An End-to-End Source-Adaptive Multi-layered Multicast (SAMM) Algorithm*

Célio Albuquerque, Brett J. Vickers and Tatsuya Suda
{celio,bvickers,suda}@ics.uci.edu
Dept. of Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425

Technical Report 98-31

Abstract

Layered encoding is often recommended as a solution to the problem of varying bandwidth constraints in video multicast applications. However, multi-layered encoding of video is not sufficient to provide ideal video quality and bandwidth utilization, because network bandwidth constraints change over time. Adaptive techniques are required. In this paper we consider a rate-based Source-Adaptive Multi-layered Multicast (SAMM) algorithm that operates on an end-to-end basis. Video senders adapt the rates of their video layers in response to congestion feedback from receivers. To prevent feedback implosion, the use of computationally simple feedback merging servers is proposed. Network switches and routers are not required to implement complex flow or congestion control algorithms. The performance of the end-to-end SAMM algorithm is evaluated through simulations and compared with that of a non-source-adaptive mechanism.

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 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. Network bandwidth constraints change constantly 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 respond constantly 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.

*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 Ltd., Hitachi America, Standard Microsystem Corp, Tokyo Electric Power Company, Nippon Telegraph and Telephone Corporation (NTT), Nippon Steel Information and Communication Systems Inc. (ENICOM), Matsushita Electric Industrial Company, and Fundação CAPES/Brazil.

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 [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.

This paper focuses on an end-to-end SAMM algorithm that should be implementable in next generation internets. Prerequisites for its implementation include router-based priority drop preference 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 paper 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 introduced by this paper 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 can be sender-driven or receiver-driven. In a sender-driven algorithm, the source adapts its transmission rate in response to congestion feedback from the receiver or 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. Examples of sender-driven adaptive video transmission algorithms include SAMM [1, 2, 3], the probabilistic feedback algorithm of Bolot *et al.* [4], and several unicast adaptive algorithms [5, 6]. Examples of receiver-driven video transmission algorithms include Receiver-driven Layered Multicast (RLM) [7], Layered Video Multicast with Retransmission (LVMR) [8], and TCP-like congestion control for layered data [9]. The Destination Set Grouping (DSG) approach [10] shares features of the receiver-driven and sender-driven approaches.

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 fixed rates. Hence, if the path to one of the receivers has an amount of available bandwidth that does not exactly match one of the layer transmission rates, the network will be underutilized and the quality of that receiver's video will be suboptimal. On the other hand, sender-driven algorithms are able to adjust their layer transmission rates in response to network bandwidth availability and 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. Because sources using sender-driven algorithms receive a continuous stream of congestion feedback from the network, they can adapt to changing bandwidth constraints, either by adding a new video layer or 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 "join experiments" and branch pruning, both of which can be expensive in terms of throughput and loss. In most receiver-driven algorithms, receivers perform occasional join experiments, during which they request a new layer of data. If packets are lost during the join experiment, then the experiment is considered a failure. 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. In a network environment where bandwidth availability is fluctuating constantly and sometimes severely, the effects of join experiments and long leave latencies can result in periods of significant packet loss and,

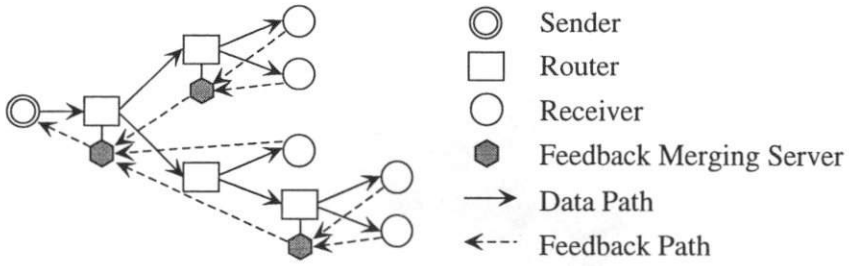


Figure 1: End-to-end SAMM architecture

for the case of video, poor video quality.

