2002

Smooth Multirate Multicast Congestion Control

Kwon, Gu-In

Boston University Computer Science Department

Smooth Multirate Multicast Congestion Control

Gu-In Kwon   John W. Byers

guin@cs.bu.edu   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 SMCC, a new multiple rate equation-based congestion control algorithm for layered multicast sessions that employs TFMCC as the primary underlying control mechanism for each layer. SMCC combines the benefits of TFMCC (smooth rate control, equation-based TCP friendliness) 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 [8]) 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 leads to very complex multiple rate congestion control designs.

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 [11] and TFMCC [13]. 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 low 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.

A. Our Work in Context

There has been a significant amount of previous work on TCP-friendly multiple rate multicast congestion control, including [1], [2], [3], [6], [8], [12]. 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 [8], [6], 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. Dynamic layers have been used in a variety of clever ways, including implicit coordination of receivers behind a bottleneck [12], reduction of IGMP leave messages [1], simulation of additive increase [2], and to achieve equation-based congestion control [7]. However, implementations of these dynamic layering schemes typically have a great deal of embedded complexity to realize these benefits in practice.

One feature shared across all 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. Our work differs in this regard, since each of the TFMCC sessions comprising the individual layers adaptively adjust their rates to the limiting receivers in the session, as we describe momentarily.

B. Contributions and Organization

We describe SMCC, a new multiple rate equation-based congestion control algorithm for cumulative layered multicast sessions that employs TFMCC as the primary underlying control mechanism for each layer. Since each layer is controlled independently by TFMCC, the properties of TFMCC hold for participants in any given layer. As such, the layer rates are both dynamic and adaptive. SMCC combines the benefits of TFMCC (smooth rate control, equation-based TCP friendliness) with the scalability and flexibility of multiple rates to provide a sound multiple rate multicast congestion control policy. In SMCC, the receivers cumulatively subscribe to appropriate layers based on its estimated rate using the TCP throughput equation [10] also employed in TFMCC. In addition to the TFMCC functionality, SMCC provides a new additive increase join attempt to avoid abrupt rate increases when the receiver attempts to subscribe to an additional layer. However, the calculated throughput using equation-based methods may not provide a sufficiently accurate indication to decide to join the next layer. To avoid these problems, the receiver in SMCC joins the next layer through the join attempt when its calculated throughput is in the range of next layer rate. Ultimately, the smooth rate change of SMCC

is ideally suited to streaming multimedia applications; but equation-based methods are general-purpose, thus SMCC works naturally for other multicast applications, such as reliable downloads [12], [4]. Finally, it is worth emphasizing that SMCC requires no additional router support beyond basic multicast functionality, and does not place any new demands on any existing multicast protocols.

The remainder of this paper is organized as follows. In Section II, we review the underlying TFMCC congestion control mechanism. In Section III, we specify SMCC and how to orchestrate an ensemble of TFMCC sessions to build a multirate congestion scheme. In Section IV, we propose a new additive increase join attempt which is performed by each receiver before joining the next layer. In Section V, we give the results from ns simulations to demonstrate the fairness of SMCC with competing TCP flows.

II. TFMCC Overview

TFMCC [13] is a single rate multicast congestion control protocol designed to provide smooth rate change over time. TFMCC extends the basic control mechanisms of TFRC [5] into the multicast domain, using equation-based methods. The fundamental idea is to have each receiver evaluate a control equation (Eqn. 1) derived from the model of TCP's long-term throughput [10], then use this to directly control the sender's transmission rate.

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

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. 1) 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.

A. Measuring the Loss Event Rate

One critical 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:

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

B. Round-trip Time Measurements

Each receiver starts with initial RTT and this initial RTT 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 on the sending rate.

