A SOURCE ADAPTIVE MULTI-LAYERED MULTICAST ALGORITHM FOR INTERNET VIDEO DISTRIBUTION[1]

Célio Albuquerque[†], Brett J. Vickers[‡], and Tatsuya Suda[†]
[†] *University of California, Irvine. {celio, suda}@ics.uci.edu*
[‡] *Rutgers University. bvickers@cs.rutgers.edu*

Keywords: Video Multicast, Layered Encoding, Adaptive Feedback Control

Abstract– Layered transmission of data is often recommended as a solution to the problem of varying bandwidth constraints in multicast applications. In the case of video multicast, this technique encodes multiple interdependent layers of video at arbitrary target rates in order to address heterogeneous bandwidth constraints between the source and multiple receivers. However, multi-layered encoding alone is not sufficient to provide high video quality and high bandwidth utilization, because bandwidth constraints change over time. Adaptive techniques capable of adjusting the rates of video layers are required to maximize video quality and network utilization.

In this paper we define a class of algorithms known as Source-Adaptive Multi-layered Multicast (SAMM) algorithms. In SAMM algorithms, the source uses congestion feedback to adjust the number of generated layers and the bit rate of each layer. Furthermore, we introduce an end-to-end SAMM algorithm, in which only end systems monitor available bandwidth and report the amount of available bandwidth to the source. Using simulations which incorporate actual multi-layered video codecs, we demonstrate that the proposed SAMM algorithm exhibits better scalability and responsiveness to congestion than algorithms which are not source-adaptive.

1 INTRODUCTION

The simultaneous multicast of video to many receivers is complicated by variation in the amount of bandwidth available throughout the network. The use of layered video is commonly recommended to address this problem. A multi-layered video encoder encodes raw video data into one or more streams, or layers, of differing priority. The layer with the highest priority, called the *base layer*, contains the most important portions of the video stream, while additional layers, called *enhancement layers*, are encoded with progressively

[1] This research is supported by the National Science Foundation through grant NCR-9628109. It has also been supported by grants from the University of California MICRO program, Hitachi America, Standard Microsystem Corp., Canon U.S.A., Novell, Tokyo Electric Power Co., Nippon Telegraph and Telephone Corp. (NTT), Nippon Steel Information and Communication Systems Inc. (ENICOM), Fujitsu, and Matsushita Electric Industrial Co., and Fundação CAPES/Brazil.

lower priorities and contain data that further refines the quality of the base layer stream. For each unique bandwidth constraint, the encoder generates an enhancement layer of video, thereby ensuring that all receivers obtain a quality of video commensurate with their available bandwidth.

However, multi-layered encoding of video is not sufficient to provide ideal video quality and bandwidth utilization. Due to competing network traffic, bandwidth constraints change continually and rapidly. To improve the bandwidth utilization of the network and optimize the quality of video obtained by each of the receivers, the sender must persistently respond to these changing network conditions. It should dynamically adjust the number of video layers it generates as well as the rate at which each layer is transmitted. For the sender to do this, it must have congestion feedback from the receivers and the network.

We define a Source-Adaptive Multi-layered Multicast (SAMM) algorithm as any multicast algorithm that uses congestion feedback to adapt the transmission rates of multiple layers of data. Our previous work [30, 31, 1, 2, 3] has focused on network-based SAMM algorithms, in which it was assumed that network switches were capable of executing complex flow and congestion control algorithms. However, in most existing networks and internetworks, where datagram routing and forwarding are often the only universally shared operations, the existence of such congestion control functions cannot be assumed.

We focus on an end-to-end SAMM algorithm that can be implemented in next generation internets. Prerequisites for its implementation include router-based priority packet discarding and flow isolation via either class-based queueing or fair queueing. In the algorithm, video receivers generate congestion feedback to the sender by monitoring the arrival rate of video traffic, and feedback packets are merged by an overlaid virtual network of feedback merging servers. Network switches or routers are not required to implement flow or congestion control algorithms.

The remainder of this work is organized as follows. Trade-offs between sender-driven and receiver-driven approaches to layered multicast are considered in section 2. The details of the end-to-end SAMM algorithm are described in section 3. An encoder rate control algorithm for adaptive, multi-layered video encoding is presented in section 4. The performance of the algorithm in terms of scalability, responsiveness, and fairness is compared with that of a non-adaptive algorithm in section 5. And concluding remarks are provided in section 6.

2 SENDER-DRIVEN VS. RECEIVER-DRIVEN ADAPTATION

Adaptation to network congestion may be sender-driven or receiver-driven. In a sender-driven algorithm, the source adapts its transmission rate in response to congestion feedback from the network or the receivers. In a receiver-driven algorithm, the source transmits several sessions of data, and the receivers adapt to congestion by changing the selection of sessions to which they listen.

2.1 Background

Sender-driven congestion control for adaptively encoded video was first examined in the context of point-to-point communications. A number of works in this area have proposed algorithms in which information about the current congestion state of the network is passed via network feedback packets to the video source, and the source adjusts its encoding rate in response [17, 18, 23, 27, 19]. These works illustrate the effectiveness of transmitting video using sender-driven adaptation to congestion but do so only for the unicast case.

One of the first examinations of sender-driven congestion control for multicast video was performed by Bolot, Turletti and Wakeman [10]. In their algorithm, the source adaptively modifies the video encoding rate in response to feedback from the receivers. This is done to reduce network congestion when necessary and increase video quality when possible. To prevent feedback implosion, each receiver probabilistically responds with congestion feedback at a frequency which is a function of the total number of receivers. While this algorithm considers the problem of multicast, it uses only a single layer of video, and thus a few severely bandwidth-constrained paths can negatively impact the rate of video transmitted across paths that have more plentiful bandwidth.

The Destination Set Grouping (DSG) algorithm by Cheung, Ammar and Li [11] was one of the first to deal with the problem of heterogeneous bandwidth constraints in multicast video distribution, and it shares features of both receiver-driven and sender-driven approaches. The algorithm attempts to satisfy heterogenous bandwidth constraints by offering a small number of independently encoded video streams, each encoded from the same raw video material but at different rates. The streams are targeted to different groups of receivers, and their rates are adjusted according to probabilistic congestion feedback from each group. However, one important drawback of this algorithm is that the transmission of independently encoded video streams results in an inefficient use of bandwidth.