Receiver-driven algorithms have the advantage that they are naturally more friendly to competing network traffic than sender-driven algorithms are. Sender-driven algorithms typically send all video data on a single transport-layer connection and use priority indications to signal the drop precedence of each layer. This inevitably results in some low-priority traffic being needlessly sent down some branches of the multicast tree, only to be discarded further downstream. Competing traffic, such as adaptive TCP flows, may be unfairly dropped or delayed within the network if it shares the same FIFO queues as sender-driven video traffic. 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 [11] or weighted fair queueing [12] within the routers or switches. There is, however, a greater complexity involved in the implementation of class-based and fair queueing.

3 Architecture and Algorithm

In the proposed end-to-end SAMM algorithm, the sender adjusts its encoding parameters, including the number of video layers it generates and the encoding rate of each layer. These adjustments are performed in response to a continuous flow of congestion feedback from the receivers. Special feedback merging servers are distributed throughout the network and prevent feedback implosion by aggregating the feedback returning from receivers. See Figure 1 for an illustration of the architecture. The complexity of the end-to-end SAMM algorithm is concentrated at network terminal points, namely the sender, the receivers, and the feedback merging servers. Network routers and switches are not assumed to perform any significantly complex or novel functions.

As the sender generates video, it segments the video into multiple layers, each of which is assigned a unique priority. The layer with the highest priority, called the *base layer*, is used to generate packets containing the most important portions of the video stream, while additional layers, called *enhancement layers*, are encoded with progressively lower priorities and are used to generate packets that receivers can use to further refine the quality of the base layer stream. There are a number of ways to generate layered video data, but for the purposes of this paper we will assume that the source coarsely quantizes the video stream's frequency coefficients to produce the base layer and adds refinement data to produce enhancement layers.

Since the end-to-end SAMM algorithm relies on priority to differentiate layers, the underlying network architecture is required to give higher drop preference to low priority data. This is necessary to ensure that less important enhancement layer data is dropped in favor of more important enhancement and base layer data. Enhancement data is useless if the layer it is enhancing is missing. (According to the latest IETF draft specification, IPv6 will support 8 bits of priority [13], and efforts are underway to define standard semantics for these bits [14].)

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, a receiver can obtain a rough estimate of the bandwidth available on the path from the sender by monitoring how fast video packets arrive. In the end-to-end SAMM algorithm, each receiver monitors the number of video packets that arrive during a fixed interval called the *receiver monitoring interval* and calculates the *received video rate*. Whenever the receiver records a lost video packet during an interval, it assumes the available bandwidth is equal to the received video rate. When no losses are recorded, the receiver assumes the amount of available bandwidth is slightly higher than the received video rate – say by a factor of 10%. This allows the sender to increase its video transmission rate when new bandwidth becomes available within the network.

After receiving a given number of video packets, the receiver returns a feedback packet to the nearest upstream feedback merging server. This feedback packet contains a rate field, which is the receiver's estimate of the available bandwidth on the path from the sender.

Feedback merging servers are deployed to prevent feedback implosion, an undesirable situation in which a large number of receivers consumes significant return-path bandwidth by sending feedback to a single sender. Feedback merging servers ultimately form a virtual network overlaid on top of the underlying datagram network as shown in Figure 1. The servers merge information from arriving feedback packets and route the resulting feedback packets toward the next feedback merging server on the path to the sender. A feedback merging server may be a dedicated node inside the network, a router which has been enhanced to perform the merging function, or one of the participating receivers.

Feedback packets contain two vectors: a rate vector $\{r_i\}$, which lists the rates requested by receivers, and a counter vector $\{c_i\}$, which lists the number of receivers requesting each rate in $\{r_i\}$. When a receiver generates a feedback packet, it contains only entries for r_1 and c_1, with r_1 set to the measured available bandwidth and c_1 set to one. For every end-to-end connection, feedback merging servers store the most up-to-date feedback packets arriving on each downstream branch and merge them whenever one branch receives receives two feedback packets from the same connection. The resulting feedback packet is then returned towards the root of the video multicast tree.

