Source Adaptive Multi-Layered Multicast Algorithms for Real-Time Video Distribution *

Brett J. Vickers[†], Célio Albuquerque[‡], and Tatsuya Suda[‡]

† Dept. of Computer Science
Rutgers University
110 Frelinghuysen Rd.
Piscataway, NJ 08854-8019
732-445-1496 (phone)
732-445-0537 (facsimile)

‡ Dept. of Information
and Computer Science
University of California, Irvine
Irvine, CA 92697-3425
949-824-4105 (phone)
949-824-2886 (facsimile)

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. We contrast two specific SAMM algorithms: an end-to-end algorithm, in which only end systems monitor available bandwidth and report the amount of available bandwidth to the source, and a network-based algorithm, in which intermediate nodes also monitor and report available bandwidth. Using simulations which incorporate actual multi-layered video codecs, we demonstrate that SAMM algorithms exhibit better scalability and responsiveness to congestion than algorithms which are not source-adaptive. We also study the performance trade-offs between end-to-end and network-based SAMM algorithms.

1 Introduction

The use of layered encoding is frequently recommended as a solution to the problem of bandwidth heterogeneity in multicast video distribution. A multi-layered video encoder compresses a raw video sequence into

*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.

one or more streams, or layers, of differing priority. The layer with the highest priority, called the *base layer*, contains data representing the most important features of the video sequence, while additional layers, called *enhancement layers*, contain data that further refines the quality of the base layer stream. Prior to sending the video into the network, the source generates from each layer a packet stream, or flow, and assigns it a unique discarding priority. Packets from the base layer are assigned the highest priority, whereas packets from enhancement layers are assigned progressively lower priorities. When a network link experiences congestion, packets from the lowest priority enhancement layers are discarded, thereby preventing the loss of base layer and higher priority enhancement layer data. Given a network with support for priority-based packet discarding, the encoder may generate an enhancement layer of video for each unique multicast bandwidth constraint, thereby ensuring that all receivers obtain a quality of video commensurate with their available bandwidth.

However, multi-layered encoding alone is not sufficient to achieve high bandwidth utilization and high video quality, because network bandwidth constraints often change over time. To improve the bandwidth utilization of the network and optimize the quality of video obtained by each of the receivers, the source must dynamically adjust the number of video layers it generates as well as the rate at which each layer is transmitted. In order to do this, the source must obtain congestion feedback from the network.

We define a Source-Adaptive Multi-layered Multicast (SAMM) algorithm as any multicast traffic control algorithm that uses congestion feedback from the network to adapt the transmission rates of multiple layers of data. In this paper, we introduce two novel and promising SAMM algorithms: one in which congestion in the network is monitored at and indicated by the network's intermediate nodes, and another in which the responsibility for congestion control resides exclusively at the source and receivers. We refer to the former as a *network-based* SAMM algorithm and the latter as an *end-to-end* SAMM algorithm. Both SAMM algorithms are closed-loop; that is, video traffic flows from the source to the receivers, and a stream of congestion feedback flows from the receivers back to the source.

This paper also introduces a network architecture defining the source, receiver and network functions necessary to support SAMM algorithms. The architecture includes *feedback mergers*, which consolidate the contents of feedback packets as they return to the source. Feedback mergers are used to prevent feedback implosion, a scalability problem that occurs as an increasing number of receivers return congestion feedback to a single source. The architecture also requires that routers implement some form of priority packet discarding, to ensure that, in case of congestion, packets from less important video layers are discarded before packets from more important layers. In order to prevent SAMM video flows from negatively impacting the performance of other adaptive flows in the network, routers must also isolate flows based on service class. Examples of flow isolation mechanisms include class-based queueing [1] and weighted fair queueing [2, 3].

This paper is a significant extension of our previous work on SAMM algorithms [4, 5, 6, 7]. In this paper, we introduce the network architecture necessary to implement SAMM algorithms, describe in detail two new SAMM algorithms, and present the results of detailed simulations of these algorithms using actual video content. The paper is organized as follows. We review related work on the multicast of adaptively encoded video and discuss trade-offs between the existing approaches in section 2. In section 3, we present the SAMM architecture and describe the details of the network-based and end-to-end SAMM algorithms introduced in this paper. In section 4, we describe a video encoder rate control algorithm capable of generating multiple layers of video at dynamically varying rates. In section 5, the scalability, responsiveness, fairness, and end-to-end delay of both SAMM algorithms are evaluated and compared to that of a non-SAMM algorithm. Finally, we provide some concluding remarks 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 [8, 9, 10, 11, 12]. 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 [13]. 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 [14] 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 [15]. 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) [16], and TCP-like Congestion Control for Layered Data [17]) 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 [18, 19, 20]. 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 feedback 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 continuously 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 [1] or weighted fair queueing [2, 3] 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 Algorithms

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 two specific SAMM algorithms: one in which the primary congestion control functions reside within the network, and one 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.

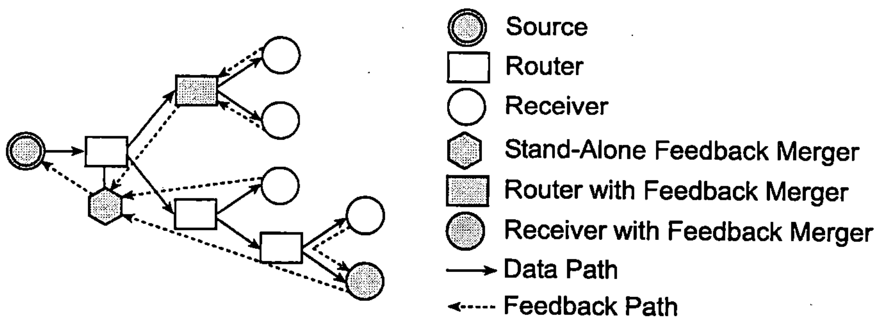

Source
Router
Receiver
Stand-Alone Feedback Merger
Router with Feedback Merger
Receiver with Feedback Merger
Data Path
Feedback Path

Figure 1: Network architecture for SAMM

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 [21], MOSPF [22], CBT [23], or PIM [24], 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 [1] and weighted fair queueing [2, 3], 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) [25], the Explicit Proportional Rate Control Algorithm (EPRCA) [26], and the network-based SAMM algorithm presented in this paper.

3.1.4 Feedback Mergers

Feedback mergers should be deployed to prevent feedback implosion, an undesirable situation in which a large number of receivers consumes 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 [27, 28, 29].

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 ones 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 Network-Based SAMM Algorithm

In this section we consider a network-based SAMM algorithm, in which routers participate in congestion control by monitoring their available bandwidth and marking passing feedback packets to indicate the explicit rates at which sources should transmit video. By exchanging congestion feedback with the network and the receivers, the video source learns the number of video layers to generate as well as the transmission rate for each layer. The algorithm is derived in large part from algorithms proposed by the ATM Forum for the rate control of Available Bit Rate (ABR) data sources [26].