McCanne, Jacobson and Vetterli proposed the first truly receiver-driven adaptation algorithm for the multicast of layered video [21]. In the algorithm,

known as Receiver-driven Layered Multicast (RLM), the video source generates a fixed number of layers, each at a fixed rate, and the receivers "subscribe" to as many layers as they have the bandwidth to receive. Congestion is monitored at the receivers by observing packet losses. This approach has the advantage that it uses video layering to address heterogeneous bandwidth constraints. However, it limits the receivers to choosing among the layers the source is willing to provide, and in many cases the provided selection may not be adequate to optimize network utilization and video quality. Furthermore, RLM is relatively slow to adapt to changes in the network's available bandwidth. If the background traffic is particularly bursty, the receivers may not be able to adapt appropriately, resulting in degraded utilization and video quality. Extensions and variants of RLM (namely, Layered Video Multicast with Retransmission (LVMR) [20], and TCP-like Congestion Control for Layered Data [29]) have recently been proposed to ameliorate some of these weaknesses.

Another potential solution to the multicast of video to receivers with heterogeneous bandwidth constraints — although it is not sender-driven or receiver-driven — is transcoding [5, 7, 6]. In this approach, a single layer of video is encoded at a high rate by the source, and intermediate network nodes transcode (i.e., decode and re-encode) the video down to a lower rate whenever their links become bottlenecked. While this approach solves the available bandwidth variation problem, it requires complex and computationally expensive video transcoders to be present throughout the network.

2.2 Trade-offs

There are several trade-offs between receiver-driven and sender-driven approaches, particularly for the case of layered video multicast. The first trade-off is the granularity of adaptation. In a receiver-driven algorithm, the source typically generates a fixed number of layers at a coarse set of fixed rates. Hence, if the path to one of the receivers has an amount of available bandwidth that does not exactly match the transmission rate of a combined set of offered video layers, the network will be underutilized and the quality of that receiver's video will be suboptimal. Sender-driven algorithms do not suffer from this problem, because they are able to fine-tune layer transmission rates in response to network bandwidth availability. They can therefore achieve better network utilization and video quality.

Another trade-off arises in the ability of sender-driven and receiver-driven algorithms to respond to rapidly fluctuating background traffic. Video sources using sender-driven algorithms receive a continuous stream of congestion feed-

back from the network, and thus they may adapt to changing bandwidth constraints either by adding a new layer of video or by adjusting the rate of an existing layer. Furthermore, this can be done rapidly, usually within a single round-trip time. Most receiver-driven algorithms, on the other hand, adapt to changing network congestion through a combination of "layer join experiments" and branch pruning, both of which occur at time intervals greater than the round-trip time.

The layer subscription and unsubscription strategies of receiver-driven algorithms also have negative consequences for overall video throughput and loss – consequences that sender-driven algorithms do not share. In most receiver-driven algorithms, receivers perform occasional join experiments, during which they request a new layer of data. If the join experiment creates congestion, packets may be lost and the experiment is considered by the receiver to be a failure. Since receiver-driven algorithms like RLM do not rely on priority discarding, packets from any video layer – even the base layer – may be lost during failed join experiments, causing brief but severe degradation in video quality for some receivers. Receiver-driven algorithms also rely on the receiver's ability to prune itself from the distribution tree of a given layer should there be insufficient bandwidth to support that layer. However, there is a significant "leave latency" associated with the pruning of a branch from a multicast tree. During this time, traffic congestion on the branch may be exacerbated, resulting in greater packet loss and delay for downstream receivers of other flows. In a network environment where bandwidth availability is continually and sometimes severely fluctuating, the effects of join experiments and long leave latencies can result in periods of significant packet loss and, for the case of video, significantly degraded video quality.

Receiver-driven algorithms have the advantage that they are naturally more friendly to competing network traffic than are sender-driven algorithms. Sender-driven algorithms typically send all video data on a single transport layer connection and use priority indications to specify the drop precedence of each layer. This inevitably results in some low priority traffic being sent needlessly down some branches of the multicast tree, only to be discarded further downstream. If this extraneous traffic shares FIFO queues with competing traffic that is adaptive (e.g., TCP flows), then the adaptive flows may experience an unfair degree of discarding or delay within the network. Receiver-driven algorithms do not share this deficiency with sender-driven algorithms, because they send each layer of video in a different flow and allow for the pruning of flows that have no downstream receivers. One way to correct this deficiency of the sender-driven algorithms is to isolate video traffic from other traffic. This can be done by implementing class-based queueing [16] or weighted fair queue-

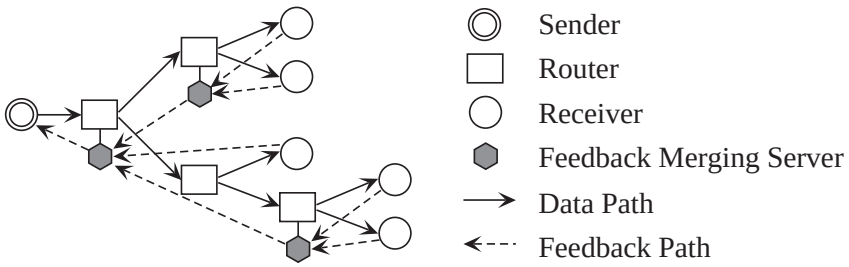


Figure 1: *Network architecture for SAMM*

ing [14, 25] within the routers or switches. There is, however, a non-negligible degree of complexity involved in the implementation of class-based and fair queueing at intermediate network nodes.

3 ARCHITECTURE AND ALGORITHM

In the SAMM paradigm, the sender adjusts its encoding parameters, including the number of video layers it generates and the encoding rate of each layer, in response to a continuous flow of congestion feedback from the network and/or the receivers. In this section, we consider a network architecture capable of supporting this paradigm and a SAMM algorithm in which congestion control is performed on an end-to-end basis with minimal network participation.

3.1 The SAMM Architecture

The network architecture necessary to implement a SAMM algorithm for video consists of four basic components: adaptive layered video sources, layered video receivers, multicast-capable routers, and nodes with feedback merging capability. A sample configuration of this architecture is shown in Figure 1.

