Leveraging Single Rate Schemes in Multiple Rate Multicast Congestion Control Design

Gu-In Kwon guin@cs.bu.edu
John W. Byers byers@cs.bu.edu
Computer Science Department
Boston University
Boston, MA 02215

***Abstract*— A significant impediment to deployment of multicast services is the daunting technical complexity of developing, testing and validating congestion control protocols fit for wide-area deployment. Protocols such as pgmcc and TFMCC have recently made considerable progress on the single rate case, i.e. where one dynamic reception rate is maintained for all receivers in the session. However, these protocols have limited applicability, since scaling to session sizes beyond tens of participants necessitates the use of multiple rate protocols. Unfortunately, while existing multiple rate protocols exhibit better scalability, they are both less mature than single rate protocols and suffer from high complexity.**

We propose a new approach to multiple rate congestion control that leverages proven single rate congestion control methods by orchestrating an ensemble of independently controlled single rate sessions. We describe a new multiple rate congestion control algorithm for layered multicast sessions that employs a single rate multicast congestion control as the primary underlying control mechanism for each layer. Our new scheme combines the benefits of single rate congestion control with the scalability and flexibility of multiple rates to provide a sound multiple rate multicast congestion control policy.

I. Introduction

Despite considerable effort and numerous technical advances, a suitable multiple rate multicast congestion control mechanism fit for wide area deployment is still yet to emerge. The primary reason appears to be the daunting complexity associated with delivering different TCP-friendly rates to different participants within the session. In all existing schemes for multiple rate congestion control, versions of layered multicast (originally proposed in [2]) are employed, whereby different multicast groups within the multicast session transmit at different rates, and participants use IGMP messages to join and leave groups to adjust their rate. But the significant challenges associated with this method are that the actions of one receiver can adversely impact other receivers; moreover, joins can place sudden load on the network, leading to unfriendliness to protocols such as TCP. Existing methods to mitigate these problems ultimately lead to very complex multiple rate congestion control designs that are difficult to evaluate.

Further evidence of the technical hurdles associated with multiple rate schemes is given by promising recent advances in *single rate* multicast congestion control, notably pgmcc [3] and TFMCC [4]. With single rate congestion control schemes, the sender transmits at a rate requested by the slowest receiver in the group. While these protocols are not designed to scale to large or heterogeneous audiences, there is building consensus that these protocols are sufficiently mature and well-tested for Internet deployment. In this paper, we seek to leverage these advances. In particular, we explore a new direction in multiple rate multicast congestion control, namely building a multiple rate scheme from an ensemble of single rate sessions, *each of which has their own independent control*. The major advantage of this method is that it leverages proven single rate congestion control mechanisms to provide an effective multiple rate scheme with relatively little additional complexity. This is in contrast to all existing multiple rate congestion control schemes, which provide only an integrated control mechanism *across* layers, and do not attempt to take advantage of control mechanisms *within* layers. As a result, these integrated controls are often extremely complex, and are difficult to test and validate.

This work appeared in preliminary form in the proceedings of INFOCOM'03 [1]. This work was supported in part by NSF grants ANI-9986397 and ANI-0093296.

A. Our Work in Context

There has been a significant amount of previous work on TCP-friendly multiple rate multicast congestion control, including [5], [6], [7], [8], [2], [9]. All of these approaches employ layered multicast, i.e. they employ a set of multicast groups that transmit at different rates to accommodate a heterogeneous, and potentially large population of receivers. Previous work has categorized these schemes as either using static or dynamic layers. In static schemes, such as [2], [8], the sending rate of any given layer remains fixed over time, and all adjustments to the reception rate are therefore exclusively receiver-driven. This approach has some drawbacks, most notably that the receiver may have insufficient information to accurately conduct join attempts, as well as necessitating abrupt rate changes. Many other schemes use dynamic layers, or layers whose transmission rate changes over time according to a predetermined pattern. Dynamic layers have been used in a variety of clever ways, including implicit coordination of receivers behind a bottleneck [9], reduction of IGMP leave messages [5], simulation of additive increase [7], and to achieve equation-based congestion control [10]. However, implementations of these dynamic layering schemes typically have a great deal of embedded complexity to realize these benefits in practice.

One feature shared by most existing multiple rate methods is that the layer rates are *non-adaptive*, i.e. the schedule of packet transmissions on each group (whether fixed-rate or dynamic) is known to the sender and to the receivers in advance. A limitation of non-adaptive schemes is their inflexibility; there are typically only a small constant number of feasible control actions that may be taken by a receiver at a given time step. For example, in many non-adaptive schemes, the magnitude by which a receiver may instantaneously increase or decrease its rate is fixed a priori, and the times at which a rate increase can be performed are widely separated. Our work differs in this regard, since each of the single congestion control sessions comprise the individual layers adaptively and continuously adjust their rates to the limiting receivers in the session, as we will describe.

Two methods for adaptive, layered multiple rate multicast were proposed in SAMM [11] and HALM [12]. However, the methods proposed in SAMM predated current notions of TCP-friendliness and were not evaluated in that context, moreover, extra router support to monitor the available bandwidth is required to achieve the best performance. The work in HALM is most similar to our own, in that they advocate periodic, adaptive reallocation of layer rates in a multirate multicast session and build upon single rate congestion control methods. In their case though, the emphasis is on periodic optimization of the layer rates at coarse time scales (tens of seconds) that is not suitable for fine-grained congestion control on the Internet.

The remainder of this paper is organized as follows. In Section II, we outline a general approach to building a multirate congestion control from a single rate scheme. We then review the two single rate congestion control mechanisms that we subsequently leverage, TFMCC and pgmcc, in Section III. In Section IV, we specify SMCC, which orchestrates an ensemble of TFMCC sessions to build a multirate congestion control scheme. We also build an alternative multirate scheme by employing the underlying pgmcc congestion control in Section V. In Section VI, we propose a new additive increase join attempt which is performed by each receiver before joining the next layer. In Section VII, we give the results from `ns` simulations to demonstrate the fairness of SMCC with competing TCP flows.

II. Overview of our approach

We describe a new multiple rate congestion control algorithm for cumulative layered multicast sessions that employs a single rate congestion control as the primary underlying control mechanism for each layer. The high-level features of our approach are as follows:

- Each receiver subscribes to a set of cumulative layers. We refer to a receiver as being an *active* participant in the uppermost layer to which it subscribes, and a *passive* participant in all other layers.
- Each layer i transmits at a rate within a designated interval and the rate floats within that interval according to a single rate congestion control regulated by active participants in that layer.
- The *lead receiver* (LR) for each layer is defined as the receiver with lowest throughput on that layer. The LR for each layer is selected from among the active participants of that layer to adjust the sending rate.
- Each receiver joins the next layer to increase the reception rate when its target rate is larger than the maximum rate available from its currently subscribed layers.
- Each receiver leaves its highest layer to decrease the reception rate when its target rate is lower than the minimum rate available from its currently subscribed layers.