In the network-based SAMM algorithm, the source periodically generates a control packet called a "forward feedback packet," which it multicasts to receivers. Upon receiving the forward feedback packet, a receiver copies the packet's contents into a "backward feedback packet" and returns it to the source, thereby closing the feedback loop. To maintain a steady flow of feedback between the source and the receiver, the source generates one new forward feedback packet for every N_f video packets sent, where N_f is a relatively small number such as 32.

As forward feedback packets travel from the source to the receivers, routers mark them in order to explicitly indicate the amount of bandwidth available in the network for the transmission of a SAMM video flow. The routers must therefore (1) monitor the amount of bandwidth available for video, (2) track the number of video multicast flows attempting to share the available bandwidth, and (3) calculate the fair share of the available bandwidth for each video multicast flow competing for the outgoing link. An existing algorithm, known as the Explicit Rate Indication for Congestion Avoidance (ERICA or ERICA+) algorithm [30], has been devised to support these functions for ABR data service, and we adopt a slightly modified version as part of the network-based SAMM algorithm. Most of ERICA's functions take place in the output ports of routers and switches, where the available bandwidth is monitored and feedback packets are marked. The functions the modified ERICA algorithm performs in each router are briefly summarized as follows:

1. Set the target utilization of the link bandwidth to some fraction of the total link capacity. A target utilization less than 100% helps the router prevent buffer overflows due to transient congestion effects. It also shortens queueing delays by keeping buffer occupancy low.
2. Monitor the number of active SAMM flows.
3. Monitor the amount of non-SAMM traffic arriving at the output port, and calculate the amount of bandwidth remaining for use by SAMM traffic. This amount is known as the "SAMM capacity."
4. Monitor the amount of SAMM traffic arriving at the port's output queue, and calculate the "overload." The overload is equal to the SAMM data arrival rate divided by the SAMM capacity. It measures the degree to which SAMM traffic is congesting (or underutilizing) the link.
5. Using the overload value, calculate the "flow share," which is equal to the flow's current transmission rate divided by the overload. The flow share represents an allocation of bandwidth that restores the link to the target utilization. It optimizes utilization of the link during periods of underload and prevents loss during periods of overload.

Field	Description	Used in forward feedback packets	Used in backward feedback packets
L	Maximum number of video layers allowed	●	●
R_C	Current combined rate of the source's video layers	●	
R_E	The maximum explicit rate allowed on the path	●	
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		●

Table 1: Contents of feedback packets used by the network-based algorithm.

6. Calculate the flow's "fair share" of the available bandwidth. The fair share is equal to the SAMM capacity divided by the number of active SAMM flows.

7. Set the explicit rate (R_E) value for the flow to the larger of the "flow share" and the "fair share." Place the explicit rate value into the forward feedback packet, but do not allow it to exceed the SAMM capacity of the link or the explicit rate value calculated by the previous hop.

All of this congestion control algorithm's monitoring operations take place over the duration of a short, fixed averaging interval. To prevent instability, new values for the overload, fair share, flow share and explicit rate are computed only once per averaging interval.

Table 1 lists the fields contained within each of the network-based algorithm's 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. The current combined rate (R_C) field contains the combined rate of all video layers currently being generated by the source. This field is used by the switches to calculate the flow share. The explicit rate field (R_E) is set by routers and indicates to each of the receivers how much bandwidth is available on the path from the source. At feedback packet generation time, the 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 process 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.

3.3 End-to-End SAMM Algorithm

The network-based algorithm, while accurately monitoring the available bandwidth and fairly dividing it among competing connections, has potential implementation difficulties. First, it requires that routers monitor their available bandwidth, perform congestion control, and mark feedback packets with explicit rate congestion information. Second, it employs forward feedback packets, which result in a reduction of the amount of bandwidth available for video data. To overcome some of these difficulties, we also consider and investigate 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 behavior of the end-to-end SAMM algorithm's source is the same as that of the network-based SAMM algorithm's source. 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. By contrast, the behavior of the end-to-end SAMM algorithm's receiver is significantly 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 how fast 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 [31].

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, 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.

After receiving a number of video packets, the receiver returns a feedback packet toward the source. The feedback packets of the end-to-end SAMM algorithm contain the same fields listed on Table 1 for the network-based SAMM algorithm *backward* feedback packets, including the rate vector (r_i) and the counter vector (c_i). The receiver reports its estimated available bandwidth in the first slot of the rate vector and sets the first slot of the counter vector to one. Feedback packets are collected and merged by feedback mergers or by the source in the exact same way as described for the network-based SAMM algorithm.

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 alternative to the more complex network-based SAMM algorithm.

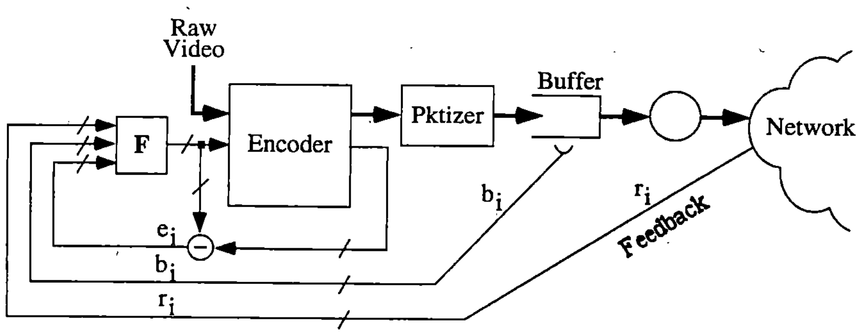


Figure 2: Video encoder and rate controller

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 network-based and end-to-end SAMM algorithms under various network configurations and conditions. These scenarios are designed to test the algorithms' responsiveness, scalability with respect to propagation delay, scalability with respect to the number of receivers, sensitivity to network buffer dimensioning, and fairness.

Unless otherwise specified, all simulations assume link capacities of 100 Mbps, propagation delays between end systems and routers of 5 μs, and propagation delays between routers of 5 ms. Without loss of generality, all packets are assumed to be the size of ATM cells (53 bytes), and two class-based queues are used at each router hop to isolate background traffic from SAMM video traffic. One queue is shared by competing background traffic while the other is shared by competing SAMM video traffic. To keep queueing delays minimal, only the number of packet buffers necessary to tolerate 1 ms of queueing delay are used. For most simulation models, this works out to approximately 200 packets per router hop for each video flow. For the network-based SAMM algorithm, the source generates feedback packets once for every 32 video packets, and for the end-to-end SAMM algorithm, the receiver generates and returns a feedback packet after receiving 32 video packets. We assume a feedback merger has been integrated into every router.

