames using a reasonable amount of buffering. However, the network-based scheme still outperforms the receiver-based scheme. For receiver D2, the network-based scheme is able to successfully transmit more than 4 *times* as many decodable frames for all buffer sizes. Similar trends were observed when the link speed was set to 900 Kbps (i.e. permanent overload in the network).

5.1.3 Further Variation in Background Traffic Rates

In Figure 6, we vary the rate at which the background source 0 (B0) generates traffic. By virtue of the location of node B0, a change in rate affects all the destinations. In this experiment, we simulate various underload and overload conditions. (Note that congestion *decreases* as one goes from left to right on the x-axis.) Along with the GL and Layering schemes, we also report the results for simple Tail Drop at the congested routers. That is, no adaptation occurs; if a

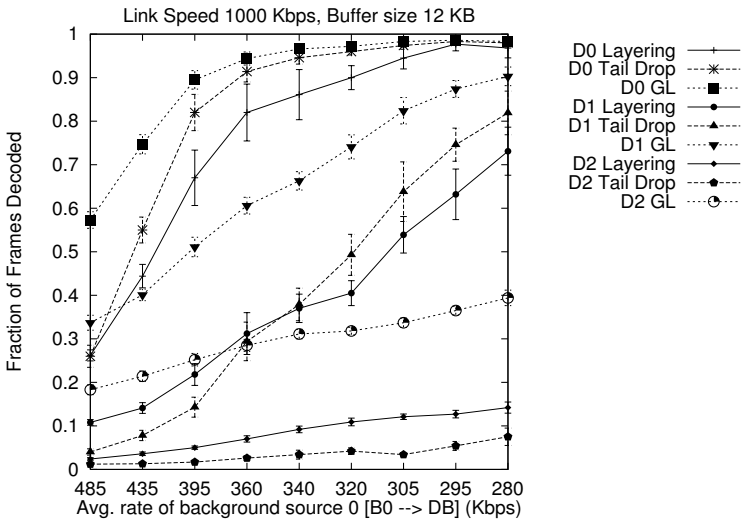

Figure 6: Single video source, 1000 Kbps link, varying background traffic from B0 $\rightarrow$ DB

packet encounters congestion, it is discarded.

Once again, the network-based scheme outperforms the receiver based scheme (Layering). However, an interesting effect is seen for the Layering and the Tail Drop schemes. For the unloaded receiver (D0), Tail Drop *outperforms* the Layering scheme. This is because the layering scheme is prone to drop a layer in response to transient short-lived congestion, even though there is enough long-term capacity. The first layer dropped is always the B-frames layer that constitutes 60% of all the frames. Each time this layer is dropped, the Layering scheme can only receive 40% of all the frames until it decides to add the B-frame layer again. This effect may be mitigated with more complex algorithms for layer dropping. Note that as the congestion reduces, all four schemes perform well, providing over 90% of the frames.

For the moderately congested receiver (D1), the network based schemes outperform the receiver-based schemes, achieving up to three times better performance under severe congestion. Once again, under less congestion, the Tail Drop scheme outperforms the Layering scheme. As congestion is increased, the difference in performance between Layering and Tail Drop decreases, and the two curves cross over at a background rate of about 340 Kbps. As expected, under high congestion, the Tail Drop scheme performs worse than layering, and the difference increases as congestion worsens. The layering scheme is able to adapt to congestion by (correctly) discarding the appropriate layers while the Tail Drop scheme is faced with indiscriminate packet loss, and the ensuing degradation to the received stream. A detailed analysis of the received streams shows that the Layering scheme often transmits a larger fraction of decodable bits, compared to Tail Drop. However, even while transmitting less useful data, the simple Tail Drop is able to outperform Layering if the congestion is not severe.

For the heavily congested receiver (D2), the network schemes outperform the receiver based schemes by at least a *factor of 3*. Further, for the reasons mentioned above, Layering