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

$\beta_{CLR} = 0.05$ is set for the CLR while $\beta_{non-CLR} = 0.5$ is used for non-CLR receivers due to infrequent RTT measurements. One-way delay adjustments are used by non-CLR receivers between the real measurements. For further details of TFRC and TFMCC, we refer the reader to [5] and [13].
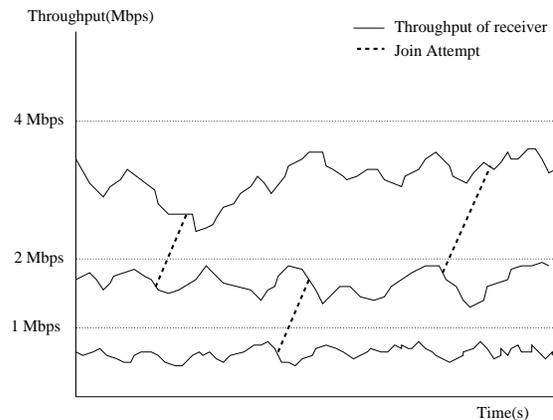

Fig. 1. SMCC with $B_0 = 1Mbps$

III. SMOOTH MULTIRATE MULTICAST CONGESTION CONTROL

Like many other multiple rate congestion control schemes, SMCC employs cumulative layered multicast. The crucial distinction is that unlike other schemes, SMCC employs *adaptive* layers, i.e. each individual layer uses TFMCC congestion control and each receiver subscribes to appropriate layers based on its calculated equation-based rate, as we describe momentarily. 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 of SMCC transmits at a rate within the interval $[0, B_i]$ and the rate floats within that interval according to TFMCC congestion control regulated by active participants in that layer.
- The current limiting receiver (CLR) for each layer is selected from among the active participants of that layer to adjust the sending rate.
- Each receiver calculates its expected throughput.
- If the expected throughput calculated from the equation is in the range of the next layer rate, 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 amount the CLR can drag down a single TFMCC session).

Figure 1 briefly shows the configuration of layers and how the sending rate of each layer is set.

In the following section, we describe how to set up the layers and define how the CLR is selected for each layer. Then, in section III-B, we describe how each receiver sends feedback and how changes of the CLR on

each layer are realized. In III-C section, we briefly show why we need additive increase join attempts.

A. *Setting up Layers and CLRs*

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 sending rates B_i of layer i follow the natural $1, 1, 2, 4, 8, \ldots$ progression. Our approach is amenable to other multiplicative layer rate increases, as advocated in [1], or to finer-grained rates of increase. We define the sending rates of the layers formally as follows: Formally, 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* of receiver j, S_j, to be the layer responsible for that receiver. The subscription level of receiver with expected throughput x is:

$$S_j = \lceil \log_2 \frac{x}{B_0} \rceil.$$

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, the current limiting receiver of a given layer i (CLR_i) is the active receiver j that has the lowest expected throughput in the range range $[B_{i-1}, B_i]$.

Now we consider the dynamics of the scheme, starting with the base layer. The sender adjusts the sending rate of L_0 based on the feedback sent by CLR_0. The receivers will not send feedback unless their calculated rate is less than the current sending rate for L_0. Everything for the sender and the receiver's behavior in L_0 follows the TFMCC scheme.

If the receiver in L_0 has the expected throughput in the range of $[B_0, 2B_0]$, the receiver may subscribe to L_1. CLR_1 is defined as the receiver which has the lowest expected throughput in the above range. The sending rate of L_1, R_1, is set to the *difference* between the rate of CLR_1 and the sending rate of the base layer, R_0.

The sending rate of L_i is then defined to be R_i; where

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

and where F_i is the feedback rate sent by CLR_i.

B. *Adjusting the Current Sending Rate*

As in TFMCC, the receivers in L_i do not send feedback unless their calculated rate is less than the current sending rate of L_i, thus avoiding feedback implosion. The CLRs are permitted to send immediate feedback without any form of suppression, so the sender can use the CLRs' feedback to adjust the transmission rate (upward or downward) for each layer.

The CLR for a layer can change in one of two ways: either a new receiver becomes the CLR or the existing CLR 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 CLR_i, but still larger than B_{i-1}, the sender will immediately reduce its rate for L_i to the requested rate in the feedback message and sets CLR_i to that receiver. 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 multiplicative decrease, on average, the reception rate is decreased by half.

If the departing receiver is the CLR on L_i, a new CLR for layer i must be elected. To accomplish this, a departing CLR 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 CLR. 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 CLR is assigned to layer i. The sending rate of layer i is then set to B_i. If any receiver which is active in layer $j > i$ drops layers $i + 1$ through j and becomes active in layer i, this receiver will become a CLR in layer i, as will a receiver who joins layer i from below. The sending rate of layer i is then adjusted by this receiver's feedback rate.