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. In this set of experiments, we use the end-to-end and network-based

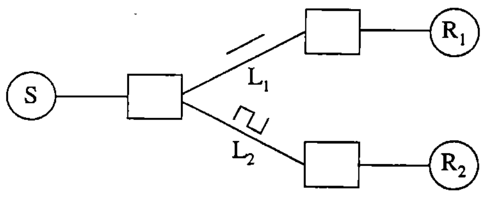


Figure 3: Simulation model for evaluating responsiveness

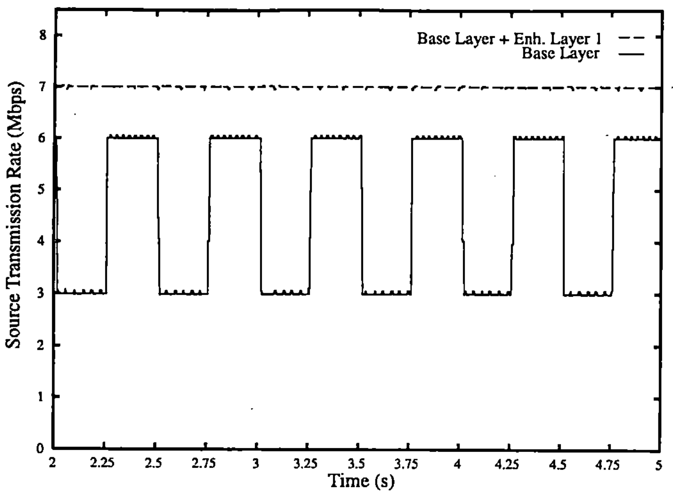
SAMM algorithms to illustrate the performance trade-offs between source-adaptive and non-source-adaptive algorithms. We also show the impact of network propagation delay on the responsiveness of the two SAMM algorithms.

We use the model in Figure 3 to evaluate the responsiveness of the SAMM algorithms. It consists of one video source 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 temporal response of the source to changes in available bandwidth on one of the links. The second experiment explores the impact of the network propagation delay on the performance of the SAMM algorithms.

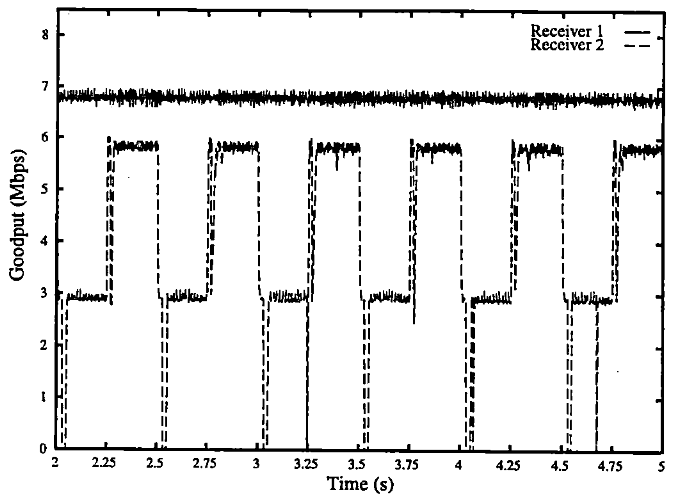
In the first experiment, all link capacities are 10 Mbps. We apply CBR background traffic at a rate of 3 Mbps to link L_1 and sharply oscillating square-wave background traffic to link L_2 in Figure 3. The square-wave traffic oscillates between constant rates of 4 and 7 Mbps over a period of 500 ms and is used to test the responsiveness of the source to sudden and substantial changes in available bandwidth. As a basis for comparison, we also examine the performance of a source 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 layered transmission mechanism that is not source-adaptive.

Figure 4 shows the results of the simulation. As shown in Figures 4(a) and (b), the adaptive sources alter the rate of one of their layers in response to the oscillating available bandwidth on link L_2. The remaining layer is transmitted at a cumulative rate of 7 Mbps, which corresponds to the available bandwidth on link L_1. Note that the source quickly responds to the square-wave traffic oscillations, usually within 10 milliseconds, which is equal to the round trip time. For the purposes of comparison, Figure 4(c) plots the cumulative transmission rates of each layer for the non-adaptive case.

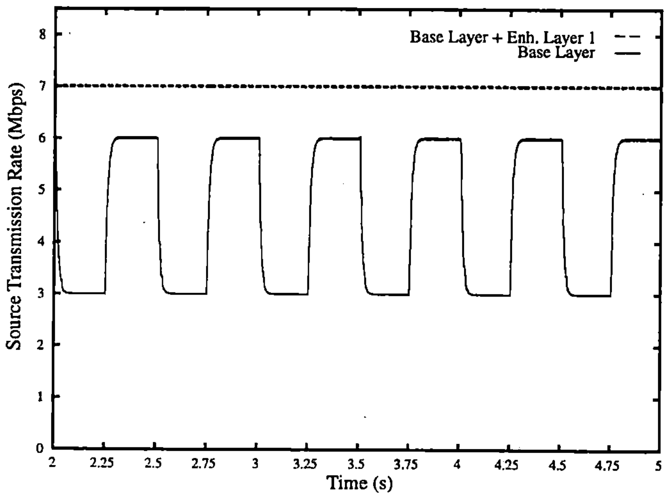
The receiver goodput values for the source adaptive and non-adaptive algorithms are shown in Figs. 4(d) through (f). Recall that video goodput is defined as the total throughput of all layers received without loss. At each receiver, a new value of the goodput is determined for each arriving video block. Clearly the SAMM