Figure 1 briefly depicts a hypothetical configuration of layers in which layer rates follow a conventional

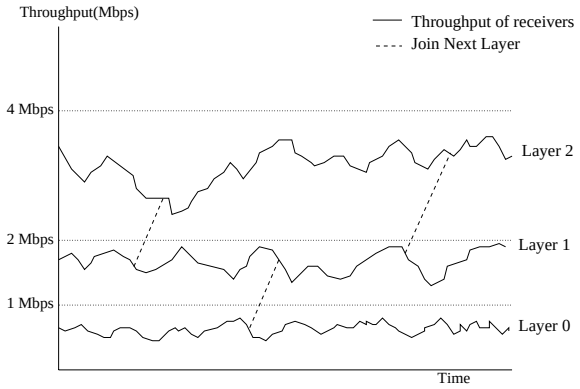


Fig. 1. Overview of Multirate Congestion Control From Single Rate Scheme

Term	Description
B_i	the maximum cumulative sending rate up through layer i
LR_i	lead receiver (LR) on layer i
L_i	layer i
R_i	actual sending rate on layer i
S_j	subscription level of receiver j
T_i	aggregate target rate requested by LR_i

TABLE I
TERMS DESCRIPTION

doubling scheme. Time elapses on the x-axis, while the y-axis indicates the cumulative sending rate from the subscribed layers. Here, receivers subscribing only to the base layer (layer zero) experience a rate which dynamically fluctuates between zero and 1Mbps. Receivers subscribing to both the base layer and Layer 1 receive a cumulative rate which dynamically fluctuates between 1Mbps and 2Mbps. In addition, receivers may employ transitional layers to transition between cumulative subscription levels. With our methods, adding a layer is done conservatively using additive increase, thus the diagonal transitions between layers depicted in Figure 1. Decrease is achieved by dropping the topmost layer (not depicted) — a practice common to most multiple rate schemes. The specifics of all of these procedures depend on the underlying single rate congestion control, and we detail these in Sections IV and V.

Our approach is quite general, and is applicable to many of the single rate congestion congestion control schemes that have been proposed. However, there are several requirements that a single rate scheme must satisfy in order for our generalized multiple rate methods to leverage it. We enumerate these needed properties from a single rate congestion control on any given layer below:

- The sending rate is controlled by the receiver with lowest throughput. This rate control removes the possibility of improper aggregation of feedback which may cause the so-called *drop-to-zero* problem [13].
- Since our methods use many LRs (one for each layer in the multicast session), it is imperative that the single rate methods for LR selection are as efficient as possible.
- Each receiver should be able to estimate its target rate. This estimation of target rate will guide the decision of subscription level change in the multirate scheme.

A. Setting up Layers and LRs

We now discuss the layer rate organization in more detail. Recall that we employ a cumulative layering scheme so that each receiver subscribes and unsubscribes to layers in sequential order. For simplicity, in the following discussion and in the remainder of the paper, we will assume that the maximum cumulative sending rates through layer i, which we denote by B_i, follow the natural $1, 2, 4, 8, \ldots$ progression. Our approach is amenable to other multiplicative layer rate increases, as advocated in [5], or to finer-grained rates of increase. Table I describes the terms we use in the following sections to describe our scheme.

We define the maximum cumulative sending rates of the layers formally as follows: We let B_0 be the maximum sending rate of the base layer, and we set B_i = $2^i * B_0$ for $i \geq 1$. From this setting of the rates, we can associate each desired reception rate with a set of subscription layers: a receiver j desiring rate r_j should subscribe to all layers i such that $B_i \leq 2r_j$. In addition, a receiver which has a computed throughput in the range [0,B_0] always subscribes to the base layer L_0. In this sense, we can map each receiver to the layer on which they are active. We say that layer L_i is *responsible* for receivers with rates in the range $[B_{i-1}, B_i]$. Equivalently, we define the *subscription level* S_j of receiver j to be the layer responsible for that receiver. Therefore, the subscription level of receiver with expected throughput x is: $S_j = \left\lceil \log_2 \frac{x}{B_0} \right\rceil$. For example, a receiver with expected throughput 6 Mbps where B_0 = 1 Mbps has a subscription level of 3 (i.e. it subscribes to L_0, L_1, L_2, and L_3) and L_3 is responsible for this receiver. At any instant in time, we let LR_i denote the slowest receiver of a given layer i, i.e. the active receiver j that has the lowest expected throughput in the range $[B_{i-1}, B_i]$.

B. Adaptively Adjusting Layer Rates

We conclude our overview with some remarks about dynamic, adaptive rate adjustment at the sender. It is first important to draw a distinction between rate-based control and window-based control, both of which are potentially applicable to our methods, provided they meet the additional assumptions stated above. In a rate-based scheme, the sender regulates the traffic by adjusting the transmission rate according to some network feedback mechanism. In a window-based scheme, a congestion window size is computed either at the sender or at the receiver(s). The sender can then send as many packets as the congestion window size allows, while the size of the congestion window changes dynamically in the presence of congestion.

Our methods are simplest when the underlying single rate control is rate-based, and our following discussion assumes this. However, with somewhat more careful consideration, it also applies to window-based schemes, and we will discuss this case in section V in the context of leveraging pgmcc.

We consider the setting of the layer rates, starting with the base layer, L_0. The sender adjusts the sending rate of the base layer based on the feedback sent by LR_0, the LR for the base layer, and we denote the actual sending rate on the base layer that results from this process by R_0. Receivers with expected throughput in the range of $[B_0, 2B_0]$ subscribe to L_1 as well as L_0. Let T_1 denote the total aggregate rate requested by the lead receiver subscribing to L_1. Then, the actual sending rate R_1 on layer 1 is set to the *difference* between T_1 and R_0. In general, the same principle is used to set the rate R_i on layer i:

$$R_i = T_i - \sum_{j=0}^{i-1} R_j, \tag{1}$$

where T_i is the aggregate target rate requested by LR_i, the lead receiver on L_i. From this setting, it is easy to show the following bounds on R_i:

$$\forall i : 0 \leq R_i \leq B_i - B_{i-2}.$$

At this point, we have provided a very high-level sketch of a general method to leverage a single-rate control in the design of a multirate multicast congestion control. But before describing the details of our protocols, we first give some additional description of the two main single rate congestion control protocols that we consider.

III. Single Rate Congestion Control

We briefly describe well designed and tested single rate multicast congestion controls that we leverage: TFMCC and pgmcc.

A. TFMCC Overview

TFMCC [4] is a single rate multicast congestion control protocol designed to provide smooth rate change over time. TFMCC extends the basic equation-based control mechanisms of TFRC [14] into the multicast domain. The fundamental idea is to have each receiver evaluate a control equation (Eqn. 2) derived from the model of TCP's long-term throughput [15], then use this to directly control the sender's transmission rate.

$$T_{TCP} = \frac{s}{RTT\left(\sqrt{\frac{2p}{3}} + (12\sqrt{\frac{3p}{8}})p(1+32p^2)\right)} \tag{2}$$

where T_{TCP} is a function of the steady-state loss event rate p, the TCP round-trip time RTT, and the packet size s.

A cursory overview of TFMCC functionality is as follows:

- Each receiver measures the packet loss rate.
- The receiver measures or estimates the round-trip time to the sender.
- The receiver uses the control equation (Eqn. 2) to derive an acceptable transmission rate from the measured loss rate and round-trip time.
- The receiver sends the calculated transmission rate to the sender.
- A feedback suppression scheme (additional details below) is used to prevent feedback implosion while ensuring that feedback from the slowest receiver always reaches the sender.
- The sender adjusts the sending rate from the feedback information.