C. *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 receivers in a group will vary over time. 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, 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 [1]. 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 [3], [2] are preferable. 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 TFRC methods. As described earlier in section II-A, the *loss rate* is computed from the *loss interval*, which is defined as the number of received packet since the last loss event. Hence, the loss interval clearly depends on the sending rate. But since the sending rate is controlled by the CLR's feedback, the loss rate currently measured by a non-CLR is not the same as if the sending rate adjusted to its feedback. In the experiments section, we show simulation results demonstrating that the loss rate measured by non-CLR is not a sufficiently accurate estimate to determine whether or not to join the next layer. In practice, depending on the specific scenario considered, the calculated throughput can either be over-estimated or under-estimated.

As described earlier, the receiver has to perform a join attempt when subscribing to the next layer. Once a receiver performing a join attempt from layer L_i attains a total reception rate equal to next layer sending rate, 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.

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 the following section IV.

D. Avoiding Oscillation

Recall that both loss rate and RTT are input parameters to calculate the rate, thus even slight increases in RTT cause the calculated rate to be slightly reduced. If we allow the receiver to join layer i when the receiver's calculated rate is very close to B_i, even a slight increase of RTT is enough to force the receiver to drop the highest layer so that it goes back to its previous layer. This may cause significant oscillation when the RTT fluctuates.

We employ a conservative method to avoid this oscillation. The receiver will join the next layer L_i if the calculated rate is in the range of $[\alpha \cdot B_i, B_{i+1}]$. We recommend using values of *damping factor* α of at least 1.1 to avoid oscillations. However, for applications which are intolerant of even occasional oscillations, such as some streaming multimedia applications, a more stable setting of $\alpha = 1.2$ may be more appropriate.

IV. ADDITIVE INCREASE JOIN ATTEMPTS

We now describe 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 [3] and STAIR [2], 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 on the link.
- Employs no IGMP *leave* messages (which can be very slow to take effect).
- Uses only a small number of additional IGMP *join* messages.

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. In this section, we first describe the basic scheme, then extend the basic scheme to overcome some limitations.

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

All layers are initially synchronized at time zero, which corresponds the beginning of an off time for all layers.
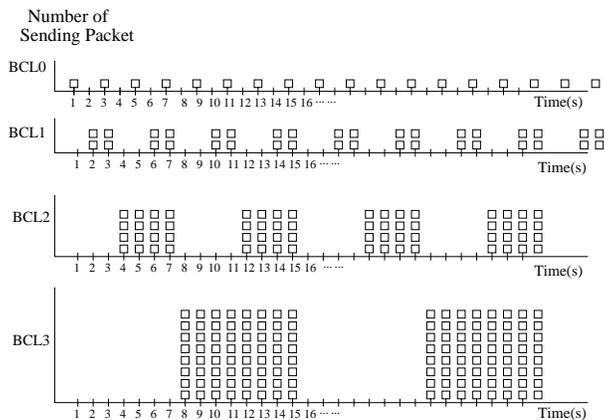
Number of
Sending Packet

Fig. 2.   Binary Counting Layers

Fig. 3.   Extended Binary Counting 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 $2^i \cdot RTT$ 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. When the receiver subscribes up to BCL_i, the number of receiving packets per RTT is automatically increased up to $2^{i+1} - 1$ 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 [3]. 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 [2], 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 is $O(N/c)$.

B. Extended Binary Counting Layers

One limitation of the basic binary counting layer scheme is that the receiver has to wait until certain specific times to join the BCLs. Suppose the receiver wants to increase the number of receiving packet from 1 to 14 packets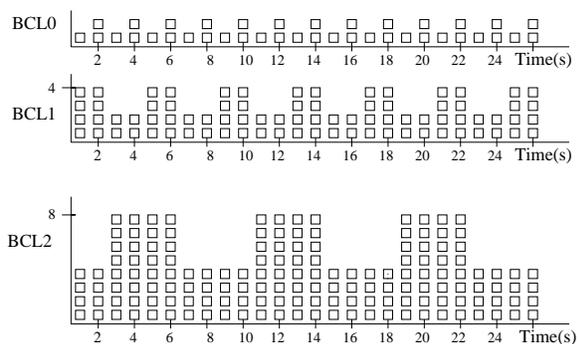 additively in Figure 2. If the receiver wants to join BCLs at 5 seconds, it has to wait until the next cycle (time 17) to initiate additive increase. One solution is to allow receivers to jump-start their additive increase with an initial set of joins (i.e. an immediate increase of 5 packets per RTT in the example above). However, this can induce sudden rate increase, and in the worst case, reduces to a naive join attempt. An alternative is the following improvement.