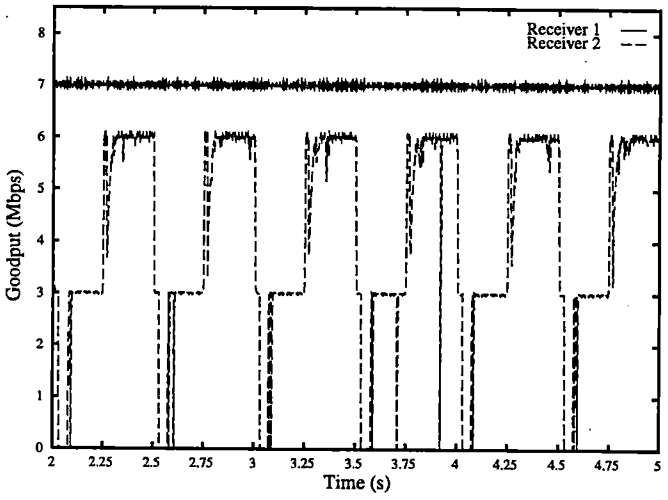
(a) Source Rates: Network-based SAMM



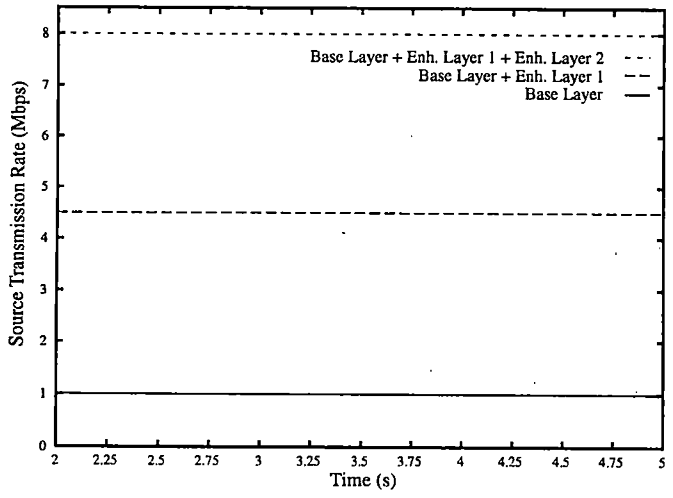
(d) Video Goodput: Network-based SAMM



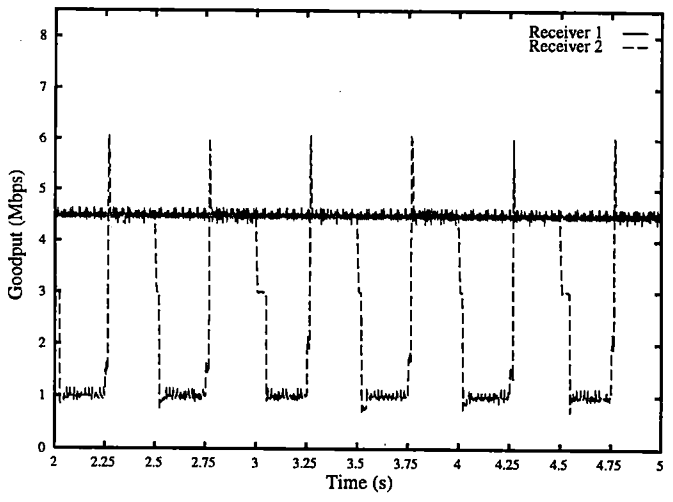
(b) Source Rates: End-to-end SAMM



(e) Video Goodput: End-to-end SAMM



(c) Source Rates: Non-Adaptive



(f) Video Goodput: Non-Adaptive

Figure 4: Responsiveness: Source transmission rates and goodput

algorithms produce better video goodput than the non-adaptive mechanism due to their ability to adjust each layer's transmission rate based on network congestion feedback. In both SAMM algorithms, there are brief periods of time when the video goodput falls to zero. This occurs whenever base layer packets are dropped by network routers due to buffer overflow, which occurs whenever the available bandwidth on link L_2 decreases suddenly to 3 Mbps.[1]

Figure 4 reveals an important trade-off between the two SAMM algorithms. The receiver impacted by oscillating background traffic on link L_2 suffers fewer losses to the base layer when using the network-based algorithm, whereas both receivers observe slightly higher peak goodput values when they use the end-to-end algorithm. There are two reasons for this trade-off. On the one hand, the network-based algorithm is able to respond more rapidly to changes in available bandwidth than the end-to-end algorithm, because congestion control decisions are made within routers, which have more complete and immediate knowledge of available bandwidth than do receivers. On the other hand, the end-to-end algorithm does not include the overhead of generating and transmitting forward feedback packets to gauge the amount of available bandwidth in the network, and so it allows more of the link capacity to be used for the transmission of video. Thus, while the network-based algorithm is more responsive to changes in available bandwidth, its greater responsiveness comes at the expense of lower peak goodput.

In the second experiment, we explore the impact of propagation delay on video goodput. We again apply CBR background traffic on link L_1 and square-wave background traffic with a period of 500 ms on link L_2. The background traffic transmission rates are the same as those for the first experiment, and propagation delays between routers are varied from 0.1 to 50 msec.

The average video goodput delivered to each receiver is plotted in Figure 5. As propagation delay increases, the average goodput delivered to receiver R_2 by the SAMM algorithms drops almost linearly. This is due to the fact that the source uses increasingly stale congestion feedback to adjust its layer transmission rates as the propagation delay increases. Despite this drawback, the SAMM algorithms produce better goodput than the non-adaptive mechanism for both receivers and all observed propagation delays in this scenario.

