STAIR: Practical AIMD Multirate Multicast Congestion Control

John Byers and Gu-In Kwon *

Computer Science Department,
Boston University
Boston, MA 02215
{byers,guin}@cs.bu.edu

Abstract. Existing approaches for multirate multicast congestion control are either friendly to TCP only over large time scales or introduce unfortunate side effects, such as significant control traffic, wasted bandwidth, or the need for modifications to existing routers. We advocate a layered multicast approach in which steady-state receiver reception rates emulate the classical TCP sawtooth derived from additive-increase, multiplicative decrease (AIMD) principles. Our approach introduces the concept of dynamic *stair* layers to simulate various rates of additive increase for receivers with heterogeneous round-trip times (RTTs), facilitated by a minimal amount of IGMP control traffic. We employ a mix of cumulative and *non-cumulative* layering to minimize the amount of excess bandwidth consumed by receivers operating asynchronously behind a shared bottleneck. We integrate these techniques together into a congestion control scheme called STAIR which is amenable to those multicast applications which can make effective use of arbitrary and time-varying subscription levels.

1 Introduction

IP Multicast will ultimately facilitate both delivery of real-time multimedia streams and reliable delivery of rich content to very large audience sizes. One of the most significant remaining impediments to widespread multicast deployment is the issue of congestion control. Internet service providers and backbone service providers need assurances that multicast traffic will not overwhelm their infrastructure. Conversely, content providers in the business of delivering content via multicast do not want artificial handicaps imposed by overly conservative multicast congestion control mechanisms. Resolution of the tensions imposed by this fundamental problem in networking motivates careful optimization of multicast congestion control algorithms and paradigms.

While TCP-friendly multicast congestion control schemes which transmit at a single rate now exist, these techniques cannot scale to large audience sizes. The apparent alternative is *multirate* congestion control, whereby different receivers in the same session can receive content at different transfer rates. Several

* Work supported in part by NSF Grants CAREER ANI-0093296 and ANI-9986397.

schemes for multirate congestion control using layered multicast [1, 7, 8, 11, 13] have been proposed. Also, an excellent survey of related work on multicast congestion control appears in [12]. A layered multicast approach employs multiple multicast groups transmitting at different rates to accomodate a large, heterogeneous population of receivers. In these protocols, recievers adapt their reception rate by subscribing to and unsubscribing from additional groups (or layers), typically leveraging the Internet Group Membership Protocol (IGMP) as the control mechanism. Also, these schemes tend to employ cumulative layering, which mandates that each receiver always subscribe to a set of layers in sequential order. Cumulative layering dovetails well with many applications, such as those which employ layered video codecs [8] for video transmission and methods for reliable multicast transport which are tolerant to frequent subscription changes [11, 4].

In conventional layering schemes, the rates for layers are exponentially distributed: the base layer's transmission rate is B_0, and all other layers i transmit at rate $B_0 * 2^{i-1}$. Therefore, subscription to an additional layer doubles the receiver's reception rate. Reception rate increase granularity of those schemes is unlike TCP's fine-grained additive-increase, multiplicative decrease (AIMD). Because of this coarse granularity, rate increases are necessarily abrupt, which runs the risk of buffer overflow; therefore, receivers must carefully infer the available bandwidth before subscribing to additional layers.

A different approach advocating fine-grained multicast congestion control to simulate AIMD was proposed in [3]. We refer to this approach as FGLM (Fine-Grained Layered Multicast). FGLM relies on *non-cumulative* layering and careful organization of layer rates to enable a receiver to increase the reception rate at the granularity of the base layer bandwidth B_0. Unlike earlier schemes, in this scheme, all receivers act autonomously with no implicit or explicit coordination between them. One substantial drawback of this approach is a constant hum of IGMP traffic at each last hop router (1 join and 2 leaves per client at every additive increase decision point). This volume of control traffic is especially problematic for last hop routers with large fanout to one multicast session, or those serving multiple sessions. Another drawback is that this approach incurs some bandwidth *dilation* at links, wasted bandwidth introduced by the uncoordinated activities of the set of downstream receivers. Finally, the use of non-cumulative layers is only amenable to applications which can make use of an arbitrary (and frequently changing) subset of subscription layers over time. The most natural applications of which we are aware are those in which any packet on any layer is equivalently useful to every receiver; such a situation arises in the digital fountain approach defined in [4], which facilitates reliable multicast by transmitting content encoded with fast forward error correcting codes.