3.1.1 Adaptive Layered Video Sources

In a SAMM algorithm, it is assumed that the video source is capable of generating layered video data. There are a number of ways for a source to generate layered video data. For instance, it may simply mark a subset of the video frames as base layer data and the remaining frames as enhancement layer data. Or, the source may coarsely quantize the video stream's frequency coefficients

to produce the base layer and add refinement coefficients to produce enhancement layers. For the purposes of this paper, we will assume sources that adopt the latter approach, since a finer granularity of layer transmission rates can be achieved this way. However, it is important to note that the SAMM architecture does not mandate that any one type of layering to be performed by the video source.

The video source must also participate in the SAMM algorithm being used. This means it must observe congestion feedback arriving from the network and adaptively modify (1) the number of video layers being generated, and (2) the encoding and transmission rates of each video layer.

3.1.2 Layered Video Receivers

Layered video receivers collect layered video data arriving from the source and reconstruct a decoded video image. All video receivers must use a layered video decoder that is compatible with the layered video encoder used by the source. Video receivers also cooperate with the SAMM algorithm by returning congestion feedback toward the source as specified by the algorithm.

3.1.3 Multicast-capable Routers

Routers or switches within the network must, at a minimum, be capable of performing the following functions:

- *Multicast forwarding and routing.* Whenever a packet reaches a branch point in its multicast distribution tree, the router produces one copy of the packet for each branch. The router also builds its multicast routing tables according to a multicast routing protocol such as DVMRP [32], MOSPF [22], CBT [8], or PIM [13], although no specific multicast routing algorithm is mandated by SAMM algorithms.

- *Priority drop preference.* To support layered video transmission, the router must be able to distinguish packets with different priorities. During periods of congestion, routers drop low priority packets in preference over high priority packets.

- *Flow isolation.* To prevent low priority packets from negatively impacting the performance of rate-adaptive flows that share the router's output links, the router isolates SAMM flows from other flows. Examples of mechanisms capable of doing this include class-based queueing [16] and weighted fair queueing [14, 25], although SAMM is not married to any one particular flow isolation mechanism.

- *Congestion control*. For network-based SAMM algorithms, the router must perform the congestion control functions required by the algorithm. Examples of congestion control algorithms that are network-based and may potentially be used as part of a network-based SAMM algorithm include Random Early Detection (RED) [15], the Explicit Proportional Rate Control Algorithm (EPRCA) [4] and others.

3.1.4 Feedback Mergers

Feedback mergers should be deployed to prevent feedback implosion, an undesirable situation in which a large number of receivers consume significant return-path bandwidth by sending feedback packets to a single source. In order to alleviate this problem, feedback mergers consolidate information from arriving feedback packets and route the resulting feedback packets upstream towards the next feedback merger on the path to the source. The idea of designating nodes in the network to alleviate feedback implosion has appeared in a number of other contexts, most notably in the context of reliable multicast [26, 24, 28].

Feedback mergers ultimately form a virtual network overlaid on top of the underlying datagram network as shown in Figure 1. The feedback merging function may be implemented at the source, at routers which have been enhanced to perform the merging function, at dedicated nodes inside the network, and/or at one or more participating receivers. Furthermore, feedback mergers do not have to be present at every branch point in the multicast tree in order to operate properly. Obviously, a larger number of feedback mergers in the network guarantees a greater reduction in the amount of feedback returning from receivers to video sources. However, in realistic scenarios, feedback mergers are likely to be incrementally deployed as the load created by feedback packets becomes a greater issue.

The primary task of the feedback mergers is to consolidate the feedback packets returning from receivers. For each video multicast flow, feedback mergers store the most recent feedback packet arriving from the nearest downstream feedback merger or receiver. A flow's stored feedback packets are merged and routed to the next upstream feedback merger whenever (1) a feedback packet from the downstream feedback merger or receiver that triggered the last merge arrives, or (2) two feedback packets from the same downstream merger or receiver arrive after the previous merge. To prevent the merging of feedback from downstream receivers that have left the multicast distribution, stored feedback packets that have not been updated are removed from the merger after a sufficient time-out interval.

In addition to its simplicity, this merging policy has several attractive properties. First, it does not require feedback mergers to know in advance how many feedback packets are going to arrive from downstream. This is important, because many multicast models (e.g., IP multicast) do not have built-in provisions for determining the membership of a multicast group. Second, the policy allows merged feedback packets to be returned at the arrival rate of the fastest incoming stream of feedback packets. This is also important, since with heterogeneous bandwidth constraints, some receivers may generate feedback at faster rates than others. This is especially true for congestion control algorithms (like the one presented in this paper) that return feedback at a rate proportional to the data arrival rate.

Note that we have not explained how the content of feedback packets is merged, since this is dependent on the congestion control algorithm being used. We leave this discussion for section 3.2.

3.2 End-to-End SAMM Algorithm

In this section we introduce an end-to-end SAMM algorithm, where congestion control functions are performed solely at the source, the receivers, and the feedback mergers. Network routers and switches are not assumed to perform any complex or novel congestion control functions apart from those necessitated by the SAMM architecture. The video source simply adjusts the number of video layers it generates and the encoding rate of each layer in response to a continuous flow of congestion feedback from the receivers. The behavior of the end-to-end SAMM algorithm's receiver is enhanced to compensate for the lack of congestion control functions within the network. The receiver estimates the available bandwidth on the path from the source by monitoring its received video rate and periodically returns feedback packets toward the source.

When a branch of the multicast tree experiences (or is relieved of) congestion, available bandwidth decreases (or increases) on the branch, and the arrival rate of video packets at downstream receivers changes accordingly. Due to this fact, an estimate of the bandwidth available on the path from the source can be obtained by monitoring the rate at which video packets arrive at the receiver. In the end-to-end SAMM algorithm, each receiver monitors the arrival rate of video packets by using Clark and Fang's time sliding window (TSW) moving average algorithm [12].

Typically, the receiver assumes the available bandwidth is equal to the received video rate. However, the actual available bandwidth may be higher than the video arrival rate when the network is under-utilized. In order to exploit the available bandwidth, the receiver may occasionally report a rate that is higher,

Table 1: *Contents of feedback packets used by the end-to-end SAMM algorithm.*

Field	Description	Used in forward feedback packets	Used in backward feedback packets
L	Maximum number of video layers allowed	•	•
N_l	Current number of video layers		•
r_i	A vector ($i = 1, \ldots, N_l$) listing the cumulative rates of each video layer		•
c_i	A vector ($i = 1, \ldots, N_l$) listing the number of receivers requesting each layer in the rate vector r_i		•

by an increment, than the observed arrival rate of video packets. The receiver reports a higher rate whenever there is a change in the observed arrival rate and no packet losses have been recorded in a given interval of time. This allows the source to capture newly available bandwidth in an incremental, and therefore, stable manner.