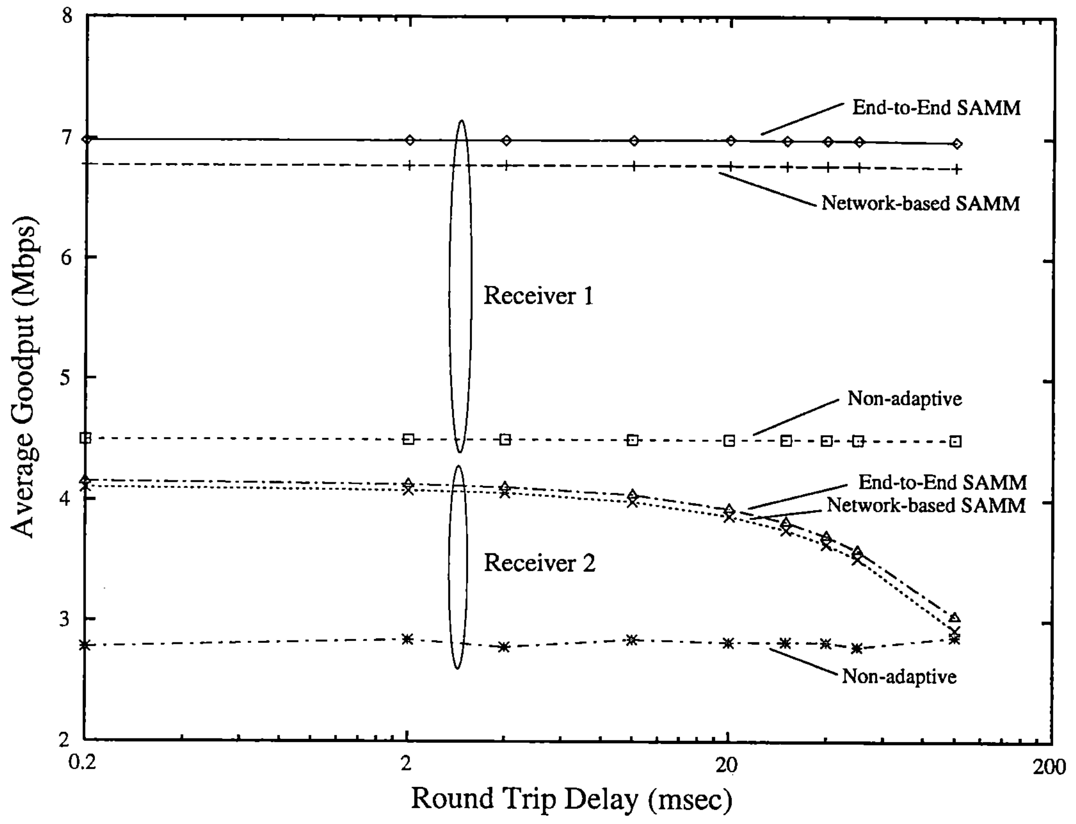


Figure 5: Responsiveness: Scalability with delay

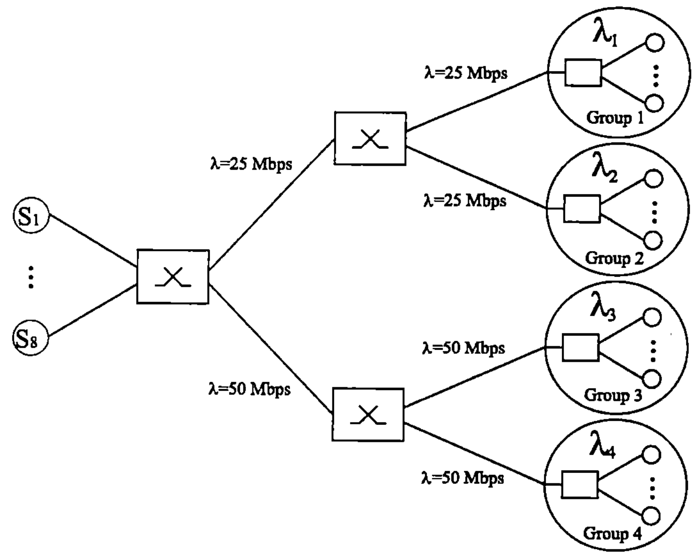


Figure 6: Simulation model for evaluating scalability

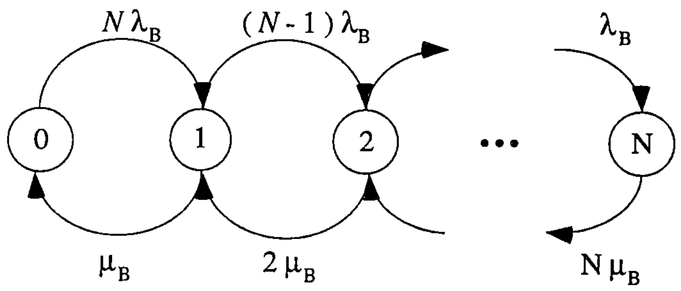


Figure 7: Background traffic model: MMPP representing N background sources

5.2 Scalability

Scalability is perhaps the most important performance measure of a multicast traffic control algorithm. Multicast connections can reach hundreds of receivers, each with varying bandwidth constraints. It is therefore important to understand how a multicast traffic control algorithm performs as the number of receivers grows.

The network model shown in Figure 6 and used to evaluate the scalability of the SAMM algorithms consists of eight video sources, four equally sized groups of receivers, and seven routers. Within each receiver group, we vary the number of receivers between 2 and 32. We generate background traffic using a superposition of 2000 on-off, interrupted Poisson processes, each with state-to-state transition rates of 100

[1]To prevent the goodput from falling to zero during these brief transitional periods, a SAMM source may opt to transmit an additional layer at a low, constant bit rate.

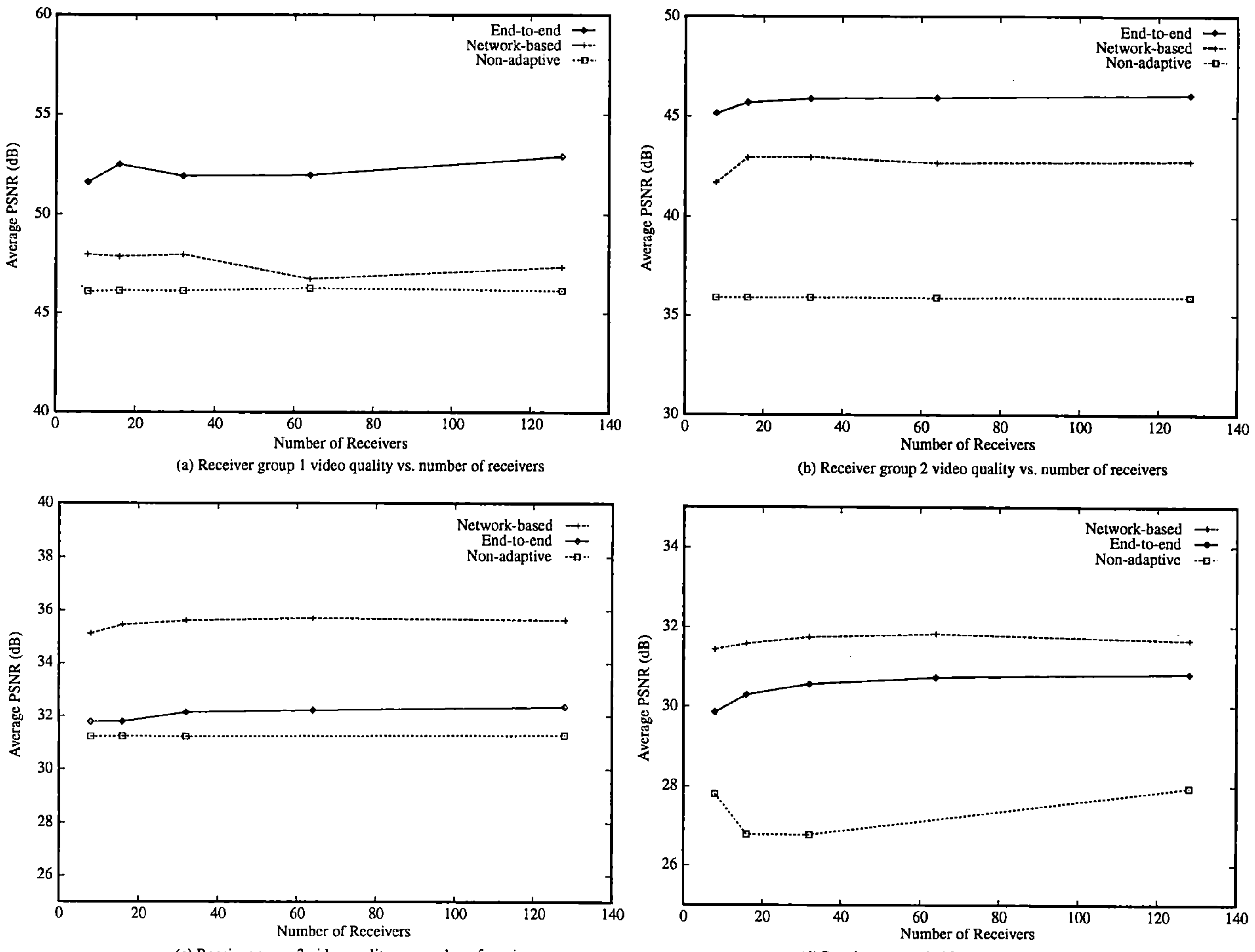
per second. This superposition is equivalent to the Markov Modulated Poisson Process (MMPP) shown in Figure 7, where $N = 2000$ and $\lambda_B = \mu_B = 100$ sec^{-1}. We choose this background traffic model because it generally produces burstier traffic than a simple Poisson process. Independent MMPP-generated background traffic streams are applied to all intermediate links at average rates (λ) of either 25 or 50 Mbps and to each leaf link as shown in Figure 6. The average traffic loads on leaf links are divided into four heterogeneous groups ($\lambda_1 = 30$ Mbps, $\lambda_2 = 50$ Mbps, $\lambda_3 = 65$ Mbps, $\lambda_4 = 80$ Mbps).