During the vector merging operation, the feedback merging server collects the rate (r_i) and counter (c_i) entries from each incoming feedback packet and stores them in a local array, sorted by rate. Each rate entry corresponds to a video rate requested by one or more downstream receivers, while the counter values indicate how many downstream receivers have requested each rate. Ultimately, the rate values will be used by the sender 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_{max}). If the number of entries in the local rate array is less than or equal to L_{max}, then the merging is considered complete. However, if the number of entries exceeds L_{max}, 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 merging server 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 relatively useful estimate of video quality because it measures the total combined rate of uncorrupted video traffic arriving at an end system.

As feedback merging servers aggregate feedback packets, they attempt to determine the goodput that downstream receivers will receive. The combined goodput G is estimated from the values listed in a 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 each entry. To determine which entry to remove from the local rate array, the merging server 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. For a specific example of this algorithm in operation, see [2].

By the time a feedback packet arrives at the sender, it contains the number of video layers to encode and a list of cumulative rates at which to encode each layer.

An optional enhancement to this algorithm is to reserve from the network a fixed amount of bandwidth equal to the *minimum transmission rate* and encode the base layer at this rate. Enhancement layers can then be encoded whenever congestion feedback indicates that further bandwidth is available. Reserving bandwidth in this way guarantees that all receivers will receive at least the base layer without losses. It is envisioned that such a guarantee may be provided by a network resource reservation algorithm like RSVP [15] or by the emerging differentiated services architecture [14].

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 video can be any type of layered video encoder (e.g., embedded zerotree wavelet, MPEG-2, etc.), which accepts uncompressed video information one block at a time. We assume the encoder receives a list of target bit rates for each video layer and that the encoder attempts to produce a layered video packet stream matching the target bit rates as close as possible. However, since it is virtually impossible to produce video at rates that precisely match the target bit rates, the encoder also outputs 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.

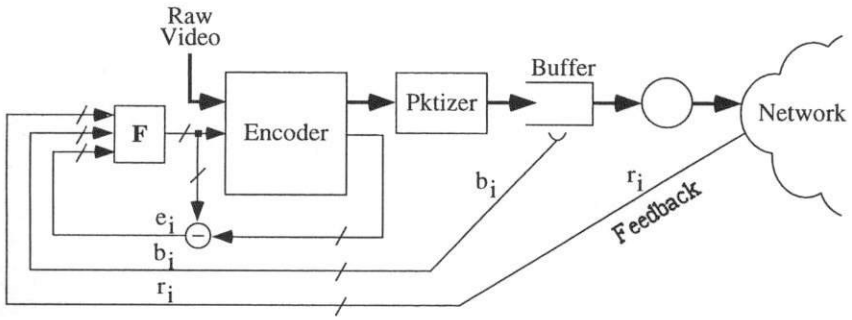


Figure 2: Video encoder and rate controller

The rate control function F determines the target bit rates for the encoder. 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 each 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 in the buffer. The constants α and β are weighting coefficients, T_d is the target buffer delay, and τ is the length of the video block interval. 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 FIFO buffer for transmission into the network. The packetizer interleaves each layer's packets according to its 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 delay, scalability with 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 SAMM algorithm.