In TFMCC, the receiver that the sender believes currently has the lowest expected throughput of the group is selected as the *current limiting receiver* (CLR). The CLR sends continuous, immediate feedback to the sender without any suppression, so the sender can use the CLR's feedback to adjust the transmission rate. In addition, any receiver whose expected throughput is lower than the sender's current rate sends a feedback message, and to avoid feedback implosion, biased feedback timers in favor of receivers with lower rates are used.

1) Measuring the Loss Event Rate: One crucial detail of TFMCC which we will return to later in the paper is the method it uses to measure packet loss. In TFMCC, a receiver aggregates the packet losses into *loss events*, defined as one or more packets lost during a round-trip time. The number of packets between consecutive loss event is called a *loss interval*. The average loss interval

size can be computed as the weighted average of the m most recent loss intervals $l_k, ..., l_{k-m+1}$:

$$l_{avg}(k) = \frac{\sum_{i=0}^{m-1} w_i l_{k-i}}{\sum_{i=0}^{m-1} w_i}$$

The weights w_i are chosen so that very recent loss intervals receive the same high weights, while the weights gradually decrease to 0 for older loss intervals. The loss event rate p used as an input for the TCP model is then taken to be the inverse of l_{avg}. The interval since the most recent loss event is incomplete, since it does not end with a loss event, but it is conservatively included in the calculation of the loss event rate if doing so reduces p:

$$p = \frac{1}{\max(l_{avg}(k), l_{avg}(k-1))}.$$

2) Round-trip Time Measurements: Each receiver starts with an initial RTT estimate that is used until a real measurement is made. A receiver measures the RTT by sending timestamped feedback to the sender, which then echoes the timestamp and receiver ID in the header of a data packet. An exponentially weighted moving average (EWMA) is used to prevent a single large RTT measurement from greatly impacting the sending rate:

$$t_{RTT} = \beta \cdot t_{RTT}^{inst} + (1 - \beta) \cdot t_{RTT}.$$

The recommended value of β for the CLR is 0.05 while all other receivers are recommended to use $\beta = 0.5$ due to their less infrequent RTT measurements. For further details of TFRC and TFMCC, we refer the reader to [14] and [4].

B. pgmcc Overview

pgmcc [3] is a single rate congestion control using a window-based controller that closely resembles the control used by TCP based on positive ACKS sent by a multicast group's representative. The high-level features of pgmcc are as follows:

- Each receiver measures its own loss rate.
- Loss rate information is periodically delivered to the sender inside negative acknowledgments (NACKs).
- The sender selects a group's representative, *acker*, as the receiver with the worst throughput according to the RTT and the reported loss rate.
- The acker sends an ACK for each received packet to the sender. The sender runs the window-based control scheme to mimic TCP congestion control.
- NACK suppression (optional) can be performed with randomization and routers which support the PGM multicast control traffic protocol.

In pgmcc, each receiver measures the loss rate by interpreting the packet arrival pattern as a discrete signal and passes it through a discrete-time linear filter. This loss rate (p) is used in the acker election procedure by employing a simplified TCP equilibrium equation: $T = \frac{1}{RTT\sqrt{p}}$.

In pgmcc, a receiver may not be able to obtain an accurate round-trip time estimate, and thus a different RTT estimate is employed. Since the RTT is used only for the acker selection purpose, the number of packet in flights is used instead of real RTT. Each receiver sends the highest known sequence number on NACK and this information can be used to compute the number of packets in flight. Even though this number of packets will vary depending on the actual sending rate, ordering receivers by packets in flight is equivalent to ordering them by RTT. Thus, acker selection can safely proceed by identifying the receiver with the lowest TCP equilibrium throughput using the method described.

IV. Multirate Multicast Congestion Control from TFMCC

The primary protocol we develop is SMCC (Smooth Multirate Multicast Congestion Control) emplying TFMCC as the underlying protocol. In section V, we compare and contrast the SMCC design with a design using pgmcc as the underlying control. TFMCC and pgmcc use different terms (CLR vs. acker) for the receiver with lowest throughput in a multicast group, so for consistency, throughout the remainder of this paper we use *lead receiver* (LR) to denote this lowest throughput receiver.

The high level features of SMCC follow the general features of deriving a multirate scheme from a single rate scheme described in Section II. The additional specific features for SMCC are as follows:

- Each receiver calculates its expected throughput as in TFMCC.
- If the expected throughput calculated from the equation is above the maximum sending rate of its current subscription level, the receiver performs a join attempt using additive increase methods.
- If a receiver's computed throughput is below the minimum receiving rate of the layer i, it drops its highest layer i. (Note that this bounds the extent to which an LR can drag down a single TFMCC session).

A. Lead Receiver Change

As in the TFMCC approach, the active participants in L_i do not send feedback unless their calculated rate

is less than T_i, thus avoiding feedback implosion. The LRs are permitted to send immediate feedback without any form of suppression, so the sender can use the LRs' feedback to adjust the transmission rate (upward or downward) for each layer.

The LR for a layer can change in one of two ways: either a new receiver becomes the LR or the existing LR leaves the group. Each of these cases is relatively easy to handle. If a receiver whose subscription level is i sends feedback that indicates a rate that is lower than the current rate of LR_i, but still larger than B_{i-1}, the sender will set LR_i to that receiver and immediately reduces its rate for L_i to the requested rate in the feedback message according to Equation (2). If a receiver on L_i has a calculated rate which is less than B_{i-1}, it unsubscribes from layer L_i. The receiver needs to issue one IGMP leave message to drop the layer. While dropping the highest layer does not guarantee a particular amount of multiplicative decrease, on average, the reception rate is decreased by half.

If the departing receiver is the LR on L_i, a new LR for layer i must be elected. To accomplish this, a departing LR first sends a control message to the sender notifying it of the departure. Upon receipt of this signal, the sender multicasts a control message to the group asking active participants to select a new LR. As in TFMCC, each receiver which is an active participant on layer L_i will set a random timer before sending feedback to the sender. To avoid feedback implosion, biased feedback timers in favor of receivers with lower rates are used.

If there are no active participants on layer i (which can happen when other participants are active on other layers j such that $j > i$), no LR is assigned to layer i. The actual sending rate of layer i is then set to $(B_i - \sum_{j=0}^{i-1} R_j)$ and the rates on higher layers are adjusted according to Equation (1). If any receiver which is active in layer $j > i$ subsequently drops layers $i+1$ through j and becomes active in layer i, this receiver will become the LR in layer i, as will a receiver who joins layer i from below. The sending rate of layer i is then adjusted by this active receiver's feedback rate.

B. Subscribing to an Additional Layer

Even though the receivers in the same group have similar calculated throughput, they may not share the same congested links. So, measured packet loss events across active receivers in a layer will vary. Often, some receivers may compute a calculated throughput value which is in the range of the next layer, and those receivers will attempt to join the next layer. As motivated in the introduction and in related work [2], naive join attempts using a single IGMP join request are problematic, as they introduce a sudden rate increase along a network path. Such a spurious join attempt may cause significant packet loss prior to the time at which the attempt is rescinded [5]. In severe cases, this substantial increase on the bottleneck link may drive TCP flows into timeout. For this reason, join attempts which mimic fine-grained additive increase are preferable [6], [7]. Here, instead of joining the next layer, the receiver increases the receiving rate slowly, i.e. by one more packet per RTT, during the join attempt.