We first examine the performance of the SAMM algorithms as the number of receivers increases. For the purposes of comparison, we also consider an algorithm which is not source-adaptive and transmits five layers of video at fixed, evenly distributed cumulative rates of 1, 3.25, 5.5, 7.75 and 10 Mbps. Using an embedded zero-tree wavelet codec with a rate controller like the one described in section 4, each video source encodes actual multi-layered video sequences (taken from the Academy Award winning short animation *Wallace and Grommit*) and transmits them through the simulated network shown in Figure 6. We assume the encoder can generate up to five video layers. When video packets arrive at receivers, they are decoded and the resulting video quality is examined by calculating its average peak signal-to-noise ratio.[2] We vary the total number of multicast receivers between 8 and 128 in order to examine scalability.

Figure 8 plots the average peak signal-to-noise ratio of the decoded video sequence for a selected receiver from each receiver group as a function of the total number of receivers. Only the video generated by one of the eight video sources is considered when computing the average peak signal-to-noise ratio, since each source behaves in a statistically equivalent manner. As the figure shows, the quality of video obtained by a receiver is determined by the average amount of bandwidth available to it, just as expected. More important from the perspective of scalability is the fact that the signal-to-noise ratio and hence the video quality at each receiver remains relatively constant as the number of receivers increases. Furthermore, the video quality delivered to members of each group of receivers is consistently higher for the SAMM algorithms than it is for the algorithm which is not source adaptive.

Figure 8 reconfirms the performance trade-off between the end-to-end and network-based SAMM algorithms that we observed in section 5.1. Recall that the network-based algorithm achieves better responsiveness by using some of the available bandwidth to transmit forward feedback packets, whereas the end-to-end

[2]The average peak signal-to-noise ratio is a logarithmic measure of the difference between the original and received video sequences. The larger the value, the lesser the distortion. It is calculated by observing the peak signal-to-noise ratio for each frame and averaging across all frames in the video sequence.



(a) Receiver group 1 video quality vs. number of receivers

(b) Receiver group 2 video quality vs. number of receivers

(c) Receiver group 3 video quality vs. number of receivers

(d) Receiver group 4 video quality vs. number of receivers

Figure 8: Scalability: Average peak signal-to-noise ratio for all receiver groups vs. total number of receivers

algorithm uses more available bandwidth to transmit video but responds less quickly to congestion. This trade-off is again observed in Figure 8. If we compare the video quality obtained by the end-to-end and network-based SAMM algorithms for members of the least bandwidth-constrained receiver groups (groups 1 and 2), we see that the feedback overhead of the network-based SAMM algorithm results in lower video quality for members of these groups. However, when we examine the video quality obtained by receivers with more highly constrained bandwidth (groups 3 and 4), we observe that responsiveness to congestion plays a greater role in determining the video quality. This is due to the fact that most bandwidth-constrained receivers receive a smaller number of video layers, and packet losses in these layers are mitigated by fewer underlying video layers. Since the network-based algorithm is more responsive than the end-to-end algorithm, it results in fewer packet losses for bandwidth-constrained receivers and therefore achieves higher video quality for bandwidth-constrained users, even despite its greater feedback overhead.

5.3 Buffer Dimensioning

SAMM algorithms require routers to implement priority-based packet discarding so that low priority packets are preferentially dropped in case of congestion. That is, when a router's output queue overflows, the lowest priority packet in the queue is discarded. Unfortunately, using priority discarding mechanisms to support layered video causes many network queues to remain full, or nearly full, for the entire duration of the video session. It is therefore important to dimension these buffers conservatively to prevent excessively long queueing delays. At the same time, the buffers must be large enough to minimize the discarding of important video packets.

Using the network model shown in Figure 6, we study the impact of router queue sizes on the video quality and end-to-end delay of the SAMM algorithms. Video goodput is used as a metric for video quality, and the size of each router's output queues is varied between 20 and 2000 cell-sized packets.

Figure 9 plots the average goodput and end-to-end delay observed by a receiver sampled from each receiver group. Although all eight SAMM sources are transmitting video, only the average video goodput from one of the sources is shown in this figure. Clearly, the average video goodput at all receivers increases as the size of output queues in the network increases. This result is expected, because fewer low priority packet losses occur when network buffers are large. More interesting, however, is the difference in the shapes of the curves for the network-based and end-to-end SAMM algorithms. The goodput curves for the network-based algorithm begin to flatten out earlier than the end-to-end curves do, suggesting that the network-based