Table 1 lists the fields contained within each of the feedback packets. When a forward feedback packet is generated, the source stores the maximum number of video layers it can support (L). The value of L depends on the the number of layers the video encoder is able to generate. For example, if the source uses a scalable encoder that can only generate four layers of video (one base layer plus three enhancement layers), then it sets L to 4. The value of L must also be less than or equal to the maximum number of priority levels the network can support.

After receiving a number of video packets, the receiver returns a feedback packet toward the source. The receiver generates a "backward feedback packet" and sets its contents to indicate the desired video rate. It does this by filling the first slot of the backward feedback packet's rate vector (r_1) with its estimated available bandwidth. It also sets the corresponding slot of the counter vector (c_1) to one in order to indicate that only one receiver has requested rate r_1 so far. The backward feedback packet is returned to the nearest upstream feedback merger. Feedback packets are collected and merged by feedback mergers or by the source.

When a feedback merger joins two or more backward feedback packets, it collects the components of the rate (r_i) and counter (c_i) vectors from each

incoming feedback packet and stores them into a local array, sorted by rate. Each entry in the local rate array corresponds to a video rate requested by one or more downstream receivers, while the entries in the counter array indicate how many downstream receivers have requested each rate. Ultimately, the rate values will be used by the source to determine the rates at which to transmit each video layer. After filling the local rate array, the number of entries in the array is compared to the maximum number of video layers allowed for the connection (L). If the number of entries in the local rate array does not exceed L, then the merging is considered complete. However, if the number of entries exceeds L, then one (or more) of the rate entries must be discarded and its counter value added to the next lower entry. To determine which entry (or entries) to discard, the feedback merger attempts to estimate the impact of dropping each listed rate on the overall video quality. This is done through the use of a simple estimated video quality metric.

The estimated video quality metric attempts to measure the combined "goodput" of video traffic that will be received by all downstream receivers. The goodput for a single receiver is defined as the total throughput of all video layers received *without loss*. For instance, suppose a sender is transmitting three layers of video at 1 Mbps each. If a receiver entirely receives the most important first two layers but only receives half of the third layer due to congestion, then its total received throughput is 2.5 Mbps, but its *goodput* is equal to the combined rate of the first two layers, namely 2 Mbps. The goodput is a useful estimate of video quality because it measures the total combined rate of traffic from uncorrupted video layers arriving at a receiver.

As the feedback merger aggregates feedback packets, it attempts to determine the goodput that downstream receivers will observe. The combined goodput G is estimated from the values listed in the rate array and calculated as follows:

$$G = \sum_{i=1}^{N} r_i \times c_i,$$

where N is the number of entries in the local rate array, and r_i and c_i are the rate and counter values for entry i. To determine which entry to remove from the local rate array, the feedback merger calculates the combined goodput that will result from each potential entry removal. The entry removal that results in the highest combined goodput is then removed from the rate array. This process is repeated until the number of entries in the local rate array is equal to the maximum number of layers allowed. The rate and counter array entries are copied into the slots of the merged packet's rate and counter vectors, and the merged packet is transmitted to the next upstream feedback merger. This pro-

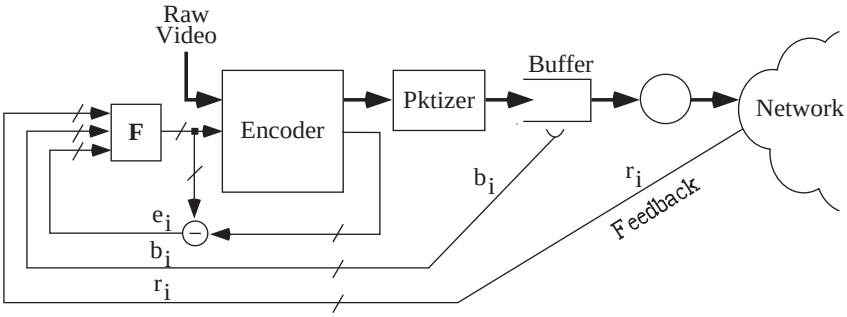


Figure 2: *Video encoder and rate controller*

cess is repeated at each upstream feedback merger until the final consolidated feedback packet arrives at the source. The feedback packet that arrives at the source will contain the number of video layers to generate as well as a list of cumulative rates at which to generate each layer.

The simplicity of the end-to-end SAMM algorithm is its most important feature. By transfering the congestion control functions to the end systems, the end-to-end SAMM algorithm becomes an attractive approach to support video multicast in Internet environments.

4 VIDEO ENCODER RATE CONTROL

Encoder rate control is necessary to ensure that SAMM algorithms can dynamically adjust the encoding rates of several video layers. One possible encoder and rate control architecture is illustrated in Figure 2. The “encoder” block shown in the figure may be any type of layered video encoder (e.g., embedded zero-tree wavelet, MPEG-2, etc.), which accepts uncompressed video information. Uncompressed raw video naturally consists of a sequence of video frames, and we assume the encoder processes frames one block at a time (as in MPEG), where a block is defined as a rectangular component of the frame. The encoder receives a list of target bit rates for each video layer and attempts to produce layered video streams at rates that closely follow the target bit rates. However, since the compression ratio is dependent on video content, it is virtually impossible to produce compressed video at rates that precisely match the target bit rates. Therefore, the encoder returns a list of the rates that it actually generated for each layer of video. This data can then be used to calculate an error term for use in the compression of the next block of video.

The rate control function F in Figure 2 determines the encoder's target bit rates for each layer. It has two purposes: first, to help the encoder produce several layers of video at rates requested by the network, and secondly, to prevent the video buffer from overflowing and underflowing. To achieve these goals, the rate controller determines the target bit rates F_i for layer i as follows:

$$F_i(r_i, b_i, e_i) = r_i - \left[\alpha e_i + \beta \left(\frac{b_i - T_d r_i}{\tau}\right)\right]$$

where r_i is the rate requested for layer i in the most recently received feedback packet, e_i is layer i's encoder rate error from the previously encoded block, and b_i is the number of bits from layer i currently stored in the buffer. T_d is the target buffer delay, which determines the target buffer occupancy at the source. τ is the length of the video block interval. For example, if the raw video is captured at a rate of 10 frames per second and each frame is divided into 10 blocks, then τ is 0.01 seconds. The constants α and β are weighting coefficients. This rate control function adjusts the target bit rates according to the encoding error of the previous block and the current occupancy of the transmission buffer.