Our work presents a better method for simulating true AIMD multicast congestion control. At a high level, our STAIR (Simulate TCP's Additive Increase/multiplicative decrease with Rate-based) multicast congestion control algorithm enables reception rates at receivers to follow the familiar sawtooth pattern which arises when using TCP's AIMD congestion control. We facilitate this by providing two key contributions. First, we define a stair layer, a layer whose rate dynamically ramps up over time from a base rate of one packet per RTT up to a maximum rate before dropping back to the base rate. The primary benefit of this component is to facilitate additive increase automatically, without the need for IGMP control messages. Second, we provide an efficient hybrid approach to combine the benefits of cumulative and non-cumulative layering *below* the stair layer. This hybrid approach provides the flexibility of non-cumulative layering, while mitigating several of the performance drawbacks associated with pure non-cumulative layering. While our STAIR approach appears complex, 1) the algorithm is straightforward to implement and easy to tune, 2) it delivers data to each receiver at a rate that is in very close correspondence to the behavior of a unicast TCP connection over the same path, and 3) it does so with a quantifiable and reasonable bandwidth cost.

2 Definitions and Building Blocks for our Approach

In order to motivate our new contributions, we begin with techniques from previous work which relate closely to our approach. In [3], four metrics for evaluating a layered multicast congestion control scheme are provided, two of which we recapitulate here.

Definition 1. *The* join complexity *of an operation (such as additive increase) under a given layering scheme is the number of multicast join messages a receiver must issue in order to perform that operation in the worst case.*

Definition 2. *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 $max_{l \in T}(C_l/M_l)$.*

Table 1 compares the performance of various layering schemes which attempt to perform AIMD congestion control. Briefly, one cannot perform additive increase in a standard cumulative protocol[1], and while non-cumulative schemes can do so, they do so only with substantial control traffic and/or bandwidth dilation per operation.

We briefly sketch one non-cumulative layering scheme used [3]. The layering scheme is defined by $B_0 = 1, B_1 = 2$, and $B_i = B_{i-1} + B_{i-2} + 1$ for $i \geq 2$. The first few rates of the layers for this scheme are 1, 2, 4, 7, 12, 20, 33,..., where the base rate can be normalized arbitrarily. Increasing the reception rate by one unit can be achieved by the following procedure: Choose the smallest layer $i \geq 0$ to which the receiver is not currently subscribed; then subscribe to layer i and unsubscribe from layers $i-1$ and $i-2$. A receiver can *approximately* halve its reception rate by unsubscribing from its highest subscription layer. While this

[1] Standard refers to the doubling scheme described in the introduction.

Sequence	Dilation	Complexity of AI	Complexity of MD
Ideal	1	zero	zero
Std. Cum	1	N/A	1 leave
Std. NonCum	2	O($\log R$)	O($\log R$)
FGLM [3]	1.6	2 joins, 1 leave	1 leave

Table 1. Performance of AIMD Congestion Control for Various Approaches

does not exactly halve the rate, the decrease is bounded by a factor which lies in the interval from approximately 0.4 to 0.6.

One salient issue with FGLM is that the base layer bandwidth B_0 is fixed once for all receivers. Setting B_0 to a small value mandates frequent subscription changes (via IGMP control messages) for receivers with small RTTs. Setting it to be large causes the problems of abrupt rate increases and buffer overruns that FGLM is designed to avoid.

3 Components of Our Approach

In this section, we describe our two main technical contributions. The first contribution is a method for minimizing the performance penalty associated with non-cumulative layering by employing a *hybrid* strategy which involves both cumulative and non-cumulative layers. Our approach retains all of the benefits of fine-grained multicast advocated in [3], with the added benefit that the dilation can be reduced from 1.62 down to $1 + \epsilon$ with only a small increase in the number of multicast groups. The second contribution introduces new, dynamic *stair* layers, which facilitate fine-grained additive increase without requiring a substantial number of IGMP control messages. Taken together, these features make the fine-grained layered multicast approach much more practical.

3.1 Combining Cumulative and Non-Cumulative Layering

In a conventional cumulative organization of multicast layers, only cumulative layers are used to achieve rates in the normalized range $[1, R]$.

- *Cumulative Layers (CL)*: The base layer rate is $c_0 = 1$, and for all other layers C_i, $1 \leq i \leq \log_\alpha R$, the rate $c_i = c_0 * \alpha^{i-1}$. When $\alpha = 2$, this corresponds to doubling of rates as each layer is added.[2].

In the fine-grained multicast scheme of [3], only non-cumulative layers are used to achieve a spectrum of rates over the same normalized range.