Another compelling reason for proceeding to the next layer slowly is due to inaccuracies in estimating the target throughput when it differs substantially from the current reception rate using TFMCC methods. As described earlier in section III-A.1, the loss rate is computed from the loss interval, which is defined as the number of received packets since the last loss event. Hence, the loss interval clearly depends on the sending rate. But since the sending rate is controlled by the LR's feedback, the loss rate currently measured by a non-LR is not the same as if the sending rate adjusted to its feedback. In section VII, we show simulation results demonstrating that the loss rate measured by non-LR is not a sufficiently accurate estimate to conclusively determine whether or not to join the next layer. In practice, depending on the specific scenario considered, the calculated throughput can either be an overestimate or an underestimate.

Our methods for performing additive increase joins are the glue that holds an ensemble of TFMCC sessions together, and constitute the key additional feature needed to provide a sound multiple rate congestion control scheme. As such, we describe them fully in Section VI.

V. Multirate Congestion Control from pgmcc

We next describe a multirate scheme employing pgmcc as the underlying control protocol to offer a contrasting perspective to the use of TFMCC. Two main differences impacting the designs are the use of window-based vs. rate-based control, and the differing responses to packet loss. The multirate scheme using pgmcc follows the property of AIMD in pgmcc, so a receiver with even one packet loss will drop its highest layer to follow multiplicative decrease. This is guaranteed to cause a subscription level change; whereas with SMCC, occasional packet loss can often be absorbed without rate reduction.

All receivers start by subscribing to the base layer, and as in pgmcc, the lowest throughput receiver will be selected as LR and the congestion window will be controlled by the ACKs and NACKs sent by this LR

on the base layer. The sender maintains the congestion window per layer and this window is increased by the ACK sent by the LR for that layer. The non-LRs on the base layer need to decide whether they should join the next layer or not since they may have the higher expected target rate than the maximum sending rate of base layer. Since in pgmcc, the loss rate and RTT measurement are used only for the acker selection purpose, we employ the well-tested loss rate and RTT measurement mechanisms used in TFRC and TFMCC to compute the target rate. Using these methods, any non-LR will join the next layer if the target rate is larger than the maximum sending rate on the base layer. If there is no LR on the subscribed layer i, this receiver will be selected as LR_i and its initial window is set to $B_{i-1} * RTT/S$ where RTT is LR_i's RTT and S is the packet size.

Since the cumulative window-based scheme does not provide the expected linear increase like TCP (i.e. increase by one for each round-trip time) when the receiver subscribes multiple layers, we employ the following mechanism to provide the proper linear increase from multiple layers. The sender computes the average sending rate from the window size and the round trip time. This sending rate on layer i is set to $T_i = W_i * S/RTT$, where W_i is the congestion window size on layer i, and S is the packet size. T_i is the aggregate target rate for layer i and it is used to set the actual sending rate (R_i) described in II.

The LR for a layer L_i can change in one of three ways: 1) a receiver who joins layer L_i from below becomes the LR, 2) a receiver who leaves layer L_{i+1} becomes the LR, or 3) the existing LR leaves the layer L_i. Whenever there is a LR change on layer i, the window size for layer i will be readjusted by the target rate reported by LR$_i$. If the window size of layer i increases up to the maximum sending rate of layer i and there is a subscribed receiver on the next layer $i+1$, all active participant on layer i will join the next layer $i+1$ through the additive join attempt. If there is no receiver on layer $i+1$, LR$_i$ will become LR$_{i+1}$ and the actual sending rate of layer i is then set to $(B_i - \sum_{j=0}^{i-1} R_j)$. All receivers on layer i for $i > 0$ will leave its highest layer at the detection of packet loss and this leave will reduce the reception rate by approximately half.

VI. Additive Increase Join Attempts

We now describe the final technical component of our methods: a new additive increase scheme to conduct join attempts between successive layers in our multicast session. Although other work has proposed the use of additive increase in multiple rate multicast congestion control, such as FGLM [6] and STAIR [7], those methods are designed as an integral part of complex, non-cumulative multicast layering schemes, and have technical limitations which make them unsuitable for this application. In contrast, the layers we propose for additive increase are *only* used when a receiver wishes to attempt to join the next successive layer. Our scheme has the following properties.

- True additive increase with respect to end-to-end bandwidth consumption.
- Employs no IGMP messages (which can be slow to take effect).
- Uses only a small number of additional IGMP join messages.

Once a receiver performing a join attempt from layer L_i attains a total reception rate equal to T_{i+1}, the target rate sent by LR$_{i+1}$, it joins layer L_{i+1} and drops the special additive increase layers. If, however, there is a packet loss during the join attempt, the receiver ceases the join attempt. We incorporate the information gained from both successful or failed join attempts into loss interval and loss rate calculations. The sender sends the next layer rate information in the packet header.

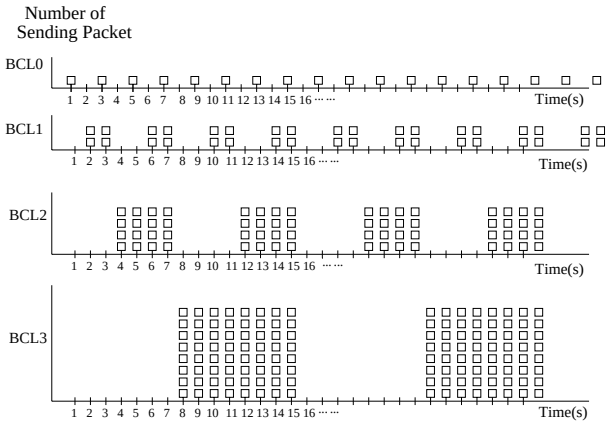


Fig. 2. Binary counting layers targeted for an RTT of 1 second

A. Introducing Binary Counting Layers

The key to our additive increase methods are binary counting layers, so named because the rates on the layers mimic aspects of counting in binary.

- *Binary Counting Layers (BCL):* The rate transmitted on $BCL_i(x)$ is an on/off function with a sending rate of 2^i packets during each on time, and where the duration of each on and off time is $x \cdot 2^i$.

In TCP, the rate of additive increase is a function of the round-trip time: the window opens by one additional packet per RTT. The set of BCL(x) layers provide the same functionality as TCP's additive increase with

a measured RTT of x seconds. BCLs accommodate asynchronous join attempts for different receivers in the multicast session, and accommodate receivers with different target rates for the join attempt.

All layers are initially synchronized at time zero, which corresponds the beginning of an off time for all layers. Figure 2 shows how each Binary Counting Layer is organized, assuming a 1 second RTT, which we use throughout this discussion for simplicity.

To achieve additive increase starting at time zero, the receiver simply subscribes to BCL_i at $2^i *$ RTT seconds. In Figure 2, where the RTT is 1 second, the receiver subscribes to BCL_0, BCL_1, BCL_2, and BCL_3 at $1s$, $2s$, $4s$, and $8s$ respectively. Once the receiver subscribes up through BCL_i, the number of receiving packets per RTT has increased by $2^{i+1} - 1$ with only i IGMP joins and no additional IGMP leaves. Avoidance of IGMP leaves is crucial, since in current versions of IGMP, it often takes a number of seconds before the leaves actually take effect; moreover, other extant methods for additive increase require use of IGMP leaves.