- *Extended Binary Counting Layers* : The rate transmitted on BCL_i is a cyclic two step function. The number of sending packet per RTT is 2^i and 2^{i+1} in the low step and in the high step respectively.

Figure 3 shows the transmission rate of each Binary Counting Layer. The receiver subscribes to BCL_i at $(2^{i+1} - 1) \cdot RTT$ second to perform the additive increase. In Figure 3, the receiver subscribes to BCL_0, BCL_1, BCL_2, and BCL_3 at $1s$, $3s$, $7s$, and $15s$ respectively to get the additive increase up to 30 packets per RTT. Now consider the waiting time if the receiver misses the join time. If the receiver has to start the increase from 1 packet to $2^i - 1$ packets, a new cycle for that increase starts at $2^i \cdot RTT$ second after the previous cycle starts in the basic BCL. However, in the extended BCL the new cycle for that increase starts at $(2^{i-1} + 1) \cdot RTT$ seconds after the previous cycle starts. For example, for the increase from 1 to 30 packets, the new cycle starts at 33 seconds and 17 seconds in the basic BCL and in the extended BCL respectively.

If the receiver just misses the right joining time, it has to wait until the next join time. To avoid this worst case, the receiver can perform the join attempt if the missed time is within 20% of the cycle time.

So far we have accommodated receivers with a specific RTT (1 sec. in our example). In practice, receivers may have widely varying RTTs, and it is desirable to simulate TCP behavior of one packet per RTT additive increase for each receiver. Extended BCL's can achieve this. In Fig-
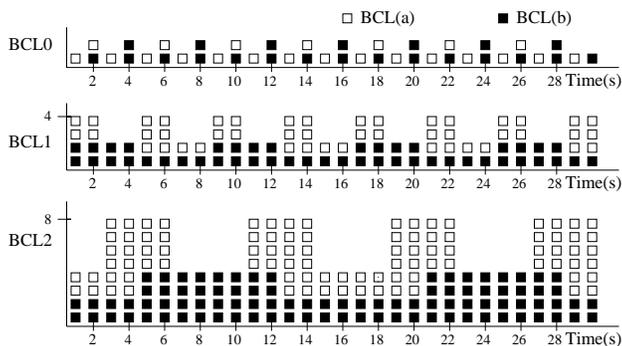
Fig. 4. Extended Binary Counting Layers with various RTT

ure 4, we show how BCL's can be organized to simultaneously accommodate receivers with RTTs of 1 second and 2 second. The scheme easily generalizes to support various RTTs which are powers of two. Each BCL_i can be represented as two multicast sessions, $BCL_i(a)$ for 1 second RTT and $BCL_i(b)$ for 2 second RTT. The packets represented as white boxes are delivered in $BCL_i(a)$ while the packets represented as black boxes are delivered in $BCL_i(b)$. The receiver with a 2 second RTT subscribes only to $BCL_i(b)$ layers, while the receiver with a 1 second RTT subscribes to *both* $BCL_i(a)$ and $BCL_i(b)$, using a cumulative approach to sublayer subscription. The effect of subscribing to both $BCL_i(a)$ and $BCL_i(b)$ is the same as subscribing to BCL_i in fig 3.

C. Proper Configuration of the Base Layer

Another consideration is that the base layer bandwidth of the base layer must be set carefully. Since TFMCC is employed in each layer, when B_0 is large, there could be a large number of receivers subscribing to the base layer and they will be dragged down to the rate specified by CLR$_0$. (This is also present in TFMCC). For this reason, we elect to set B_0 to be as small as possible. However, there is another tradeoff to consider.

In practice, the sender will maintain several BLC(i)s to emulate a range of different RTTs. However, the fixed packet size and the average maximum rate induced by the join attempt on L_0 give a lower bound on the range of RTT's that can be accommodated. The average maximum rate induced by the join attempt on L_0 is B_0. Let the number of packet per RTT t for the rate B_0 be h so that the join attempt increases the number of packet from 1 to h. Given a fixed packet size S in bytes, the height of join attempt h, and the average rate induced by the join attempt on the base layer, we require that:

$$t \geq \frac{S * 8 * h}{B_0}$$

For example, with a packet size of 1KB, $B_0 = 1$Mbps, and a desired value of $h = 4$, then the smallest allowable RTT in BCL is 32ms. As B_0 becomes smaller, the smallest allowable RTT increases. So, the small value of B_0 makes the smallest allowable RTT to increase.