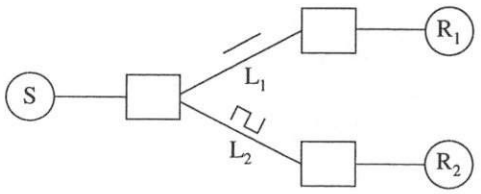


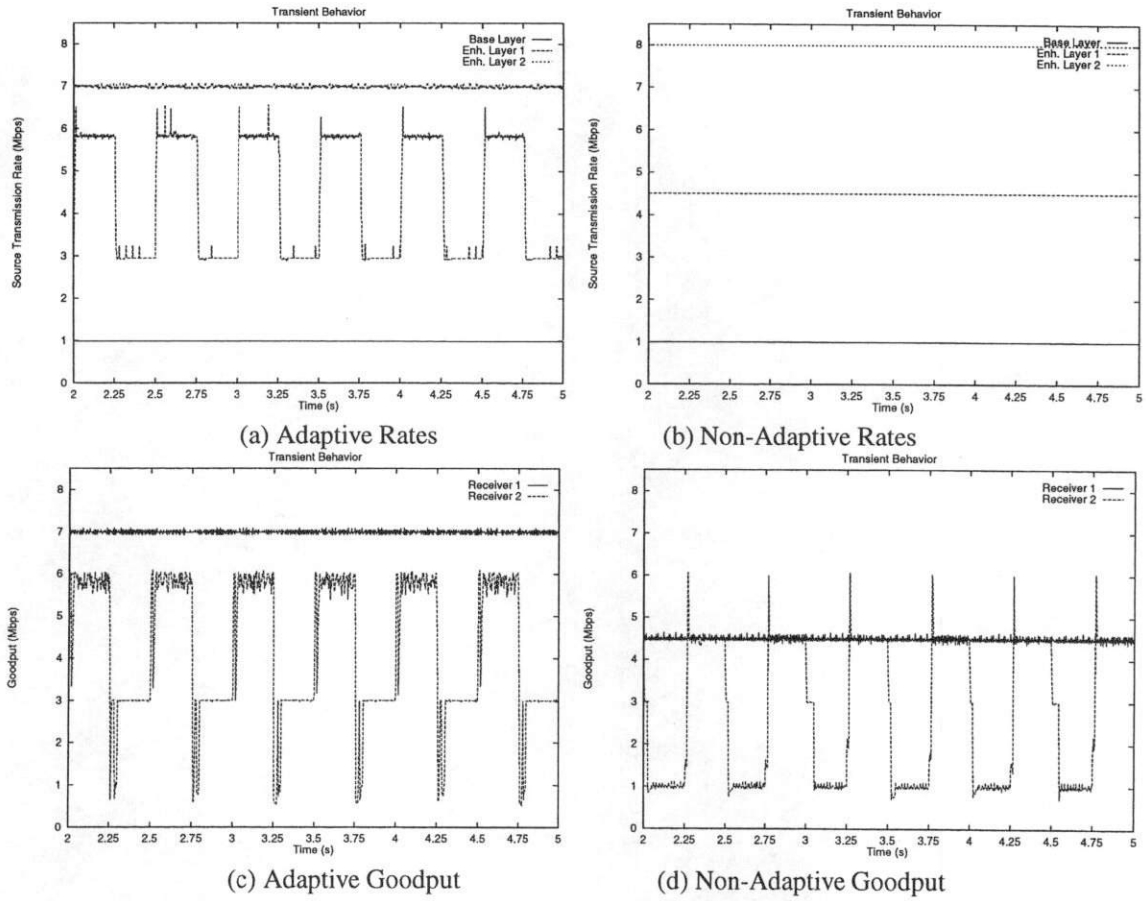
Figure 3: Simulation model for evaluating responsiveness

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 mechanism.

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 a sender which 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.

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 purposes of comparison, Figure 4(b) plots the cumulative transmission rates of each layer for the non-adaptive case.

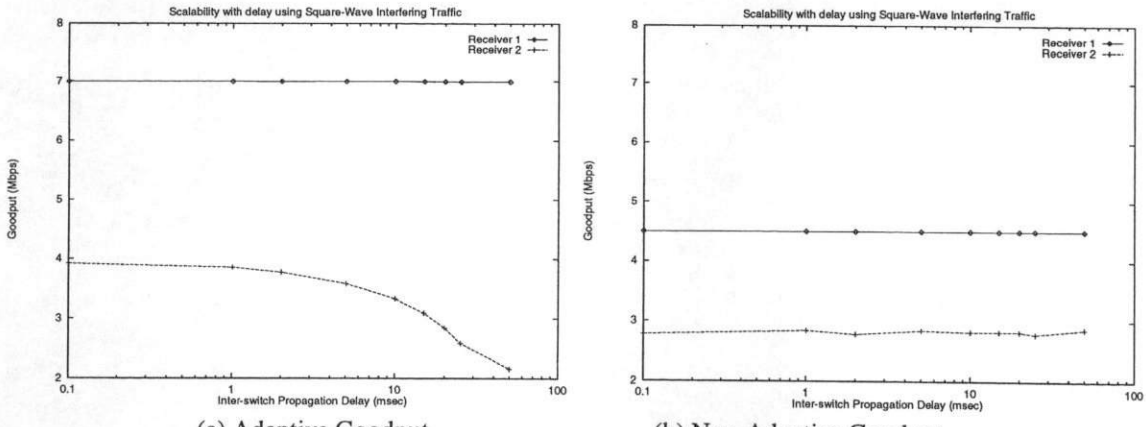
The receiver goodputs for the adaptive and non-adaptive mechanisms are shown in Figs. 4(c) and 4(d). Recall that video goodput is defined as the total throughput of all video layers received *without loss* during a block transmission interval.