Previous work has defined *join* and *leave complexity*, i.e. the number of IGMP joins and leaves per operation, to be useful performance metrics for layered multicast [6]. For SMCC, the notion of operation does not map cleanly onto the additive increase process, so we will consider the complexity of N successive additive increases. From the description above, it is clear that this process requires $\log N$ joins (and no leaves). In other approaches to additive increase, such as [7], the receiver periodically increases its rate by a constant amount c using a constant number of operations (typically 1 join and 2 leaves). Thus the complexity of N successive additive increases in these schemes is N/c, i.e. linear in N. The full version of the SMCC paper [1] provides an alternative to BCLs to reduce the waiting time to join and shows how these BCLs can be organized to simultaneously accommodate receivers with various RTTs which are powers of two.

B. Cost of additional BCLs for join attempt

One cost of additional layers to facilitate additive increase is that they consume additional bandwidth beyond what is used by the normal cumulative layers. To measure this cost, we use the measure of dilation, defined in [6] and recapitulated here.

Definition 1: For a layering scheme which supports reception rates in the range $[1, R]$, and for a given link l in a multicast tree, let $M_l \leq R$ be the maximum reception rate of the set of receivers downstream of l and let D_l be the bandwidth demanded in aggregate by receivers downstream of l. The *dilation* of link l is then defined to be D_l/M_l. Similarly, the dilation imposed by a multicast session on tree T is taken to be $\max_{l \in T}(D_l/M_l)$.

Lemma 1: The worst case dilation of SMCC with single set of BCLs is 1.75.

Proof: Let us suppose the highest layer subscribed to by any downstream receiver is the jth layer. The maximum rate induced by the join attempt of a receiver k is $B_j - B_{j-2}$ when the following case holds: 1) the cumulative sending rate up through L_j is the maximum rate B_i, and 2) the cumulative sending rate up through L_{j-1} is slightly higher than the minimum rate B_{j-2}.

When an active receiver k in L_{j-1} has a calculated rate that is in the range of L_j, it performs a join attempt, which lasts until the total reception rate is equal to the next layer's cumulative sending rate B_j. Therefore, the maximum rate induced by the join attempt is $B_j - B_{j-2}$. The maximum reception rate of the set of receivers is B_j and the bandwidth demanded in aggregate by receivers is $B_j + B_j - B_{j-2}$. Therefore,

$$\text{dilation} = \frac{B_j + B_j - B_{j-2}}{B_j} = 1.75$$

■

Even though this worst-case dilation is not negligible, in practice it occurs only rarely (when a join attempt occurs across a bottleneck link); moreover, the average dilation during a join attempt is much smaller than this worst-case.

VII. Experiments

We have tested the behavior of SMCC using the ns simulator [16]. In most of the experiments we describe here, we use RED gateways, primarily as a source of randomness to remove simulation artifacts such as phase effects that may not be present in the real world. Use of RED vs. drop-tail gateways does not appear to materially affect the performance of our protocol. The RED gateways are set up in the following way: we set the queue size to twice the bandwidth-delay product of the link, set minthresh to 5% of the queue size and maxthresh to 50% of the queue size with the gentle setting turned on. Our TCP connections use the standard TCP Reno implementation provided with ns.

A. Preliminary Fairness Tests

Since the single rate TFMCC was well tested on the "dumbbell" topology [4], we set our initial topology to have multiple bottlenecks so that various SMCC receivers experience different network conditions. This

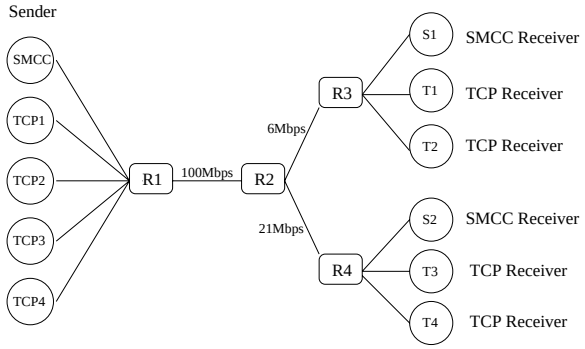


Fig. 3. Topology used to study TCP-fairness

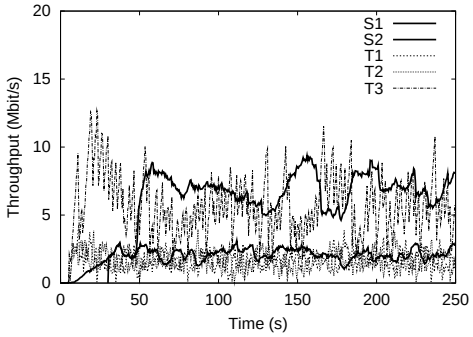
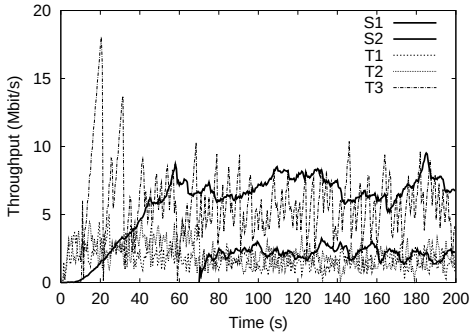
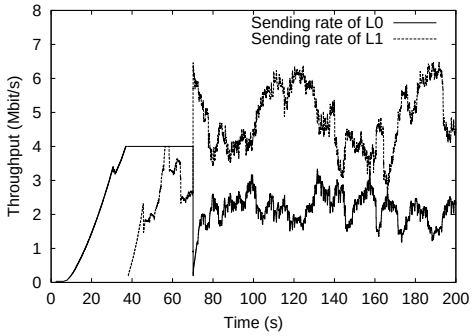


Fig. 4. Two SMCC receivers with TCP flows,$B_0 = 4Mbps$



(a) Throughput of each receiver



(b) Sending rate of L0 and L1

Fig. 5. Late join of low-rate SMCC receiver. B_0 = 4Mbps

initial topology is depicted in Figure 3. We set the propagation delay on each link is set to 8ms; each receiver therefore has a 64ms RTT in our simulations. Varying the delay on the links did not materially impact the performance of our protocol in the simulations we conducted. A full set of all the experiments we conducted as well as the ns source code are available online at http://cs-people.bu.edu/guin/smcc.html.

We consider a single SMCC session with two SMCC receivers and two parallel TCP flows sharing the same bottleneck link for each SMCC receiver. SMCC receiver S1 competes with 2 TCP connections on a 6Mbps link, giving a fair rate of 2 Mbps. S2 competes with 2 TCP flows on a 21Mbps link, for a fair rate of 7Mbps. We set B_0 to 4Mbps so that the sender's maximum transmission rate on the base layer L_0 is 4Mbps. The throughput of each of the flows is plotted in Figure 4. S2 joins the base layer L_0 at 30.0 seconds, and it performs a join attempt at 47.6 seconds. After S2 subscribes to L_1 at 48.2 seconds, it shares fairly with the parallel TCP flows on the 21Mbps bottleneck link, while low-rate SMCC 1 shares fairly with 2 TCP flows on the 6Mbps link.