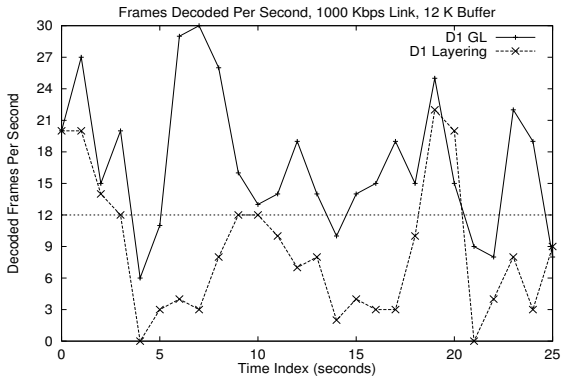

Figure 7: Single video source, decoded frames per second at destination D1

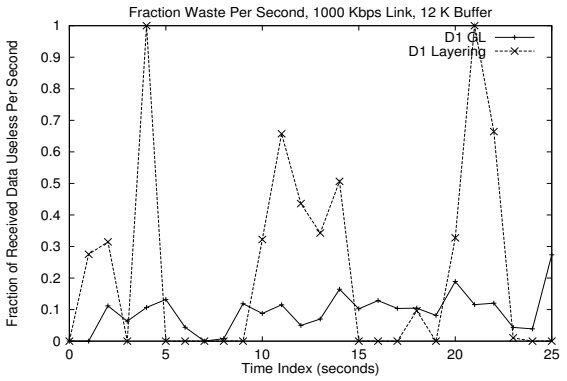

Figure 8: Single video source, fraction of data wasted per second at destination D1

always outperforms simple Tail Drop at this receiver.

5.1.4 Detail Evaluation of Received Stream

In Figures 7, 8 and 9 we investigate, in detail, the received stream for one run of the one video source experiment with the buffer size fixed at 12 Kbytes and the link capacity fixed at 1000 Kbps. The background traffic was generated using the values given in Section 5.1. In these plots, we only analyze the stream received at the moderately congested receiver (D1). These results are representative of the trends observed at other receivers.

Figure 7 shows the number of frames decoded at receiver D1 for the first 25 seconds of the simulation. Note that node D1 is congested by both sources B0 and B1 and hence, on average can only receive the I- and P-frames (i.e., 12 frames per second). The network-based scheme achieves over nine frames per second for about 96% of the interval, and over 12 frames per second for 75% of the interval. The minimum performance is 6 frames per second. Thus, on average the GL scheme transmits well above the nominal 12 frames per second. On the other hand, the Layering scheme spends one-third of the time below three frames per second, and about 70% of the interval strictly below 12 frames per second. Hence, the perceived quality is nearly always worse using the layering scheme.

Figure 8 shows the fraction of bits received that could not be used to decode a frame because other required data for decoding was missing. This plot provides some insight into the rather poor performance of the Layering scheme in the previous plot: when the decoded frames per second metric is low, it is because the wrong bits are being transmitted on the congested link. Lastly, we note that the GL scheme consistently transmits about 10% useless data. This is because our implementation does not buffer a frame before transmitting. In some cases, the beginning packets of a frame have already been transmitted when the middle/end of the frame encounters congestion and has to be discarded. In these cases, the receiver has to discard the received frame fragment since it is not decodable.

Figure 9 shows the actual frame sequence numbers and types that are decoded at destination D1, and is annotated with the layer additions and drops for the Layering scheme.

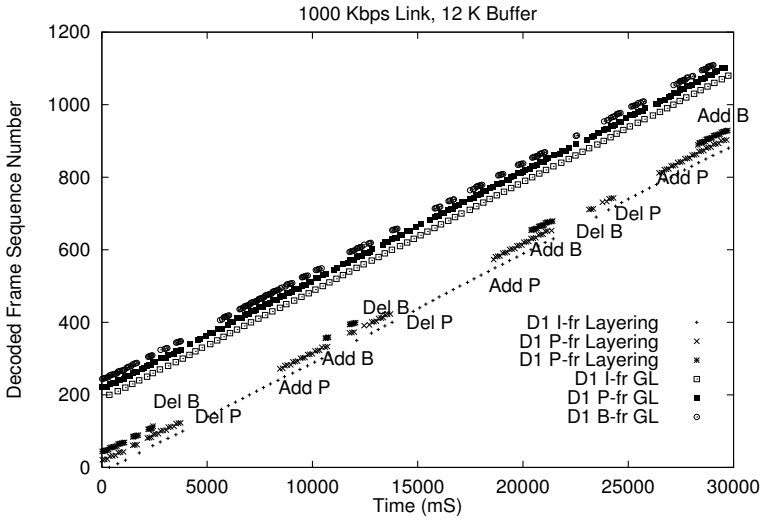

Figure 9: Decoded Frames over time at destination D1

To facilitate readability, the P- and B-frame sequence numbers are offset by 10 and 20 respectively, in the plot. Further, the entire GL stream sequence is offset by 200. This result clearly shows the reasons for the sub-par performance of Layering. The Layering scheme has to continuously probe the conditions in the network in order to get the maximum number of frames. In our case, as soon as the B-frame layer is added, the receiver is doomed to indiscriminate losses (as the capacity on the link between R1 and R2 exceeds 100%). This explains the periods when all data that gets through is useless, and the frames decoded per second goes to zero. On the other hand, the GL scheme is able to accommodate all the P-frames and is also able to send an appreciable number of B-frames. This is because the congestion rates are variable, and the majority of B-frames are very small (less than 1Kbyte). We also note that due to transient congestion, there are times when even the GL scheme is not able to transmit P-frames (and this leads to less than 12 frames decoded per second). These snapshot results show the value of placing application-specific adaptation precisely at the congestion point *inside*, as compared to the probe-and-adjust mechanisms of receiver adaptation.