After being generated by the encoder, the layered bit streams are packetized and placed into the source buffer in Figure 2 for transmission into the network. Using a simple weighted round robin, the packetizer interleaves packets from each layer according to the layer's target bit rate in order to keep packets from clumping into layers. The packets are then fed into the network at the combined transmission rate of all the layers.

5 PERFORMANCE

This section presents the results of several simulations designed to evaluate the performance of the end-to-end SAMM algorithm under various configurations. These configurations are designed to test the responsiveness, scalability with respect to delay, scalability with respect to the number of receivers, and fairness of the algorithm.

Unless otherwise specified, all simulations assume link capacities of 10 Mbps, propagation delays between end systems and routers of 5 μs, and propagation delays between routers of 100 μs. All packets are the size of ATM cells (53 bytes), and two class-based queues are used at each router hop to isolate background traffic from video traffic. To keep queueing delays minimal, only the amount of buffers necessary to tolerate 10 ms of feedback delay on a series of 10 Mbps links are used. For most simulation models, this works out to approximately 200 packets per router hop for each video flow. A receiver monitoring

interval of 10 ms is assumed, and feedback packets are generated by receivers once for every 32 video packets received. Every router is assumed to be connected to a feedback merging server.

5.1 Responsiveness

One of the most important requirements of a source rate adaptation algorithm is that it be able to respond rapidly to changes in network congestion. This simulation experiment illustrates the trade-offs between source-adaptive and non-source-adaptive algorithms. It also shows the impact of network propagation delay on the responsiveness of the end-to-end SAMM algorithm.

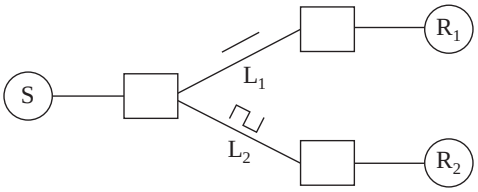
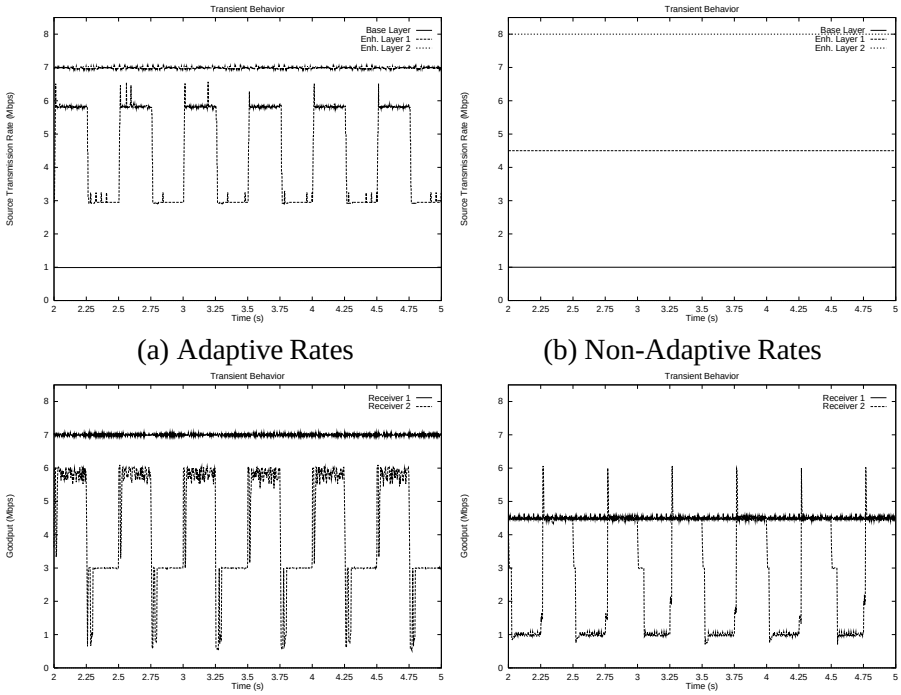


Figure 3: *Simulation model for evaluating responsiveness*

The model shown in Figure 3 is used to evaluate the responsiveness of the algorithm. It consists of one video sender and two receivers. Background traffic is applied on links L_1 and L_2, and two responsiveness experiments are conducted. The first experiment is designed to explore the transient response of the sender to changes in available bandwidth on one of the links. The second experiment explores the impact of the network propagation delay on the effectiveness of the algorithm.

In the first experiment, we apply CBR background traffic at a rate of 3 Mbps to link L_1 and sharply oscillating square-wave background traffic to link L_2. The square-wave traffic oscillates between constant rates of 4 and 7 Mbps over a period of 500 ms and is used to test the responsiveness of the sender to sudden and substantial changes in available bandwidth. As a basis for comparison, we also examine the performance of an algorithm in which the sender is non-adaptive and transmits three layers of video at cumulative rates of 1, 4.5 and 8 Mbps. This set of rates is admittedly arbitrary, but so is any choice of rates for a non-adaptive layered transmission mechanism.



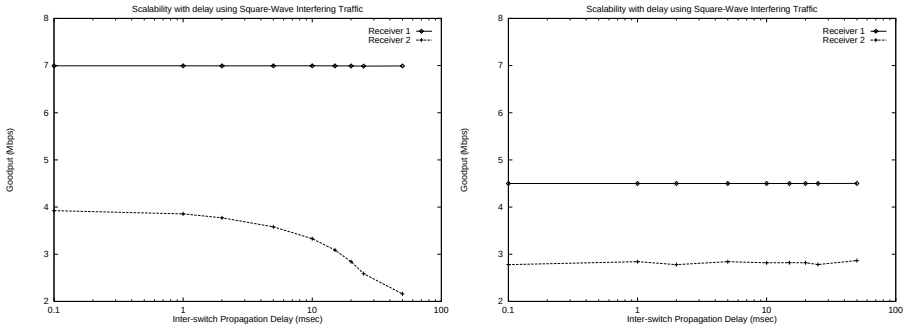
(a) Adaptive Rates (b) Non-Adaptive Rates

(c) Adaptive Goodput (d) Non-Adaptive Goodput

Figure 4: *Responsiveness temporal behavior*

Figure 4 shows the results of the simulation. As expected, the sender adapts the rate of one of its layers in response to the oscillating available bandwidth on link L_2. The remaining two layers are transmitted at cumulative rates of 1 and 7 Mbps, which correspond to the minimum transmission rate and the available bandwidth on link L_1, respectively. Note that the sender responds quickly to the square-wave traffic oscillations, usually within 10 milliseconds (the length of the receiver monitoring interval). The small spikes in the transmission rates are observed due to occasional overestimations of the available bandwidth by receiver R_2. For the purpose of comparison, Figure 4(b) plots the cumulative transmission rates of each layer for the non-adaptive case.