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

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

Figure 4: Responsiveness temporal behavior



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

Figure 5: Responsiveness scalability with delay

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.

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 a multicast mechanism. Multicast connections 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.

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 ($\rho_1 = 2$ Mbps, $\rho_2 = 4$ Mbps, $\rho_3 = 6$ Mbps, $\rho_4 = 8$ Mbps). Background traffic is generated by a 10-state Markov-Modulated 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.

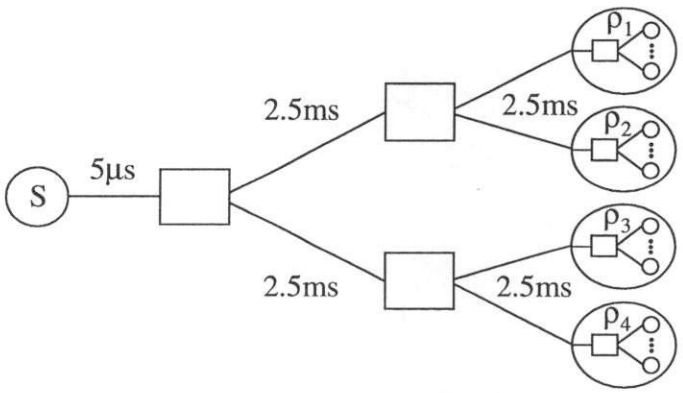


Figure 6: Simulation model for evaluating scalability

We first examine the performance of the SAMM algorithm as the number of receivers increases and the maximum number of video layers is varied from 2 to 8. Figure 7 plots the results. 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

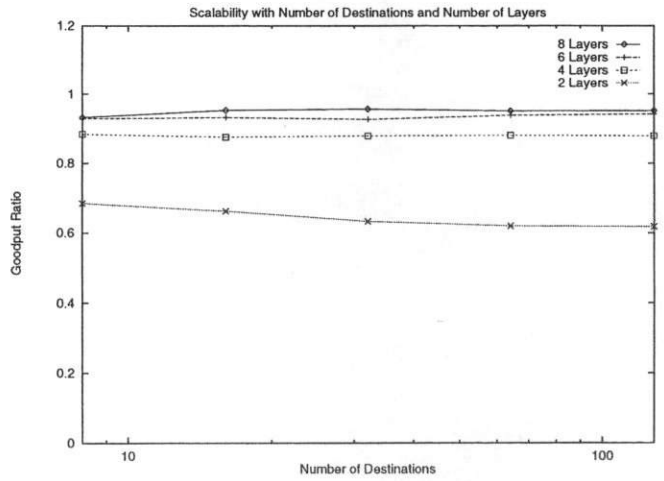


Figure 7: Average goodput ratio for all receivers vs. Number of receivers

thereby video quality) can be improved by increasing the maximum number of layers generated by the sender.