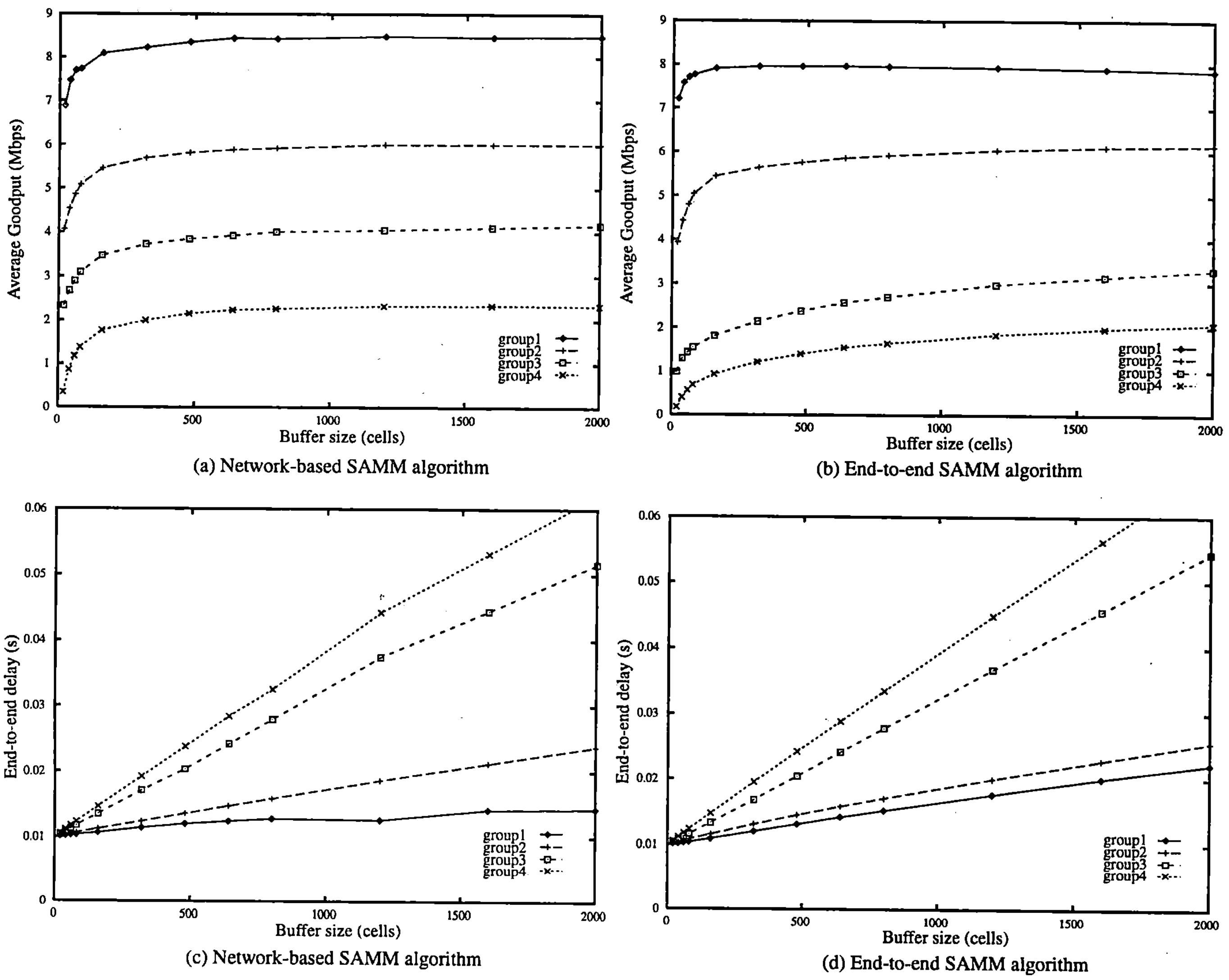


Figure 9: Average goodput and end-to-end delay for all groups vs. Buffer size

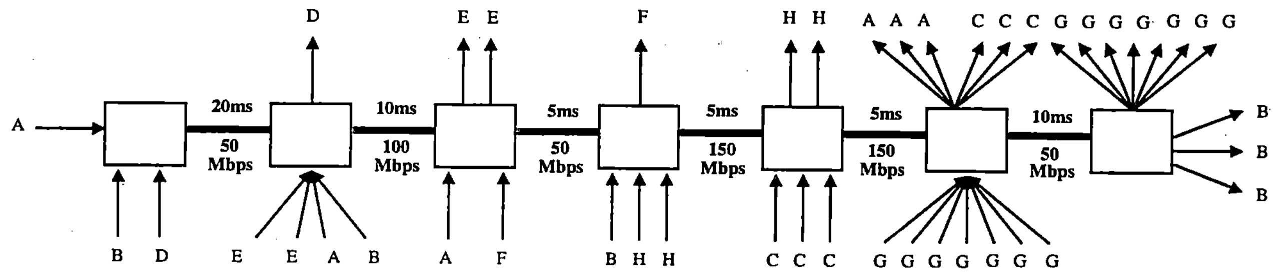


Figure 10: GFC-2: Simulation model for evaluating fairness

algorithm can achieve its best goodput with smaller router buffers.

In both SAMM algorithms, the end-to-end delays increase linearly with network buffer size. Moreover, receivers with more severe bandwidth constraints (groups 3 and 4) experience larger end-to-end delays, because queues in routers along the paths to these receivers fill with more low priority traffic that is ultimately discarded. The network-based algorithm exhibits smaller end-to-end delays than the end-to-end algorithm due to the ability of the network-based algorithm to regulate source transmission rates according to router queue occupancies.

5.4 Fairness

Another important factor in the evaluation of traffic control algorithms is their fairness. If an algorithm fails to divide bandwidth equally between competing flows, then some flows may unfairly receive better service than others. To evaluate the fairness of the SAMM algorithms, we use a standard network model known as the second General Fairness Configuration (GFC-2) [32, 33]. In this model, there are 22 competing sources, 22 receivers and 7 routers, and all links serve as bottlenecks for at least one of the 22 flows. This configuration is illustrated in Figure 10. All flows with the same label are bottlenecked at the same link. All entry and exit links have propagation delays of 5 ms and link capacities of 150 Mbps.

The allocation of bandwidth to competing video traffic streams is said to be optimal if it is *max-min fair.* At the output port of a given router, an allocation of bandwidth is said to be max-min fair if all active flows not bottlenecked at another router are allocated an equal share of the available bandwidth [34].

Table 2 contains the results obtained by simulating the GFC-2 model with both SAMM algorithms. In the first column, the ideal max-min fair allocations of bandwidth are shown for each type of flow in the GFC-2 configuration. The next two columns list simulation results showing the rates to which a selected

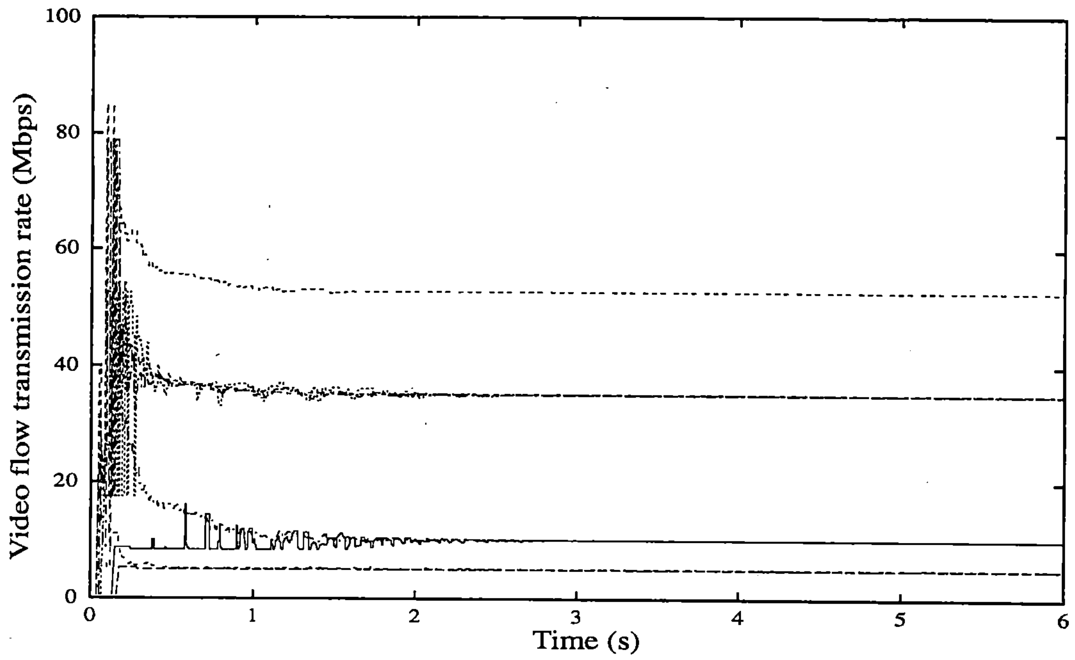
Flow Type	Max-min fair share	Network-based SAMM	End-to-end (E-to-E) SAMM	E-to-E SAMM w/round robin
A	10	10.12	11.99	10.04
B	5	5.00	3.26	5.01
C	35	35.18	37.66	35.21
D	35	35.21	39.97	34.79
E	35	35.21	38.98	34.93
F	10	10.01	16.18	10.18
G	5	5.08	6.80	5.26
H	52.5	52.72	56.50	52.81