The receiver goodputs for the adaptive and non-adaptive mechanisms are shown in Figs. 4(c) and 4(d). Recall that video goodput is defined as the total throughput of all video layers received *without loss* during a block transmission interval. Clearly the SAMM algorithm produces better goodput than the non-adaptive scheme due to its ability to adjust encoding behavior based on network congestion feedback. Although receiver R_2 experiences degradations of goodput during downward transitions due to buffer overflow, they are brief and the overall goodput levels are desirable. In contrast, the goodput of the non-adaptive mechanism suffers significantly from its inability to take the current state of the network into account.



(a) Adaptive Goodput (b) Non-Adaptive Goodput

Figure 5: *Responsiveness scalability with delay*

In the second experiment, we explore the impact of propagation delay on the goodput. We apply CBR background traffic on link L_1 and square-wave background traffic with a period of 200 ms on link L_2. The background traffic transmission rates are the same as for the first experiment. Propagation delays between routers are varied from 0.1 to 50 msec, and each simulation is run for

60 simulation seconds.

The average goodput delivered to each receiver is plotted in Figure 5. As propagation delay increases to the order of magnitude of the network transition interval, the average goodput delivered to receiver 2 by the SAMM algorithm drops almost linearly. This is due to the fact that as the propagation delay increases, the sender uses increasingly stale congestion feedback to adjust its layer transmission rates. Despite this drawback, the SAMM algorithm generally produces better goodput than the non-adaptive mechanism for both receivers and nearly all delays. The only exception is the goodput at receiver 2 for very high propagation delay ($>$20 ms).

5.2 Scalability

Scalability is perhaps the most important performance measure of any multicast mechanism. Multicast datagrams can reach dozens or even hundreds of receivers, each with varying bandwidth constraints. It is therefore important to understand how a multicast mechanism performs as the number of receivers grows.

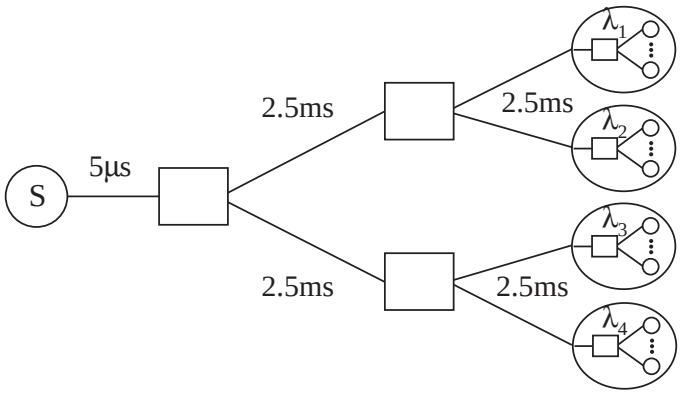


Figure 6: *Simulation model for evaluating scalability*

The network model shown in Figure 6 consists of one video sender, four groups of receivers, and seven routers. Within each receiver group, the number of receivers can be varied between 2 and 32. Independent background traffic streams are applied to each leaf link, and the traffic loads are divided into four heterogeneous groups ($\lambda_1 = 2$ Mbps, $\lambda_2 = 4$ Mbps, $\lambda_3 = 6$ Mbps, $\lambda_4 = 8$ Mbps). Background traffic is generated by a 10-state Markov-Modulated

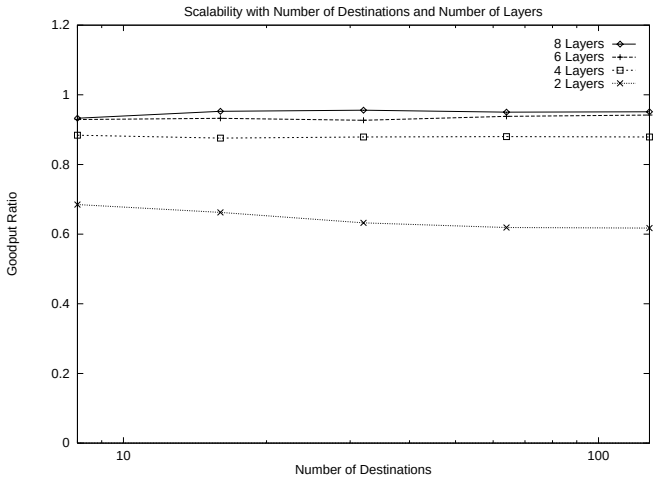


Figure 7: *Average goodput ratio for all receivers vs. Number of receivers*

Poisson Process with state transition rates of 100 $sec^{-}1$. This traffic model captures the superposition of 10 on-off, interrupted Poisson processes and is generally much burstier than a simple Poisson process.

We first examine the performance of the SAMM algorithm as the number of receivers increases in Figure 7. The maximum number of video layers is varied from 2 to 8 in this figure. The goodput ratio is defined as the fraction of the available bandwidth used to transport uncorrupted video layers. To calculate the goodput ratio, the combined rate of video layers fully received by all receivers is divided by the total amount of bandwidth available to all receivers. These results reveal that the SAMM algorithm scales well with the number of receivers. They also illustrate the expected result that video goodput (and thereby video quality) can be improved by increasing the maximum number of layers generated by the sender.

In the second scalability experiment we encode and decode actual video sequences and transmit them through the simulated network shown in Figure 6. For this experiment we use an embedded zero-tree wavelet encoder to generate multiple layers of video from a raw video sequence. The raw video sequence we use is the Academy Award winning short animation *Wallace & Grommit*. The number of multicast receivers is varied between 8 and 128, up to 4 video layers are used, and the background traffic used in the first scalability experiment is reapplied to the leaf links.