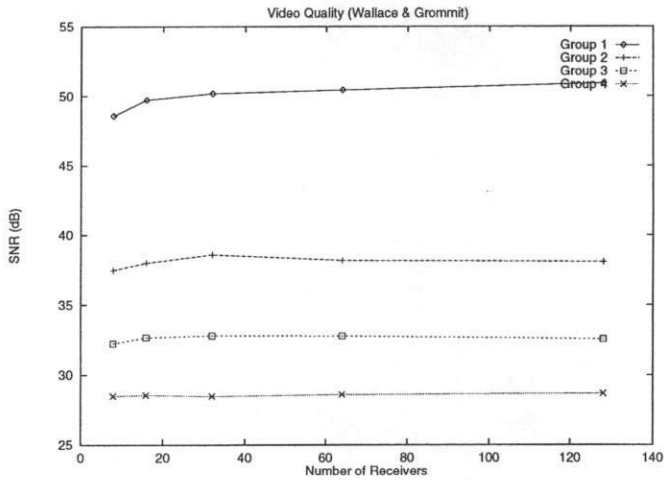


Figure 8: Average signal-to-noise ratio for all receivers vs. Number of receivers

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 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.

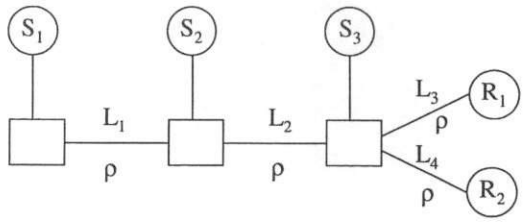
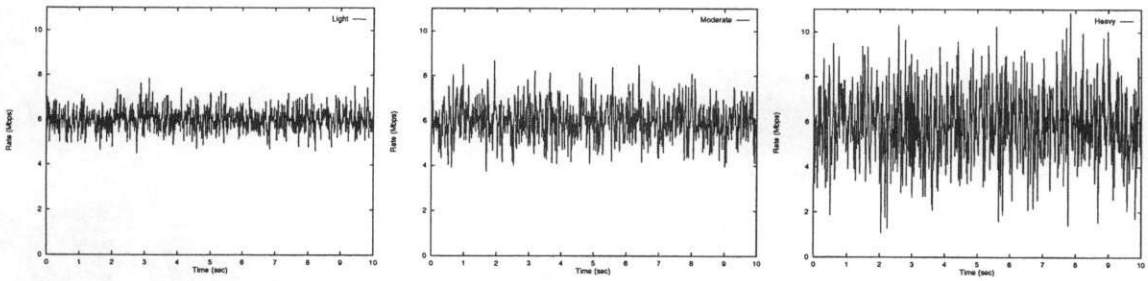


Figure 9: Simulation model for evaluating fairness



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

Figure 10: Sample traces of MMPP background traffic

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

Table 1: Video transmission rates and fairness with FIFO queues and fair queueing

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 [16]. 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 1. 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 eliminated 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

In this paper we have introduced the notion of a source adaptive multi-layered multicast (SAMM) algorithm and have proposed and investigated a simple end-to-end SAMM algorithm for possible use in next generation Internets. In the algorithm the sender transmits several layers of video and adjusts their rates in response to congestion feedback from the receivers. Feedback implosion is prevented by deploying feedback mergers throughout the network. The algorithm makes very few demands on intermediate network nodes, requiring only a priority drop preference mechanism and class-based flow isolation. Simulation results presented in this paper indicate good responsiveness and scalability performance of the end-to-end SAMM algorithm. Our present and future work includes exploration of the impact of the algorithm on an actual network, through implementation on a modified IP network testbed.

References

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

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

[3] C. Albuquerque, B.J. Vickers, and T. Suda. Multicast Flow Control with Explicit Rate Feedback for Adaptive Real-Time Video Services. *to appear at SPIE Performance and Control of Network Systems*, November 1998.

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

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

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

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

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

[9] L. Visciano and J. Crowcroft. TCP-like Congestion Control for Layered Multicast Data Transfer. *Proc. of IEEE Infocom*, April 1998.

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

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

[12] A. Demers, S. Keshav and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. *Proc. of ACM SIGCOMM*, pages 3–12, 1989.

[13] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. *IETF Internet Draft*, November 1997.

[14] K. Nichols and S. Blake. Definition of the Differentiated Services Field (DS Byte) in the IPv4 and IPv6 Headers. *IETF Internet Draft*, May 1998.

[15] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. *IETF Internet Draft*, June 1997. ftp://ds.internic.net/internet-drafts/draft-ietf-rsvp-spec-16.txt.

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