5.2 Multiple Video Sources

In this section, we consider cases in which there are two video sources (at S0 and S1), each of which generate 615 Kbps MPEG video. The link speed was fixed at 1500 Kbps, and the average rates of background traffic from B0, B1, B2 were set to 250, 170, 320 Kbps. These rates are chosen such that on average, there is enough bandwidth on the link between R0 and R1 so that receiver D0 can receive both streams. Further, the extra 170 Kbps from source B1 congests the path to D1 such that, on average, D1 can only receive the I- and the P- frames from streams. Finally, there is only enough bandwidth on the path to D2 such that it can, on average, only receive the I-frames from both the sources.

5.2.1 Multiple Equal Weight Sources

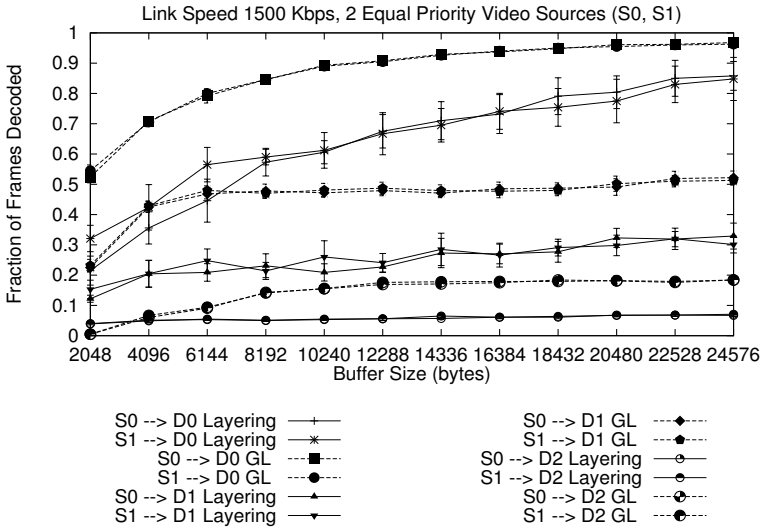

Figure 10: Two equal priority video sources, varying buffer size, 1500 Kbps link

Figure 10 shows the case when both the sources S0 and S1 are considered equal priority at each receiver, i.e., each receiver tries to maximize the number of frames received from each source. In this case, both adaptation schemes treat the sources independently. In effect, each stream acts as congestion for the other.

Figure 10 shows the fraction of frames decoded for each source at each receiver for the network-based adaptation scheme (GL), and the receiver-based adaptation scheme (Layering). We see that the number of frames decoded for each source under the GL scheme is extremely close, and intuitively, compared to Figure 5, larger amounts of buffering is required to achieve the same fraction of decoded frames. As in the single source case, for congested destinations (D1 and D2), the network-based adaptation outperforms the receiver-adaptation schemes. Like the single source case, the gains for the congested receivers, are independent of buffer size. We also note that there is significantly more variablity in the fraction of frames decoded (as shown by the larger confidence intervals) in case of layering. This is because of the effects of running multiple simultaneous layering algorithms at each receiver. Recent work on layering schemes have produced algorithms in which receivers "learn" from neighbor's join and leave experiments [13]. Though we did not implement these algorithms in our experiments, we expect such algorithms to reduce the observed variability in frames decoded for different sources.

5.2.2 Multiple Prioritized Sources

In the following experiments, we consider the case of multiple sources with different priorities. In our case, source S0 is deemed more important than source S1. In what follows, we extend the GL and Layering schemes to handle multiple priorities as follows:

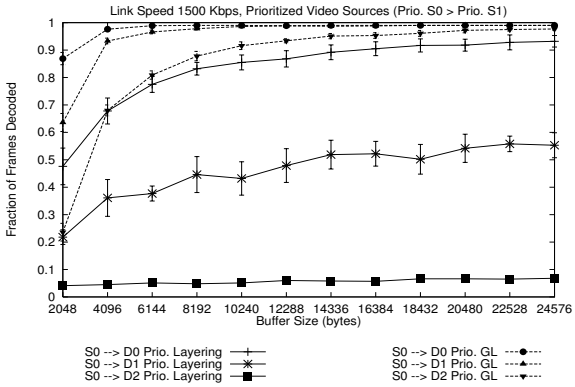

Figure 11: Prioritized video sources, varying buffer size, 1500 Kbps link

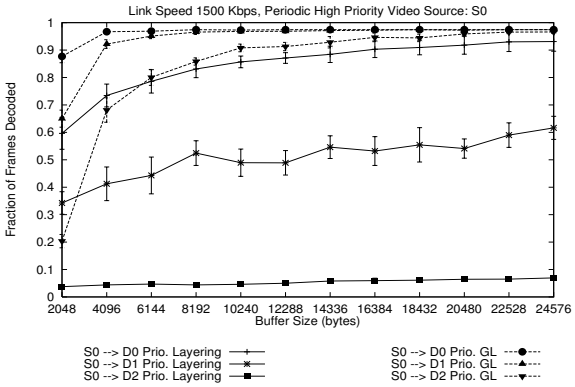

Figure 12: Periodic high priority source, varying buffer size, 1500 Kbps link

- **Priority GL.** A flow priority field is added to the application-level header for each video packet. If a packet from a high priority frame is congested, then the priority discard algorithm may evict packets of lower priority frames in order to accommodate the high priority packet. Further, for any packet discarded, all its associated packets (other packets of the frame and other frames that may depend on this packet) are also discarded. It is interesting to note that it is not clear how a prioritized discard algorithm can be implemented at a router without explicit access to packets already enqueued.

- **Priority Layering.** We extend the regular layering algorithm as follows: if a high priority flow receives a "poor" quality stream over the last 2 seconds (i.e. decodable fraction of bytes less than the low watermark), then if possible, layers of all *lower* priority streams are deleted before deleting any layer from the higher priority scheme. The issue of handling multiple priority streams at receivers is currently a topic of considerable research, and we acknowledge that there may be more intricate ways to extend the simple layering scheme than what we have implemented that provide better performance.

Figure 11 corresponds to the scenario in which both the high and low priority source runs for 30 seconds. We report the fraction of frames decoded for each source, destination pair. In each case, we consider the prioritized version of the network adaptation (Prio. GL), and receiver adaptation (Prio. Layering). For the Prio. GL algorithm, Figure 11 shows that a modest amount of buffering (approx. 10Kbytes) is enough for *all* receivers to decode over 90% of the high priority frames. On the other hand, the layering experiments of the *lower* priority stream in the receiver based adaptation scheme causes more buffer space (18 Kbytes) to be required before the uncongested receiver (D0) can receive 90% of the high priority stream. Note that for this amount of buffer space under network-based adaptation, receiver D0 receives over 90% of the *lower* priority stream along with 100% of the high priority stream. Unfortunately, no amount of buffering is enough for the congested receivers under prioritized layering to receive more than 50% of the high priority stream. In fact, receiver D2 is not even able to receive 10% of the high priority stream. This is because the join experiments of the lower priority stream continuously disrupt the high priority stream by overloading the links

between the high priority sender and the receiver. In other experiments, in which the link speed is increased to 1650 Kbps, receiver D1 is able to receive 90% of the high priority stream with 20 Kbytes of buffer at each router, while receiver D2 peaks at about 15% of the high priority stream (with 12 Kbytes buffer at each node).

5.2.3 Periodic High Priority Source

The situation is similar in the next experiment, in which the high priority source is periodic. In this experiment, the high priority source transmits for ten seconds each starting at 5 and 20 seconds. Figure 12 shows the fraction of high-priority frames decoded. The result is nearly identical to the continuous high priority source — the active mechanisms achieve more than 90% goodput for the high priority source with about 12 Kbytes buffer at each router while Layering requires 18 Kbytes to get the same goodput for destination D0. Once again, destinations D1 and D2 are not able to decode more than 65%, 9% of the high priority stream, respectively. Figure 13 shows the decoded frames sequences at destination D2 for one particular run for each adaptation scheme. Once again, the plots have been offset to enable readability. The problems with layering are evident as soon as the high priority stream begins transmission (5th second). As all the layers for the low priority stream was being received, there is a period of uncontrolled loss when nothing is decoded for either stream. Then the adaptation takes effect and the lower priority layers are dropped in quick succession. As soon as the lower priority stream performs a join experiment (approximately 12 seconds), the link is overloaded, and the receiver encounters uncontrolled congestion. The same pattern repeats the next time the high priority source transmits. The network based adaptation is simple and nearly perfect. As the high priority stream starts, the low priority stream is "thinned" and all the frames of the high priority stream are received. There is a period of transient congestion around the 23rd second in which even the high priority source loses an I-frame (and the associated GOP).