Figure 8 plots the average peak signal-to-noise ratio of the decoded video

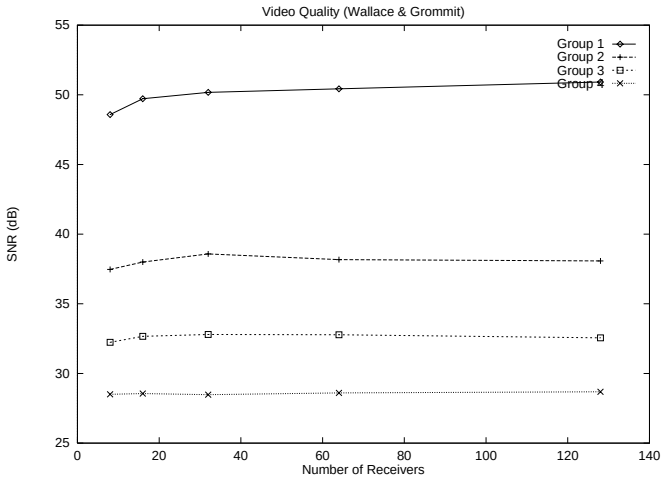


Figure 8: *Average signal-to-noise ratio for all receivers vs. Number of receivers*

sequence for a sampled receiver from each receiver group. (The peak signal-to-noise ratio is a measure of the video quality. The larger the value, the lesser the distortion. It is calculated by comparing the original and the received video image.) The video quality at each receiver remains relatively flat as the number of receivers increases, confirming that the SAMM algorithm is scalable. Furthermore, the quality of video obtained by a receiver is determined by the amount of bandwidth available to it, just as expected.

5.3 Fairness

An important factor in the evaluation of any traffic control mechanism is its fairness. If the mechanism fails to divide bandwidth equally between competing connections, then some connections may unfairly receive better service than others. We use the simple "parking lot" model depicted in Figure 9 to examine the fairness of the SAMM algorithm. Propagation delays on links $L_1, \ldots, L_4$ are 10 msec, representing distances of 2000 km, and each of these links is loaded with 6 Mbps of background traffic generated by four independent N-state MMPP processes. To adjust the burstiness of the background traffic, three values for the number of MMPP states are used: $N = 10$ (heavily bursty), $N = 50$ (moderately bursty) and $N = 2000$ (lightly bursty). Sample traces for each degree of burstiness are shown in Figure 10.

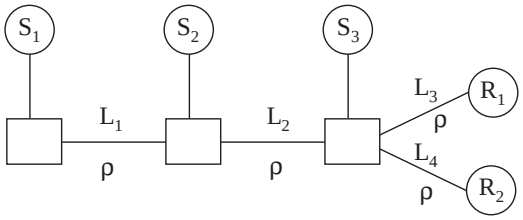
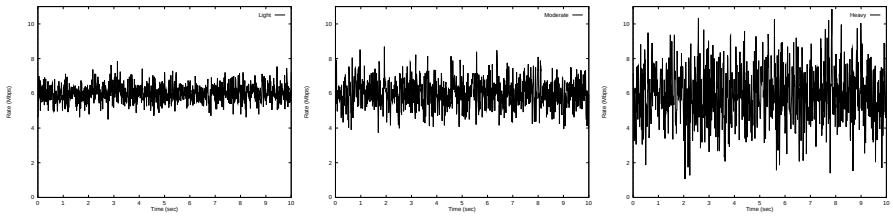


Figure 9: *Simulation model for evaluating fairness*



(a) Lightly bursty (b) Moderately bursty (c) Heavily bursty

Figure 10: *Sample traces of MMPP background traffic*

The allocation of bandwidth to competing video traffic streams is said to be optimal if it is *max-min fair*. A max-min fair allocation of bandwidth occurs when all active connections not bottlenecked at an upstream node are allocated an equal share of the available bandwidth at every downstream node [9]. In the model shown in Figure 9, a max-min fair allocation of bandwidth occurs if all three sources transmit at the same rate. To measure fairness, we calculate the standard deviation σ of the rates that each source transmits across the bottleneck links L_3 and L_4. An optimally fair allocation results in a standard deviation of zero.

Results for this set of simulations are shown in Table 2. There is a consistent degradation of fairness as the burstiness of the interfering traffic increases. This result was expected, since it is difficult for senders more distant from the shared bottleneck links (L_3 and L_4) to adapt their rates in response to rapid changes in the available bandwidth. Senders close to the shared bottleneck links unfairly grab a larger portion of the available bandwidth, especially when the background traffic is bursty. This kind of unfairness can be elimi-

Table 2: *Video transmission rates and fairness with FIFO queues and fair queueing*

	Lightly Bursty Background			
	Rate (Mbps)			Fairness
Scheduling	S_1	S_2	S_3	σ
FIFO	1.318	1.332	1.350	0.025
Fair queuing	1.331	1.331	1.331	0.000
	Moderately Bursty Background			
	Rate (Mbps)			Fairness
	S_1	S_2	S_3	σ
FIFO	1.312	1.316	1.349	0.035
Fair queuing	1.333	1.333	1.333	0.000
	Heavily Bursty Background			
	Rate (Mbps)			Fairness
	S_1	S_2	S_3	σ
FIFO	1.299	1.312	1.400	0.094
Fair queuing	1.333	1.333	1.333	0.000

nated by using fair queueing within each of the router output ports. If traffic flows from senders S_1, S_2 and S_3 are buffered in isolated queues and served on a round-robin basis, then their allocations of bottleneck link bandwidth become virtually identical as shown in the table.

6 CONCLUSION

We have introduced the class of algorithms known as source adaptive multilayered multicast (SAMM) algorithms and have studied their use for the multicast distribution of video. We have also proposed and investigated a simple end-to-end SAMM algorithm for possible use in the Internet. In SAMM algorithms, the source transmits several layers of video and adjusts their rates in response to congestion feedback from the receivers and/or the network.