Table 2: Max-min fair allocation and average video transmission rates (in Mbps)

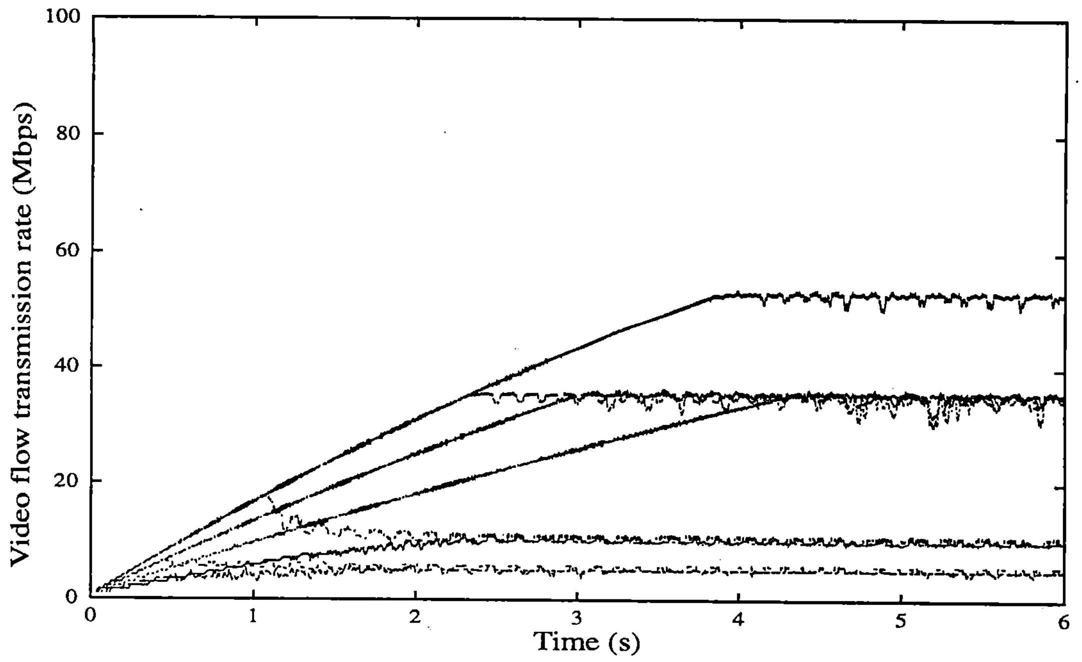
flow from each flow type converges at equilibrium for the network-based and end-to-end SAMM algorithms. The network-based algorithm is clearly able to achieve bandwidth allocations that are close to max-min fair, since explicit computations of the fair share are performed in every router. On the other hand, the end-to-end algorithm cannot enforce fairness, and hence flows with shorter round-trip times tend to consume a larger amount of bandwidth. This results in an allocation of bandwidth that is unfair and strongly dependent on the distance between source and receiver.

One possible way to improve the fairness of the end-to-end SAMM algorithm is to impose some form of per-flow scheduling at the routers. The simplest form of per-flow scheduling is a round robin algorithm, which cycles packet services between flows. This solution has greater implementation complexity than the simple end-to-end SAMM algorithm, which uses only FIFO queues at routers, but it is still less complex than the network-based algorithm, which requires rate monitoring and explicit rate calculation. The right-most column of Table 2 lists simulation results showing the observed bandwidth allocations for the end-to-end SAMM algorithm enhanced with round robin scheduling. It shows that a simple round robin scheduler at each router can substantially improve the fairness of the end-to-end SAMM algorithm.

It is also important to examine how rapidly the two SAMM algorithms converge to their equilibrium allocations. Figure 11 shows how the transmission rate of a selected source in each flow type converges over time when the network-based and enhanced end-to-end SAMM algorithms are used. The network-based algorithm clearly converges much more rapidly to a max-min fair allocation than does the enhanced end-to-end algorithm. Again, this is due to the explicit rate mechanisms built into the network-based algorithm.



(a) Network-based SAMM algorithm



(b) End-to-end SAMM algorithm with Round Robin Scheduling

Figure 11: Video Source Rates for GFC-2 configuration

6 Conclusion

In this paper we have introduced the class of algorithms known as source adaptive multi-layered multicast (SAMM) algorithms, and we have studied their use for the multicast distribution of video. We have proposed and investigated a simple end-to-end SAMM algorithm for possible use in the Internet and a more sophisticated network-based SAMM algorithm which requires monitoring and reporting of available bandwidth by intermediate network nodes. In both 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 class-based flow isolation mechanisms 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. Note that we have not addressed the placement or dynamic instantiation of feedback mergers in the network, opting instead to leave this issue for future research.

Simulation results presented in this paper indicate that SAMM algorithms are capable of producing better video quality and network utilization than algorithms which transmit video layers at fixed rates. Furthermore, both SAMM algorithms exhibit good performance in terms of goodput, video quality and scalability while requiring only a minor amount of buffer allocation at routers in the network. The principle difference between the two algorithms resides in their ability to respond to changes in available bandwidth and the speed with which they converge to fair allocations of bandwidth. By introducing additional complexity into the router nodes, the network-based algorithm is able to respond more rapidly and fairly to congestion within the network.

References

[1] 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.

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

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

[4] 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.

[5] 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.

[6] C. Albuquerque, B.J. Vickers, and T. Suda. 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*, November 1998.

[7] C. Albuquerque, B.J. Vickers, and T. Suda. An End-toEnd Source-Adaptive Multi-Layered Multicast (SAMM) Algorithm. *Technical Report ICS-TR 98-31, University of California, Irvine*, November 1998.

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

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

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

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

[12] 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.

[13] 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.

[14] 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.

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

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

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

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

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

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

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

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

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

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

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

[26] ATM Forum Technical Committee, Traffic Management Working Group. ATM Forum Traffic Management Specification Version 4.0, April 1996.

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

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

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

[30] R. Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, and R. Viswanathan. ERICA Switch Algorithm: A Complete Description. *ATM Forum/96-1172*, August 1996.

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

[32] R. Simcoe. Test Configurations for Fairness and Other Tests. Technical Report 94-0557, ATM Forum, July 1994.

[33] B. Vandalore, S. Fahmy, R. Jain, R. Goyal, and M. Goyal. A Definition of Generalized Fairness and its Support in Switch Algorithms. Technical Report 98-0151, ATM Forum, February 1998.

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