D. 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 [3] 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 C_l be the bandwidth demanded in aggregate by receivers downstream of l. the *dilation* of link l is then defined to be C_l/M_l. Similarly, the dilation imposed by a multicast session on tree T is taken to be $max_{l \in T}(C_l/M_l)$.

Lemma 1: The worst case *dilation* of SMCC with BCL 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 $B_j - B_{j-2}$ in the following case. Suppose the sending rate of L_j is the maximum rate B_i and the sending rate of 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 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.

V. EXPERIMENTS

We have tested the behavior of SMCC using the ns simulator [9]. The simulation results show that SMCC exhibits good inter-path fairness when competing with TCP
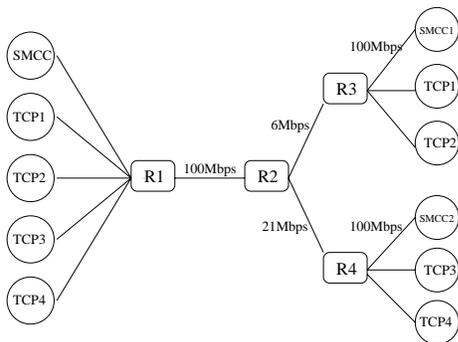
Fig. 5. Topology for Fairness

Fig. 6. Two SMCC receivers with TCP flows, $B_0 = 4Mbps$

Fig. 7. Late-join of low-rate SMCC receiver, $B_0 = 4$Mbps

traffic in a wide variety of scenarios. 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. Our TCP connections use the standard TCP Reno implementation provided with ns.

A. Fairness

Since the single rate TFMCC was well tested on the "dumbbell" topology, we set our initial topology to have multiple bottlenecks so that various SMCC receivers have different network conditions. Our initial topology is depicted in Figure5. The propagation delay on each link is set to 4ms.

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 1 competes with 2 TCP connections on a 6Mbps link, giving a fair rate of 2 Mbps. SMCC 2 competes with 2 TCP flows for a 21Mbps link, for a fair rate is 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 6. SMCC 2 joins the base layer L_0 at 30 seconds, and it performs a join attempt at 32.2 seconds. After SMCC 2 subscribes to L_1 at 32.5 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 CLR, 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 7 shows the throughput of SMCC receivers when the low-rate receiver, SMCC 1, joins late.
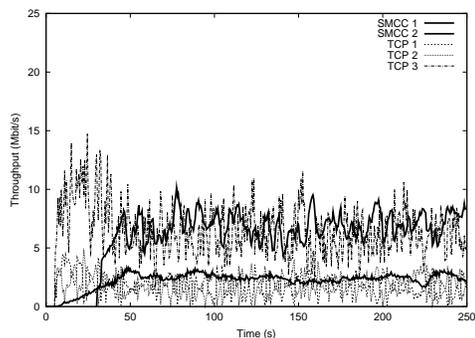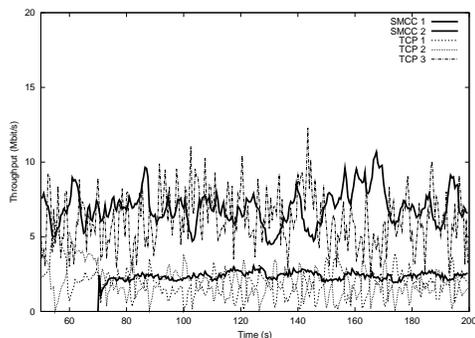
At the time SMCC 1 joins the session (70 seconds), the transmitted rate on the base layer is close to the maximum 4Mbps, while the rate on L_1 has been smoothly adjusting between 1 and 5Mbps to accommodate SMCC 2. The fair share for SMCC 2 behind the 6Mbps bottleneck link with 2 TCP competing flows is roughly 2Mbps, thus it immediately starts to experience a high loss rate. SMCC 1 is selected as CLR_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 SMCC 1's feedback, the throughput of SMCC 2 is *not* adversely affected, since SMCC 2 is the CLR for L_1, and the rate on L_1 *smoothly increases* to pick up the slack as the rate on L_0 decreases. However, had there been other receivers subscribing only to the base layer, then the late join of low-rate receiver clearly would affect other receivers at a same subscription level. This general rule is one of the keys to the scalability of our approach: degradation in the form of additional congestion along a path to a CLR will only impose a throughput degradation to *receivers at the same subscription level at that time*. Rates received at other subscription levels are generally *not* impacted.