We have also introduced a network architecture defining the source, receiver and network functions necessary to support SAMM algorithms. The architecture mandates that routers implement some form of priority packet discarding in order to support layered transmissions, as well as a class-based flow isolation mechanism at routers to prevent SAMM flows from negatively impacting the performance of other flows in the network. The architecture also

includes feedback mergers, which prevent feedback implosion by consolidating the contents of feedback packets returning to the source.

Simulation results indicate that the proposed SAMM algorithm is capable of producing better video quality and network utilization than algorithms which transmit video layers at fixed rates. Furthermore, the proposed end-to-end SAMM algorithm exhibits good performance in terms of goodput, video quality and scalability.

References

[1] Albuquerque C., Vickers B.J., and Suda T. (1998) Multicast Flow Control with Explicit Rate Feedback for Adaptive Real-Time Video Services. *Proc. of SPIE's 1998 Performance and Control of Network Systems II*.

[2] Albuquerque C., Vickers B.J., and Suda T. (1999) An End-toEnd Source-Adaptive Multi-Layered Multicast (SAMM) Algorithm. *Proc. of the Int'l. Workshop on Packet Video*.

[3] Albuquerque C., Vickers B.J., and Suda T. (2000) Credit-Based Source-Adaptive Multi-Layered Video Multicast. *Performance Evaluation Journal*, (40)1-3, pages 135-159.

[4] ATM Forum Technical Committee, Traffic Management Working Group. (1996) ATM Forum Traffic Management Specification Version 4.0.

[5] Amir E., McCanne S., and Zhang H. (1995) An Application-level Video Gateway. In *Proc. of ACM Multimedia*.

[6] Amir E., McCanne S., and Katz R. (1998) An Active Service Framework and Its Application to Real-time Multimedia Transcoding. In *Proc. of ACM SIGCOMM*, pages 178–189.

[7] Assunção P. and Ghanbari M. (1996) Multi-Casting of MPEG-2 Video with Multiple Bandwidth Constraints. In *Proc. of the 7th Int'l. Workshop on Packet Video*, pages 235–238.

[8] Ballardie T., Francis P., and Crowcroft J. (1993) Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing. *Proc. of ACM SIGCOMM*.

[9] Bartsekas D. and Gallagher R. (1987) *Data Networks, second edition*. Prentice Hall.

[10] Bolot J., Turletti T., and Wakeman I. (1994) Scalable Feedback Control for Multicast Video Distribution in the Internet. In *Proc. of ACM SIGCOMM*, pages 58–67.

[11] Cheung S., Ammar M., and Li X. (1996) On the Use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution. In *Proc. of IEEE Infocom*.

[12] Clark D. and Fang W. (1998) Explicit Allocation of Best Effort Packet Delivery Service. Technical report, MIT LCS.

[13] Deering S., Estrin D., Farinacci D., Jacobson V., Liu C., and Wei L. (1996) The PIM Architecture for Wide-Area Multicast Routing. *IEEE/ACM Transactions on Networking*, 4(2):153–162.

[14] Demers A., Keshav S., and Shenker S. (1989) Analysis and Simulation of a Fair Queueing Algorithm. *Proc. of ACM SIGCOMM*.

[15] Floyd S. and Jacobson V. (1993) Random Early Detections Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4).

[16] Floyd S. and Jacobson V. (1995) Link Sharing and Resource Management Models for Packet Networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386.

[17] Gilge M. and Gusella R. (1991) Motion Video Coding for Packet-Switching Networks – An Integrated Approach. *SPIE Conference on Visual Communications and Image Processing*.

[18] Kanakia H., Mishra P., and Reibman A. (1993) An Adaptive Congestion Control Scheme for Real-Time Packet Video Transport. *Proc. of ACM SIGCOMM*.

[19] Lakshman T.V., Mishra P.P., and Ramakrishnan K.K. (1997) Transporting Compressed Video over ATM Networks with Explicit Rate Feedback Control. In *Proc. of IEEE Infocom*.

[20] Li X., Paul S., and Ammar M. (1998) Layered Video Multicast with Retransmissions (LVMR): Evaluation of Hierarchical Rate Control. *Proc. of IEEE Infocom*.

[21] McCanne S., Jacobson V., and Vetterli M. (1996) Receiver-Driven Layered Multicast. In *Proc. of ACM SIGCOMM*, pages 117–130.

[22] Moy J. (1994) Multicast Extensions to OSPF. Request for Comments 1584, Internet Engineering Task Force.

[23] Omori Y., Suda T., Lin G., and Kosugi Y. (1994) Feedback-based Congestion Control for VBR Video in ATM Networks. In *Proc. of the 6th Int'l. Workshop on Packet Video*.

[24] Papadopoulos C., Parulkar G., and Varghese G. (1998) An Error Control Scheme for Large-Scale Multicast Applications. *Proc. of IEEE Infocom*, pages 1188–1196.

[25] Parekh A. and Gallager R. (1993) A Generalized Processor Sharing Approach to Flow Control – the Single Node Case. *IEEE/ACM Transactions on Networking*, pages 344–357.

[26] Paul S., Sabnani K., Lin J., and Bhattacharyya S. (1997) Reliable Multicast Transport Protocol (RMTP). *IEEE Journal on Selected Areas in Communications*.

[27] Sharon C., Devetsikiotis M., Lambadaris L., and Kaye A. (1995) Rate Control of VBR H.261 Video on Frame Relay Networks. In *Proc. of the International Conference on Communications (ICC)*, pages 1443–1447.

[28] Speakman T., Farinacci D., Lin S., and Tweedly S. (1998) PGM Reliable Multicast Specification. Internet draft (work in progress), Internet Engineering Task Force. ftp://ftp.ietf.org/internet-drafts/draft-speakman-pgm-spec-02.txt.

[29] Vicisano L. and Crowcroft J. (1998) TCP-like Congestion Control for Layered Multicast Data Transfer. *Proc. of IEEE Infocom*.

[30] Vickers B. J., Lee M. and Suda T. (1997) Feedback Control Mechanisms for Real-Time Multipoint Video Services. *IEEE Journal on Selected Areas in Communications*, 15(3).

[31] Vickers B.J., Albuquerque C., and Suda T. (1998) Adaptive Multicast of Multi-Layered Video: Rate-Based and Credit-Based Approaches. *Proc. of IEEE Infocom*.

[32] Waitzman D., Deering S., and Partridge C. (1988) Distance Vector Multicast Routing Protocol. Request for Comments 1075, Internet Engineering Task Force.