5.3 Alternate Definitions of Decodable

Our experiments have assumed that all packets in a frame and all of its dependencies must be received in order for the frame to be decodable. We generalize the notion of a frame being decodable as follows: Given that all the layers a particular frame depends on have been decoded, a frame is decodable if and only if a fixed fraction F (or more) of the bits in the frame are received. Further, a layer is decodable if and only if a fixed fraction L of the frames within the layer could be decoded. Consider the following example: Let $F = 0.8$, and $L = 0.75$. Then an I-frame would be decodable if $\geq 80\%$ of the bits in the frame were received. As P-frames only depend on I-frames, a P-frame would be decodable if and only if more than 80% of it were received, given that more than 80% of the I-frame had been received. The B-frames depend on both the P-frames and the I-frames. Thus, for a B-frame to be decodable, more then 80% of it must be received, along with more then 75% of all the P- and I-frame it depended on. Note that in our experiments, our notion of decodable corresponds to L and F both equal to 1.0. If, due to sophisticated framing and coding, each packet could be decoded on its own, then the value of L would be 0.0.

In Figure 14, we show the result of an experiment in which the value of F and L are not 1.0. For this experiment, we fix the value of L to be 0.6, and vary the value of F along

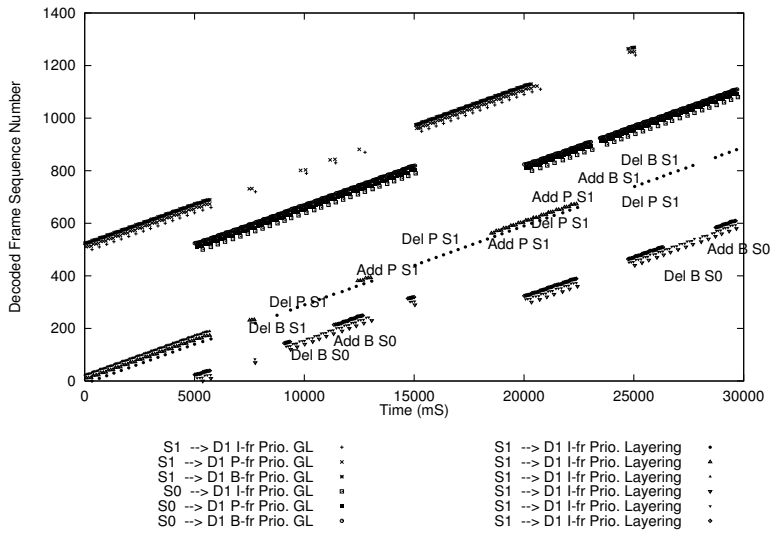

Figure 13: Periodic high priority source, Received Frame Sequence

the x-axis. The results show the fraction of frames decoded at the moderately congested receiver under the single sources scenario. Recall that this link speed corresponds roughly to an average offered load of about 95% on the links between the routers. The result also includes a new form of network adaptation: the routers employ a predefined static priority to discard packets under congestion (SP). In our case, packets carrying I-frames have strictly higher priority than packets carrying P-frames. Further, the packets carrying B-frames have the lowest priority of all. Thus, if an "I-packet" encounters congestion, all B- and P- packets may be discarded to ensure the I-packet can be accommodated. However, this scheme is not cognizant of frame boundaries etc., and may transmit fractions of frames.

The result in Figure 14 shows that if the decoder is sophisticated, ($F < 0.85$), then for $L = 0.8$), Layering outperforms the network based schemes. Further, there is a region where the static priority scheme outperforms the GL mechanism. This is because the network based mechanisms are engineered to transmit more I-frames, and in case of GL, more complete frames. Thus, the GL mechanism does not degrade appreciably when the amount of the frame required in order for it to be decodable reaches 100%. In case fractions of frames are considered useful, Layering (and even simple tail-drop) is the best scheme as every byte that reaches the destination is deemed useful. Thus, we see that an extremely "smart" end-system can compensate for the lack of sophisticated mechanisms within the network.

6 Discussion

We have presented a set of network-based techniques for reducing the effect of congestion in a best-effort multicast video distribution. Our results show that over a wide range of operating points, these techniques are superior overall to the layered receiver-driven approach, in terms of

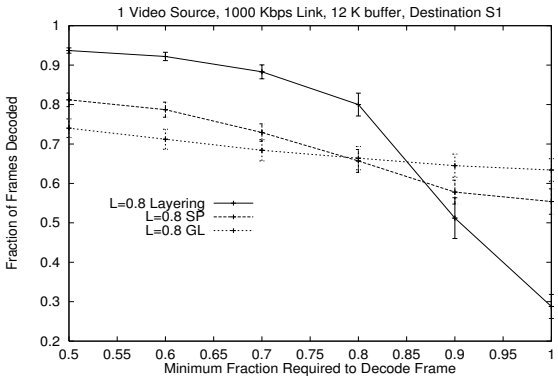

Figure 14: Decoder efficiency

both end-user quality (as measured by fraction of total MPEG frames decodable), and network efficiency (as measured by the fraction of bytes delivered that were useless). However, these packet discard and queue manipulation strategies require supporting mechanisms in the nodes of the network (at least in those nodes where packets may be dropped because of congestion), while the end-system-based techniques do not. For this reason we consider some practical concerns related to these mechanisms, and argue that they are not insurmountable.