B. Late Join of Low-rate Receiver

In TFMCC, a late join by a low-rate receiver results in that low-rate receiver being selected as LR, causing the sending rate of the entire session to be adjusted by its feedback. In SMCC, the late join of a low-rate receiver does not affect other receivers' throughput on higher layers. Figure 5 (a) shows the throughput of SMCC receivers when the low-rate receiver, S1, joins late.

At the time S1 joins the session (70 seconds), the transmitted rate on the base layer is the maximum 4Mbps, while the rate on L_1 has been smoothly adjusting between 1 and 4Mbps to accommodate S2. The fair share for S2 behind the 6Mbps bottleneck link with two TCP competing flows is roughly 2Mbps, thus it immediately starts to experience a high loss rate. S1 is selected as LR_0 within a second, and its feedback subsequently controls the transmission rate of L_0. While the transmission rate of L_0 has changed from 4Mbps to S1's feedback, the throughput of S2 is *not* adversely affected, since S2 is the LR for L_1, and the rate on L_1 instantaneously increases to compensate for the rate decrease on L_0. Figure 5 demonstrates the discontinuities in the sending rates across L_0 and L_1 after time 70 seconds due to the late join of the low-rate receiver.

However, had there been other receivers subscribing only to the base layer, then the late join of a low-rate receiver clearly would affect other receivers at a same subscription level. The following rule is one of the keys to the scalability of our approach: degradation in the form of additional congestion along a path to a LR

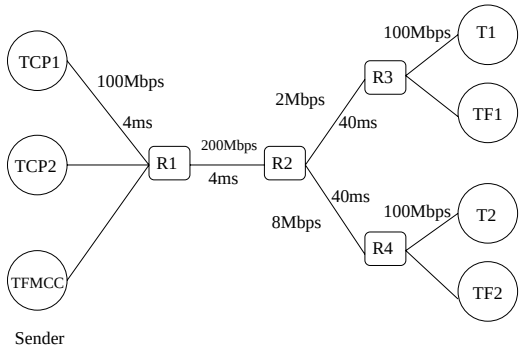


Fig. 6. Topology for calculated rate inaccuracy

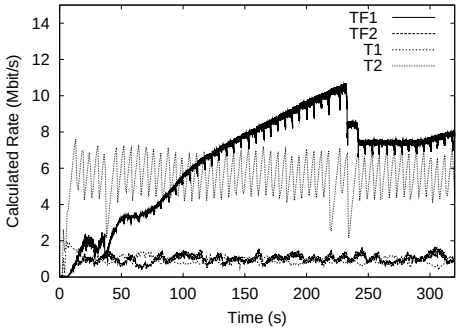


Fig. 7. Throughput of TF1 and TF2 over time. Competing TCP connections plotted in the background.

Receiver	Parameter	Time				
		50 s	100 s	150 s	200 s	250 s
TF1	RTT(second)	0.111	0.135	0.115	0.110	0.113
	Loss rate(%)	1.097	0.358	0.811	0.811	0.799
	Rate(Mbps)	0.761	1.175	0.880	0.919	0.902
TF2	RTT(second)	0.106	0.107	0.109	0.109	0.107
	Loss rate(%)	0.082	0.027	0.014	0.009	0.015
	Rate(Mbps)	3.220	5.583	7.671	9.366	7.442

TABLE II
CALCULATED TARGET RATE OF TFMCC RECEIVERS

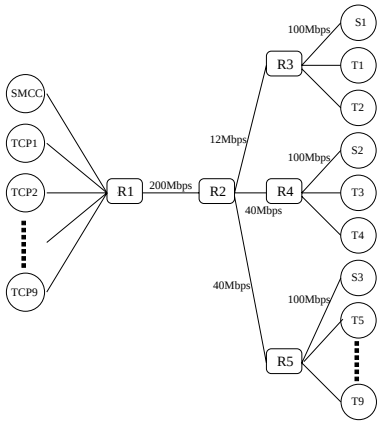


Fig. 8. Topology for assessing impact of dynamics of competing flow

will only impose throughput degradation to *receivers at the same subscription level at that time*. Rates received at other subscription levels are generally not impacted substantially.

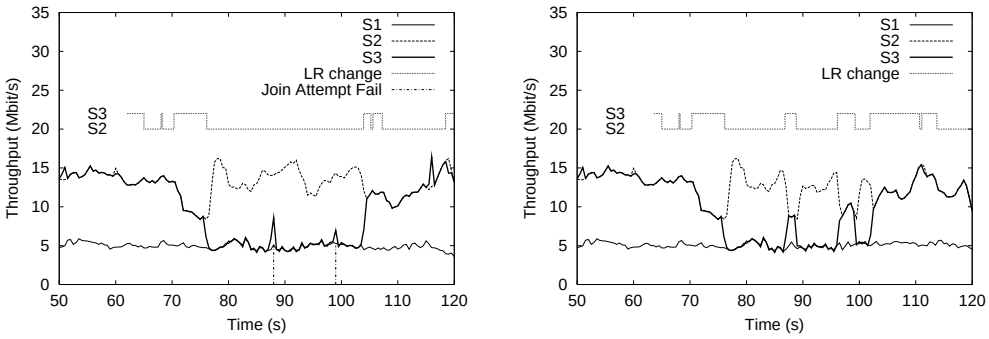
C. Inaccuracy of Non-LR Estimated Target Rate

Using TFMCC methods, a receiver which is not the LR may not have sufficient information to correctly estimate its targeted rate. In particular, the loss rate measured by non-LR receivers does not provide accurate information about the bottleneck bandwidth since the control equation was not modeled for this case, when the sender's transmission rate is independent of the receiver's packet loss events. The relevance of this point for SMCC is that a non-LR receiver may not always be able to accurately assess whether it can safely join the next layer.

Figures 6 and 7 and Table II depict this scenario. In Figure 6, TFMCC receivers TF1 and TF2 are competing with two TCP connections, T1 and T2, over a 2 Mbps bottleneck link and an 8 Mbps bottleneck link, respectively. TF1 and TF2 are not sharing the same bottleneck link, thus their losses are largely independent. Figure 7 shows each TCP flow's throughput and each TFMCC receiver's target rate calculated from the measured RTT and the loss rate. In the simulation, TF1 is quickly selected as LR_0 and it fairly shares the 2 Mbps link with T1 throughout the simulation. Indeed, TF1's target rate over time, as depicted in Table II, is a reasonable approximation to its fair rate. In contrast, TF2's target rate, also depicted in Table II, is initially inaccurate (and badly underestimates the target rate) up through time 150 seconds. It then briefly overestimates its fair rate at time 200 seconds, and also overshoots its target subscription level, before converging around time 250 seconds. These estimation inaccuracies are another reason why we recommend and use conservative additive increase join attempts.

D. Responding to dynamics of competing traffic

We used a topology (Fig 8) to test the responsiveness to dynamic changes of local competing traffic, i.e. how increased traffic on local bottleneck links affects the receivers' throughput on different bottleneck links. As the competing traffic increases across a bottleneck, proportional fairness ensures that an SMCC receiver sharing the same bottleneck will get less throughput, and in the event that receiver is selected as LR, the other receivers with the same subscription level also get less throughput even though they do not share the bottleneck with the LR. However, the extent of the degradation is bounded by a penalty of at most a factor of 2 on all layers



(a) With Additive Join Attempt (b) Without Additive Join Attempt

Fig. 9. Impact of dynamic competing traffic, B_0 = 8 Mbps

except for the base layer. Moreover, we will show that in practice, the degradation is typically much smaller than this worst-case bound.