C. Avoiding Oscillation

We find that when the equal share of an SMCC receiver is close to the boundary of layer rate, causing rela-

tively frequent subscription changes, the throughput of the SMCC receivers is only slightly less aggressive than that of the competing TCP flow. Also, increasing the *damping factor* α has the effect of reducing hysteresis but at the expense of reducing SMCC throughput somewhat.
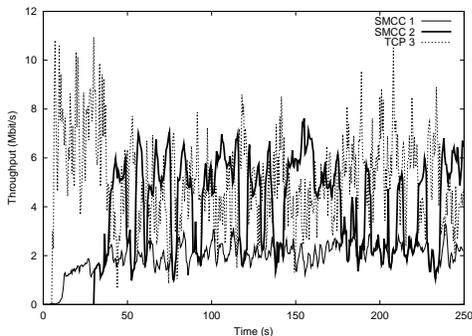

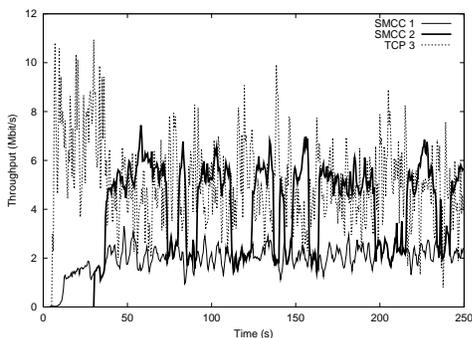Fig. 8. *damping factor* $\alpha = 1.1$

Fig. 9. *damping factor* $\alpha = 1.2$

Figure 8 shows the throughput of two SMCC receivers, using the same basic topology, but with the following adjustments on the link speeds. SMCC 1 is competing with 2 TCP connections on a 6Mbps link, while SMCC 2 is competing with 2 TCP connections on 15Mbps, for a fair rate is 5Mbps. SMCC 2 experiences oscillation of the subscription level since its equal share is quite close to the layer rate boundary (4 Mbps). Note that this oscillation does not affect the inter-path fairness of SMCC. The average throughput of SMCC 2 attained was 4.3 Mbps and the average throughput attained by TCP flows were 4.8Mbps and 5.0Mbps. Figure 9 shows the throughput of each SMCC receiver with damping factor $\alpha = 1.2$. This is representative of a large number of experiments we conducted; in this experiment, hysteresis is reduced by having increased the damping factor to 1.2. In this particular scenario, the average throughput of SMCC 2 attained was 4.3Mbps in this simulation, but a more typical result is slightly diminished throughput when the damping factor is increased.
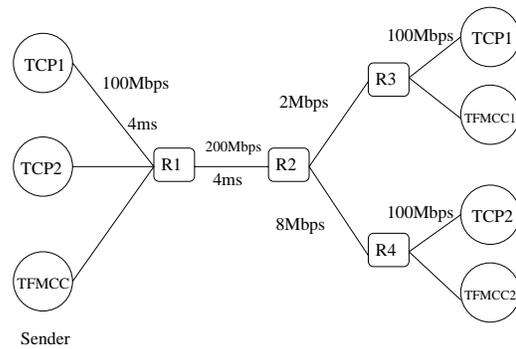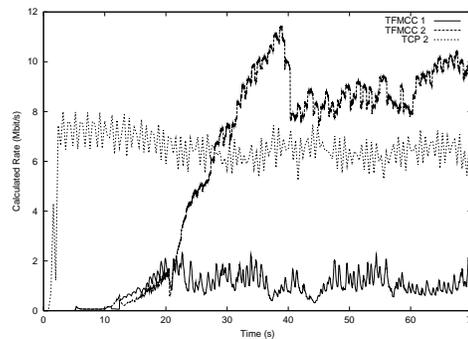

Fig. 10. Topology

Fig. 11. Inaccuracy of Non-CLR Calculated Rate

D. Inaccuracy of Non-CLR Calculated Rate

TFRC and TFMCC show the TCP-friendliness when the sending rate is adjusted by the receiver's target rate. In this section, we show that a non-CLR's calculated target rate is not the correct estimation to decide to join the next layer when the sending rate is not adjusted by a non-CLR's feedback.