Concerns about these kinds of mechanisms fall generally into the categories of scalability, complexity, and, more generally, network "citizenship". The *scalability* issue is simply that these techniques require per-flow state. There is currently no consensus as to whether any kind of per-flow mechanism can be practically realized in the backbone of the network. One response is that the per-flow state required for these techniques is on the order of a few bytes per flow per outgoing queue. This is approximately the same amount of state as is required for multicast *routing*, according to present schemes.[4] Thus if multicast routing can be supported, the additional cost to implement the state required for these techniques should not be unreasonable.

The issue of *complexity* focuses on the fact that the operations required for these techniques —e.g., examining the contents of the queue in order to evict packets matching a particular pattern— are somewhat complex, and must be done at line speeds. In the following section, we present results of network-based adaptation schemes that are more "light-weight" in their complexity.

6.1 Alternate Models of Network-based Adaptation

The GL adaptation scheme in Section 5 incorporated all the three mechanisms for discard discussed in Section 2.1. In Figure 15, we set the link speed to 1000 Kbps, and present results for the single source scenario when one or more of these mechanisms are not implemented at the routers. (This result is representative of the performance noticed at the other receivers). To facilitate comparison, we have included the GL result (which incorporates all three mechanisms) from Figure 5 in this figure as well. Figure 15 contains results for two additional schemes: "Discard-on-Arrival Only" and "Discard-Own + Discard-On-Arrival". Recall that

[4] State required for multicast routing is $O(n)$, where n is the number of flows.

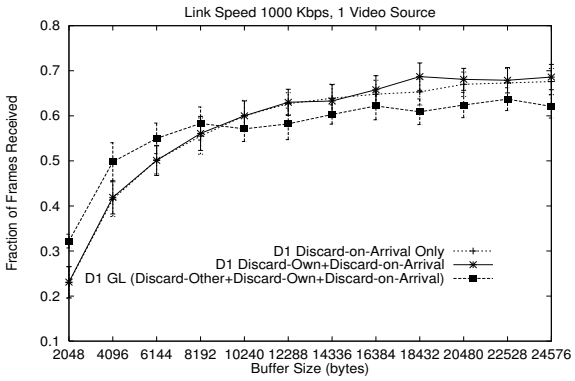

Figure 15: Single video source, varying buffer size, 1000 Kbps Link, Alternate Network-based adaptation schemes

using the Discard-Own mechanism, when a packet encounters congestion, it along with all the packets for *its* frame is discarded. No packets other are discarded from the queue in order to create space for this frame. Under the "Discard-on-Arrival Only" scheme packets that encounter congestion are discarded, the output queue is not manipulated at all, and state is kept such that subsequent packets that depend on the current (discarded) packet are discarded before entry into the queue.

Interestingly, these two (simpler) mechanisms perform extremely well compared to GL (for all receivers), and for some buffer sizes are able to *outperform* GL. Our detail analysis (not shown) of the individual streams show that these schemes are able to transmit more "total" decodable frames as they transmit more B-frames. However the number of I-frames and P-frames sent using the GL scheme is more than either the Discard-Own+Discard-On-Arrival or the Discard-On-Arrival scheme (as the GL scheme selectively deletes B-frames to transmit congested I- and P-frames. As noted before, if an I-frame cannot be decoded, the receiver is not able to decode any other frame for 1 GOP time period (1/3 second). Thus, the perceived quality of the video at the receiver shows less variation using the GL scheme as compared to the other schemes. Note that even the "Discard-on-Arrival Only" scheme, which does not use any potentially costly queue manipulation mechanism, clearly outperforms the layering schemes (compare to Figure 5). Thus, this result makes a strong case for placing even a very small amount of application-specific state at network nodes.