In Figure 8, receiver S1 is competing with two TCP flows for a 12Mbps bottleneck link, while both S2 and S3 are competing with two different TCP flows for a different 40Mbps bottleneck link. We now set B_0 = 8Mbps and all receivers have an RTT of 32ms. S2 and S3 do not share the same bottleneck link but their expected throughput is initially the same. Therefore, they have the same subscription level until new competing traffic starts.

Figure 9 (a) shows the throughput of each of the three SMCC receivers over time, as well as the LR (either S2 or S3) on L_1 over time. Initially, the simulation starts with the three SMCC receivers and TCP flows 1 through 6. At 70 seconds, 3 additional TCP flows (T7, T8, T9) sharing the 40Mbps bottleneck enter the system . Therefore, S3's fair share drops from roughly 13Mbps to roughly 7Mbps. S3 is selected as LR_1 at 70.3 seconds and the sending rate for L_1 steadily decreases, once it is controlled by its feedback. The receiver with the same subscription level, S2, suffers performance degradation as it gets the packets sent at the S3 feedback rate. But S2's receiving rate is adversely affected by the increase of traffic on the path to S3 only so long as S3 is LR_1. At time 75.7 seconds, S3 drops its highest layer, L_1 when its calculated rate drops to 7.74Mbps. S2 is elected as new LR for L_1 at 76.2 seconds and its feedback controls the sending rate of L_1, which then quickly rebounds. Meanwhile, L_0 continues to be limited by S1, who continues to have a lower fair share than S3, so S3 receives at a rate of approximately 5Mbps during this time.

Although S3's fair share is only 7Mbps, for reasons described in Section IV-B, it cannot make a highly accurate assessment of its expected throughput while receiving at only 5Mbps, and these inaccurate estimates induce it to make join attempts to L_1. S3 experiences two join attempts, both of which fail due to packet loss, between 70 seconds and 100 seconds. These two join attempts, marked by small spikes away from the S1 baseline, occur at 87.1 seconds and at 98.3 seconds. The little spikes around this time indicate these join attempt failures.

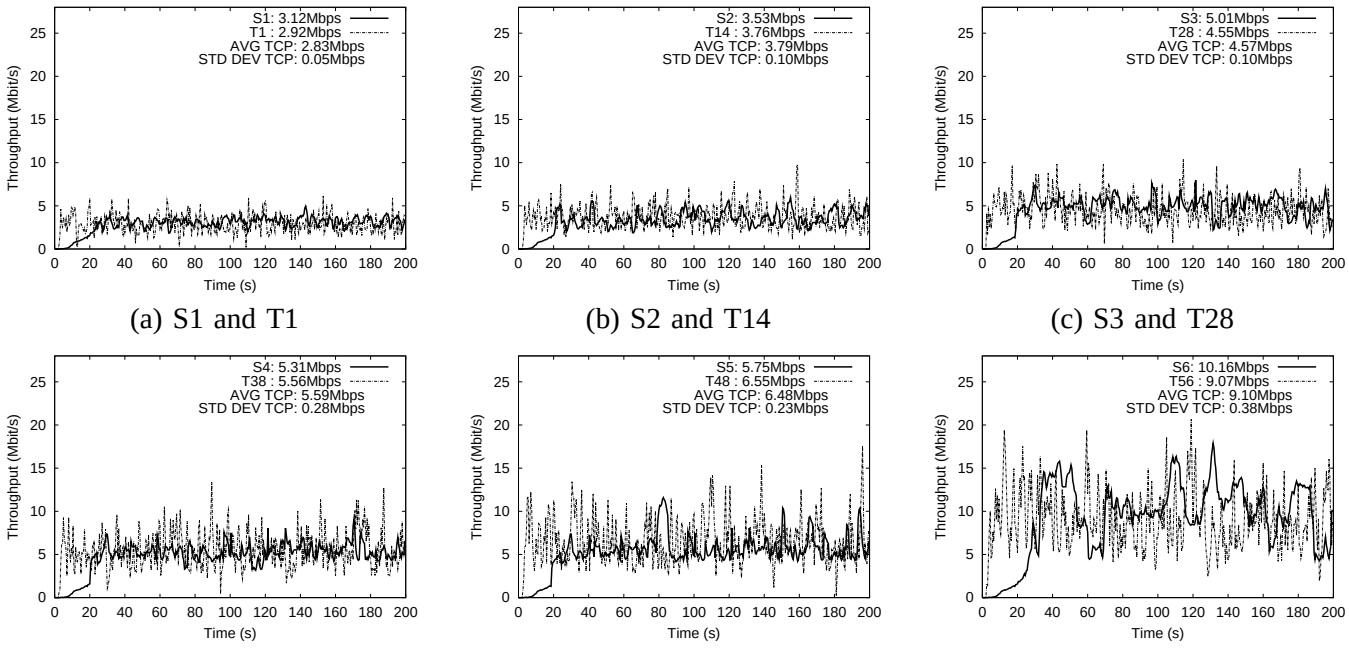
Finally, the three additional TCP flows leave at time 100 seconds. S3 performs a successful join attempt at 103.4 seconds and it reaches L_1 at 103.9 seconds, at which time it resumes sharing with S2.

Figure 9 (b) shows the identical simulation of each SMCC receiver but *without* the benefits of additive increase join attempts. Instead, in this simulation, the receiver naively joins an additional layer whenever the calculated rate is in the range of the sending rate of the higher layer. S3 joins the next layer at 86.8 seconds and it becomes LR for L_1 until 88.8 seconds. During this time, the sending rate of L_1 is dragged down to the rate of S3, impacting the reception rate of S2. After dropping back down, S3 joins L_2 at 96.1 seconds again and it is selected as LR_2 until 99.3 seconds. Spurious joins such as these can cause significant performance degradation; an effect which is that much more severe when *multiple* receivers perform spurious joins, thereby constantly dragging down the rates on higher layers.

In contrast, with additive increase joins, even when a receiver initiates joins which are ultimately unsuccessful, it does not diminish the throughput received by other session participants during that time.

E. Fairness with heterogeneous receivers

Finally, we used a topology with multiple bottlenecks (Figure 10) to test the performance of SMCC with a set of heterogeneous receivers where the differences between the receivers' target rates is relatively small.