In Figure 10, TFMCC 1 and TFMCC 2 are competing with two TCP connections, TCP 1 and TCP 2 over a 2 Mbps bottleneck link and 8 Mbps bottleneck link, respectively. The propagation delay on each link is set to 4ms. Figure 11 shows each TFMCC receiver's calculated rate and the throughput of the TCP 2. TFMCC 1 is selected as CLR and it is fairly sharing 2 Mbps link with TCP 1 connection (not depicted for clarity.). Since the sending rate for TFMCC session is controlled by CLR, TFMCC 2's receiving rate is controlled by TFMCC 1 in most time. In Figure 10, TFMCC 1 and TFMCC 2 are not sharing the same bottleneck link, and the losses are independent. TFMCC 2's *loss rate* is measured where it is receiving packets at TFMCC 1's target rate (around 1 Mbps) and TCP 2 is using the rest of available bandwidth on 8 Mbps link. Figure 11 shows the TFMCC 2's target rate is overestimated, and it is in fact higher than the link capacity.

Every TFMCC receiver calculates the target rate with loss rate and RTT. As described earlier, the loss rate is computed from the loss interval which is defined as
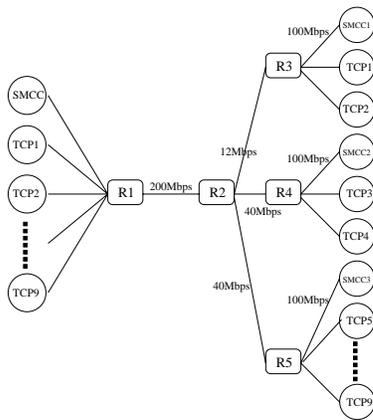
Fig. 12. Second Topology

the number of received packet since the last loss event. Hence, the loss interval clearly depends on the sending rate. Since the Non-CLR's target rate is higher than CLR's target rate, if the sending rate is adjusted by Non-CLR's target rate, then loss interval of Non-CLR will increase quicker until it induces congestion. In this case, the target rate of Non-CLR with controlled by CLR's rate is under-estimated. It may not possible to decide whether the Non-CLR's current target rate is under-estimated or over-estimated. This is another reason motivating the conservative join attempt using additive increase scheme for join attempts.

E. Dynamic changes of competing traffic

We used a second topology (Fig 12) 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 CLR, the other receivers with the same subscription level also get less throughput even though they do not share the bottleneck with the CLR. It is in this sense that are methods do not provide highly optimized fairness on a per-receiver basis, as receivers may impact one another. However, the extent of the degradation is bounded by a penalty of at most a factor of 2 on all layers except for the base layer. Specifically, once the CLR drops its highest layer due to heavy congestion, a receiver will be selected as CLR among the others which have the same subscription level. Hence, the local traffic increase affects the other receivers on the same subscription level only for the duration of time that its competing SMCC receiver is selected as CLR for the subscription level.

In Figure 12, SMCC 1 receiver is competing with 2 TCP flows for a 12Mbps bottleneck link, while both SMCC 2 and SMCC 3 are competing with different 2 TCP flows for a different 40Mbps bottleneck link. We now set $B_0 = 8$Mbps and all receivers have an RTT of 32ms. SMCC 2 and SMCC 3 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 13 (a) shows the throughput of each of the three SMCC receivers over time, as well as the CLR (either SMCC 2 or SMCC 3) 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 (TCP 7, TCP 8, TCP 9) sharing the 40Mbps bottleneck enter the system . Therefore, SMCC 3's fair share drops from roughly 13Mbps to roughly 7Mbps. SMCC 3 is selected as CLR_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, SMCC 2, suffers performance degradation as it gets the packets sent at the SMCC 3 feedback rate. But SMCC 2's receiving rate is adversely affected by the increase of traffic on the path to SMCC 3 only so long as SMCC 3 is CLR_1. At time 75.7 seconds, SMCC 3 drops its highest layer, L_1 when its calculated rate drops to 7.74Mbps. SMCC 2 is elected as new CLR 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 SMCC 1, who continues to have a lower fair share than SMCC 3, so SMCC 3 receives at a rate of approximately 5Mbps during this time.