6.2 Citizenship

Perhaps the most important concern is about the *citizenship* of these techniques—that is, their effect on other traffic in a best-effort network. In particular, TCP backs off under congestion, so that all flows can receive their fair share of bandwidth through the bottleneck. It is well-known that if a TCP connection shares a bottleneck with another flow that does not adapt, or does so according to a different algorithm, the non-TCP flows may end up with an unfairly large share of the available bandwidth. The techniques investigated here —both

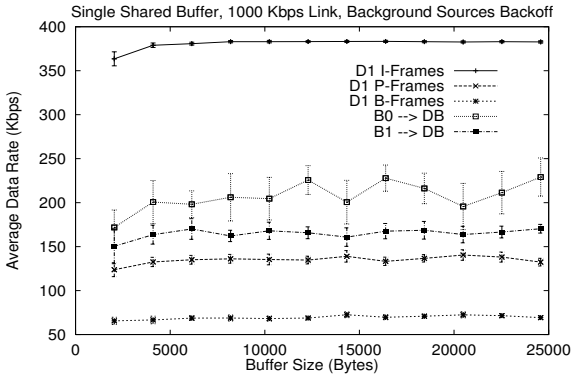

Figure 16: Single video source, varying buffer size, 1000 Kbps Link, Single shared buffer at routers with backoff-sources

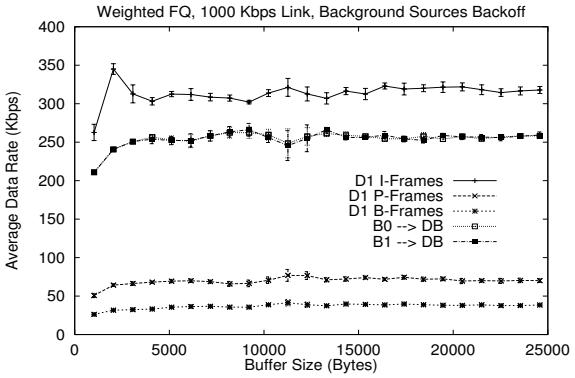

Figure 17: Single video source, varying buffer size, 1000 Kbps Link, Weighted Fair Queuing at routers with backoff-sources

network-based and end-system-based— react to congestion and adapt to available bandwidth, but differently than TCP. Our simulations confirm that a multicast flow of this kind would end up with an "unfair" share of bandwidth in competition with TCP. However, this is a problem for the receiver-driven techniques as well—as it is, potentially, for *any* protocol that adapts differently than TCP.

The best approach to ensuring "fair" sharing of bandwidth among such different kinds of flows seems to be the use of weighted fair queuing [3] or similar techniques. Consider reserving half the bandwidth of an outgoing link for TCP and half for video or other flows that adapt using different mechanisms. This ensures that bandwidth is shared fairly between *type* of flows, while the individual mechanisms ensure that it is shared efficiently (and fairly) *within*. Moreover, the scheduling mechanism need only scale to handle a small number of classes, determined by the number of different protocols in use.

Figures 16 illustrates the concern of citizenship when the background sources back off upon congestion. We use the same topology and source rates as the single source experiment in Section 5.1.2 and report the average data rate available to each sender under the network based adaptation scheme. In case of the video source, we present the individual rates for the I-, P- and B-frames (as perceived by receiver D1). The data rates for B0 and B1 are the rates observed at the destination BD. We note that the data streams back off and the video stream captures more than its share of the bandwidth (greater than 500 Kbps). Further, there is significant variance in the data rates obtained by the data streams. Figure 17 shows the results when the routers perform weighted fair queuing, and each "flow" is weighted equally (all the video layers form one flow and each background source forms one flow). In this case, the data flows get their requisite 250 Kbps each, and the video flow to D1 gets the remaining 500 Kbps. We also ran the previous experiment in which all the background sources formed a single flow (and thus received 500 Kbps total on the link between R2 and R3) and in that case the background sources obtained an average of approximately 170 Kbps each (as there were three contending data sources on the R2 – R3 link).

Finally, we observe that "partial packet discard" and "early packet discard" mechanisms have been effectively implemented in ATM switches, where a similar phenomenon occurs—loss of cells due to congestion causes higher-level data units to become useless, and can reduce goodput to near zero if not controlled.

7 Related Work

Related to our work are efforts in both multicast video distribution and in network-based support for applications. In the area of multicast video distribution, our receiver-based adaptation is based on the main ideas in the Receiver-driven Layered Multicast approach [14]. However, we do not include the more sophisticated approaches to control the frequency of join experiments and to share information about join experiments amongst receivers as described by McCanne et al. [14] or Li et al. [13]. These techniques may allow some improvement in the performance of the receiver-based scheme, however the fundamental challenge remains: receivers have limited information about the presence and absence of congestion in the network. Mechanisms placed in the network avoid join experiments and effectively coordinate receivers.