(a) S1 and T1 (b) S2 and T14 (c) S3 and T28

(d) S4 and T38 (e) S5 and T48 (f) S6 and T56

Fig. 11. Throughput of SMCC receivers, B_0 = 4 Mbps

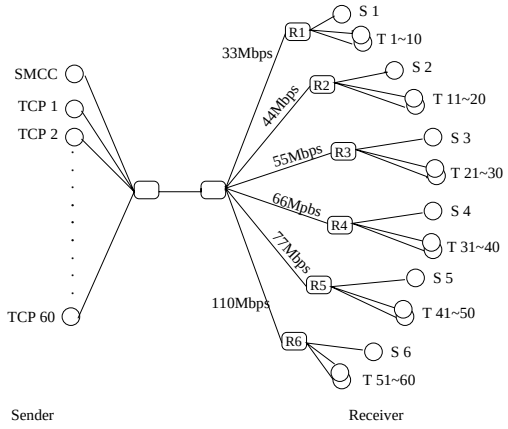


Fig. 10. Topology for Many Heterogeneous Receivers, $B_0 = 4Mbps$

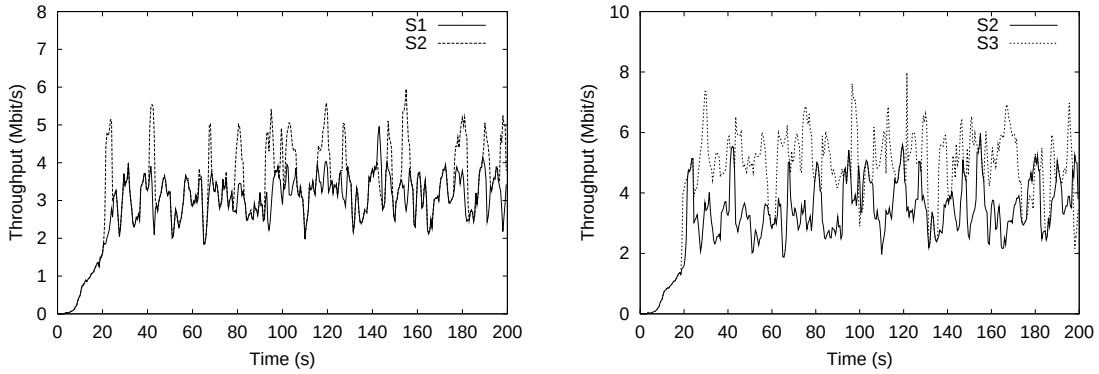
We consider a single SMCC session with six SMCC receivers and ten parallel TCP flows sharing the same bottleneck link for each SMCC receiver, but each SMCC receiver is not behind the same bottleneck link. S1 competes with 10 TCP connections on a 33Mbps link, giving a fair rate of 3 Mbps and the fair rates of the other SMCC receivers (S2 to S6) are 4Mbps, 5Mbps, 6Mbps, 7Mbps, and 10Mbps respectively.

We plot the throughput of each SMCC flow and the throughput of one of the competing TCP flows in Figure 11. In each case, we chose the TCP connection whose mean rate was closest to the average of the ten competing flows as our representative. The throughput of each SMCC receiver fairly shares the bottleneck link with the parallel TCP flows even though lower-rate receivers are often present and drag down the rate on each level. In practice, non-LR receivers tend to periodically join the next higher layer as their estimated throughput begins to deviate substantially from the LR's target rate. The receiver S6 in panel (f) of Figure 11 is an example of relatively frequent subscription changes; note that its performance is still not as bursty as the competing TCP connection.

Next, consider Figure 12 (a) which plots the reception rate of S1 and S2. Like S6, S2 has relatively high subscription changes since its fair rate of 4Mbps is equal to B_0. S1, with a fair rate of 3Mbps, is typically selected as the LR on L_0. Whenever S2 joins L_1, it quickly becomes the LR and may impact the throughput of receivers on that layer. The plot depicted in Figure 12 (b) shows this impact. For example, at time 67.08 seconds, S2 becomes LR_1 and drags the cumulative rate T_1 down from 6.7Mbps to 5.0Mbps. At 99.6 seconds and 176.4 seconds, the sending rate is set from 7.1Mbps to 4.1Mbps and from 6.8Mbps to 4.6Mbps, respectively. There are other cases where the newly joined S2 becomes LR on L_1, but its degradation of rate is within 10%.

VIII. Conclusion

We have presented a multirate multicast congestion control design that leverages proven single rate congestion control methods by orchestrating an ensemble of



(a) Throughput of S1 and S2 (b) Throughput of S2 and S3

Fig. 12. Throughput Comparison of S1, S2, and S3

independently controlled single rate sessions. A compelling argument for this new methodology is its evident simplicity: unlike all other viable multiple rate congestion control protocols, ours requires only a small amount of carefully crafted new functionality. By maintaining appropriate invariants on the session rates of individual TFMCC flows, specifying a clean mapping from reception rates to subscription levels and providing a non-disruptive method for additive increase join attempts, we build a sound multiple rate multicast congestion control scheme called SMCC. A final advantage of our approach is its modular design; TFMCC or pgmcc could easily be replaced by an improved equation-based rate or window-based control mechanism.

REFERENCES

[1] G. Kwon and J. Byers, "Smooth multirate multicast congestion control," in *Proc. of IEEE INFOCOM '03*, 2003, Full version appears as BU-CS-TR-2002-025, Boston University, 2002.

[2] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-Driven Layered Multicast," in *Proc. of ACM SIGCOMM '96*, August 1996.

[3] L. Rizzo, "pgmcc: A TCP-friendly single-rate multicast congestion control scheme," in *Proc. of ACM SIGCOMM '00*, 2000.

[4] J. Widmer and M. Handley, "Extending equation-based congestion control to multicast applications," in *Proc. of ACM SIGCOMM '01*, 2001.

[5] J. Byers, G. Horn, M. Luby, M. Mitzenmacher, and W. Shaver, "FLID-DL: Congestion Control for Layered Multicast," *IEEE J-SAC Special Issue on Network Support for Multicast Communication*, vol. 20(8), pp. 1558–1570, Oct. 2002, A preliminary version appeared in NGC '00.

[6] J. Byers, M. Luby, and M. Mitzenmacher, "Fine-Grained Layered Multicast," in *Proc. of IEEE INFOCOM '01*, April 2001.

[7] J. Byers and G. Kwon, "STAIR: Practical AIMD Multirate Multicast Congestion Control ," in *Proc. of NGC '01*, 2001, Full version appears as BU-CS-TR-2001-018, Boston University, 2001.

[8] A. Legout and E. Biersack, "PLM: Fast convergence for cumulative layered multicast transmission schemes," in *Proc. of ACM SIGMETRICS*, 2000.

[9] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like Congestion Control for Layered Multicast Data Transfer," in *Proc. of IEEE INFOCOM '98*, April 1998.

[10] M. Luby, V. Goyal, S. Skaria, and G. Horn, "Wave and Equation Based Rate Control Using Multicast Round Trip Time," in *Proc. of ACM SIGCOMM '02*, 2002.

[11] B. J. Vickers, C. Albuquerque, and T. Suda, "Source-adaptive multilayered multicast algorithms for real-time video distribution," *IEEE/ACM Transactions on Networking*, vol. 8, no. 6, pp. 720–733, 2000.

[12] J. Liu, B. Li, and Y. Zhang, "A Hybrid Adaptation Protocol for TCP-friendly Layered Multicast and Its Optimal Rate Allocation," in *Proc. of IEEE INFOCOM '02*, June 2002.

[13] B. Whetten and J. Conlan, "A Rate Based Congestion Control Scheme for Reliable Multicast," technical White Paper, GlobalCast Communications, October 1998.

[14] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. of ACM SIGCOMM '00*, 2000.

[15] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 133–145, Apr. 2000.

[16] ns: UCB/LBNL/VINT Network Simulator (version 2). Available at http://www-mash.cs.berkeley.edu/ns/ns.html.