Although SMCC 3's fair share is only 7Mbps, for reasons described in Section III-C, 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. SMCC 3 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 SMCC 1 baseline, occur at 87.1 seconds and at 98.3 seconds. The little spikes around 87 seconds and 98 seconds indicate these join attempt failures.

Finally, the three additional TCP flows leave at time 100 seconds. SMCC 3 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 SMCC 2.

Figure 13 (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 cal-

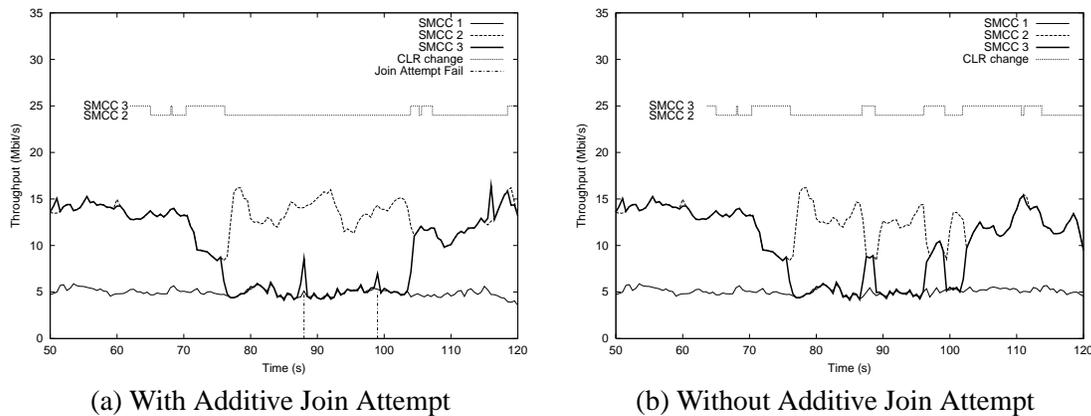(a) With Additive Join Attempt   (b) Without Additive Join Attempt

Fig. 13.   Dynamic changes of competing traffic, $B_0 = 8$ Mbps

culated rate is in the range of the sending rate of the higher layer. SMCC 3 joins the next layer at 86.8 seconds and it becomes CLR for L_1 until 88.8 seconds. During this time, the sending rate of L_1 is dragged down to the rate of SMCC 3, impacting the reception rate of SMCC 2. After dropping back down, SMCC 3 again joins the next layer at 96.1 seconds again and it is again selected as CLR 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.

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.

VI. CONCLUSION

We have presented SMCC, a multirate equation-based multicast congestion control that leverages a proven single rate congestion control method (TFMCC) by orchestrating an ensemble of 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 the 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. A final advantage of our approach is its modular design; TFMCC could easily be replaced by an alternative, or an improved equation-based rate control mechanism.

REFERENCES

[1] J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter, and W. Shaver. FLID-DL: Congestion Control for Layered Multicast. In *Proceedings of NGC*, November 2000. Journal version to appear in IEEE J-SAC, Special Issue on Network Support for Multicast Communications, 2002.

[2] J. Byers and G. Kwon. STAIR: Practical AIMD Multirate Multicast Congestion Control . In *Proceedings of NGC*, 2001. Full version appears as BU-CS-TR-2001-018, Boston University, 2001.

[3] J. Byers, M. Luby, and M. Mitzenmacher. Fine-Grained Layered Multicast. In *Proc. of IEEE INFOCOM*, April 2001.

[4] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. In *Proc. of ACM SIGCOMM*, 1998. To appear in IEEE J-SAC.

[5] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proc. of ACM SIGCOMM*, 2000.

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

[7] 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*, 2002.

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

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

[10] 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*, 8(2):133–145, April 2000.

[11] L. Rizzo. pgmcc: A TCP-friendly single-rate multicast congestion control scheme. In *Proc. of ACM SIGCOMM*, 2000.

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

[13] J. Widmer and M. Handley. Extending equation-based congestion control to multicast applications. In *Proc. of ACM SIGCOMM*, 2001.