The Thin Streams approach [16] attempts to mitigate the problems that occur when the streams available to receivers are coarse-grained. The "thick" streams of video layering are divided into "thin" streams, each sent on a separate multicast group. The effect is that a join or leave causes less oscillation in network behavior, since the bandwidth of a stream is reduced. This approach, however, requires more state and overhead in the network to management multicast groups and more computation at the receivers to merge streams. The thin streams encoding could also be used by a network-based adaptation scheme, with natural extensions to the work we describe.

Under the topic of active networking, various research groups are considering ways that state and/or computation in the network might benefit applications. Selected work in this area includes support for reliable multicast [12], protocol deployment [1], and wide-area caching [5]. Other efforts in active networking bear on the feasibility of proposing that network routers perform computation on behalf of user flows.

The existence of an open and programmable network infrastructure is not necessary for the deployment of network-based mechanisms. If sufficient value is added by a mechanism, router vendors will have incentive to include it. Examples of recent mechanisms proposed include packet filtering (to protect end systems from attempts to exploit security holes) and TCP "ack spoofing" (to improve reliability over lossy links) [2].

8 Conclusions

The challenges of multicast video distribution have led researchers to move from sender-based adaptation schemes to receiver-based adaptation schemes. Receiver-based schemes have advantages with respect to scalability and performance, however, receivers are still limited in their ability to detect changes in congestion in a timely manner and receive the best possible quality. We investigate a third location for adaptation, namely in the network.

Using an extensive set of simulations, we demonstrate that network-based adaptation can

yield significant performance gains for multicast video distribution. Improvements are also seen for measures of network efficiency. In addition, we argue that the state and computation requirements of our mechanisms are comparable with current and emerging functionality in routers.

In future work, we plan to explore the performance when a limited set of routers support network-based adaptation. We believe that strategic placement of such routers can preserve most of the benefits seen in this paper.

References

[1] D. Scott Alexander, Marianne Shaw, Scott M. Nettles, and Jonathan M. Smith. Active bridging. In *Proc. of ACM SIGCOMM '97*, Cannes, France, 1997.

[2] Hari Balakrishnan, Srinivasan Seshan, Elan Amir, and Randy H. Katz. Improving TCP/IP performance over wireless networks. In *Proc. of 1st ACM Conference on Mobile Computing and Networking*, Berkeley, California, November 1995.

[3] Joe C. R. Bennett and Hui Zhang. Hierarchical packet fair queueing algorithms. In *ACM Sigcomm '96*, pages 143–156, Stanford, California, August 1996.

[4] Samrat Bhattacharjee, Kenneth. L. Calvert, and Ellen W. Zegura. An Architecture for Active Networking. In *Proceedings of High Performance Networking 97*, 1997.

[5] Samrat Bhattacharjee, Kenneth L. Calvert, and Ellen W. Zegura. Self-organizing wide-area network caches. In *IEEE Infocom'98*, 1998.

[6] Steven L. Blake, Sarah A. Rajala, and Fengmin Gong. Efficient techniques for two-layer coding of video sequences. Technical report, North Carolina State University, Raleigh, NC, November 1993.

[7] J.-C. Bolot, T. Turletti, and I. Wakeman. Scalable feedback control for multicast video distribution in the Internet. In *ACM Sigcomm '94*, 1994.

[8] F. Bonomi and K. Fendick. The rate-based flow control framework for the available bit rate ATM service. *IEEE Network Magazine*, March/April 1995.

[9] J. Enssle. Modelling and statistical multiplexing of VBR MPEG compressed video in ATM networks. In *Proceedings of the 4th Open Workshop on High Speed Networks, IND, Universitt Stuttgart*, page 10, Brest, France, 1994.

[10] Rahul Garg. Characterization of video traffic. Technical Report TR-95-007, International Computer Science Institute, Berkeley, CA, 1995. Available at ftp://ftp.icsi.berkeley.edu/pub/techreports/1995/ tr-95-007.ps.gz.

[11] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM '88*, 1988.

[12] Ulana Legedza, David J. Wetherall, and John Guttag. Improving the performance of distributed applications using active networks. IEEE Infocom'98, 1998.

[13] Xue Li, Sanjoy Paul, and Mostafa H. Ammar. Layered Video Multicast with Retransmission (LVMR): Evaluation of hierarchical rate control. In *Proceedings of IEEE Infocom'98*, San Francisco, CA, March 1998.

[14] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *ACM Sigcomm '96*, pages 117–130, 1996.

[15] D. Wetherall, J. Guttag, and D. L. Tennenhouse. ANTS: A toolkit for building and dynamically deploying network protocols. In *IEEE OPENARCH'98*, San Francisco, CA, April 1998.

[16] L. Wu, R. Sharma, and B. Smith. Thin streams: an architecture for multicasting layered video. In *NOSSDAV'97*, 1997.