- *Non-Cumulative Layers (NCL)* : The non-cumulative layering scheme Fib1 presented in [3] has layers N_i whose rates are specified by the Fibonacci-like sequence $n_0 = 1, n_1 = 2$, and $n_i = n_{i-1} + n_{i-2} + 1$ for $i \geq 2$.

[2] Bandwidths can be scaled up multiplicatively by a base layer bandwidth B_0 in these schemes

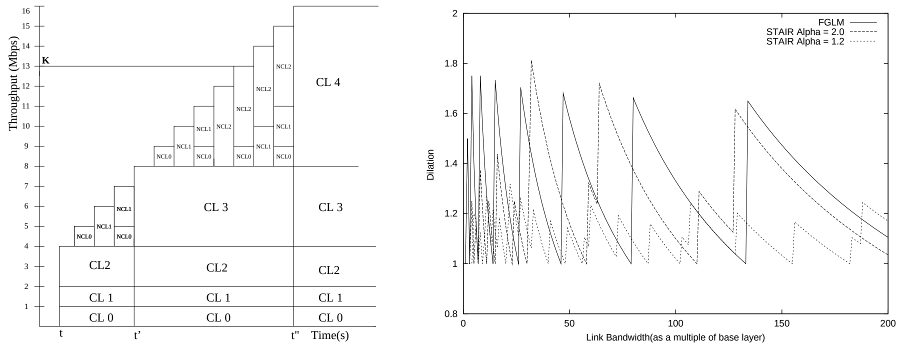

Fig. 1. (left) Hybrid Layer Scheme : $K = \alpha^j + r$, K = $2^3 + 5$ when $\alpha = 2$. CL denotes Cumulative Layer, NCL denotes Non-Cumulative Layer,
(right) Maximal dilation at a link as a function of available link bandwidth

Note that both CLs and NCLs are *static* layers for which the transmission rate to the layer fixed for the duration of the session.

In the hybrid scheme which we propose, we will require that *both* a set of cumulative layers C_i and a set of non-cumulative layers N_i are available for subscription. To attain a given subscription rate K, a receiver will subscribe to set of cumulative layers to attain a rate that is the next lowest power of α, capped by a set of non-cumulative rates to achieve a rate of exactly K, as depicted in Figure 1(left). In particular, we let $j = \lfloor \log_\alpha K \rfloor$ and write $K = \alpha^j + r$, then subscribe to layers $C_0, \ldots, C_j$ as well as the set of non-cumulative layers $\{N_r\}$ that the FGLM scheme would employ to attain a rate of r. As prescribed by FGLM, fine-grained increase (adding c_0) requires one join and two leaves, except for the relatively infrequent case when we move to a rate that is an exact power of α. In this case, we unsubscribe from all non-cumulative layers and subscribe to one additional cumulative layer. Multiplicative decrease now requires one leave from a cumulative layer and one leave from a non-cumulative layer. Comparing against a standard non-cumulative scheme, which used $\log_{1.6} R$ layers, we have now added $\log_\alpha R$ cumulative layers, or a constant factor increase. What we have gained is a dramatic improvement in dilation, expressed as the following lemma.

Lemma 1. *The dilation of the hybrid scheme is* $1 + 1.62\frac{(\alpha-1)}{\alpha}$.

Proof. We proceed by proving an upper bound on the dilation of an arbitrary link ℓ, which gives a corresponding bound on the dilation of the session. For each user U_j downstream of ℓ, consider the rate it obtains over cumulative layers a_j and the rate it obtains over non-cumulative layers b_j separately and denote its total rate by u_j. Let the user with maximal total rate $a_j + b_j$ be denoted by $\hat{U}$ and its rates be denoted by $\hat{a}$ and $\hat{b}$ respectively. Now reconsider user U_j. If $a_j < \hat{a}$, then by the layering scheme employed, $u_j = a_j + b_j < \alpha a_j$. Adding αb_j to both sides gives :

$$u_j + \alpha b_j < \alpha a_j + \alpha b_j = \alpha(u_j) \tag{1}$$

$$\text{Simplifying yields: } b_j < \frac{u_j(\alpha-1)}{\alpha} \le \frac{\hat{u}(\alpha-1)}{\alpha}$$

Otherwise, if $a_j = \hat{a}$, then by maximality $b_j \le \hat{b} < (\alpha-1)\hat{a}$. In either case, b_j is less than $\frac{\hat{u}(\alpha-1)}{\alpha}$, so $\max_j b_j$ is as well. From the dilation lemma proved in [3], a set of users subscribing to non-cumulative layers experience limiting worst-case dilation of 1.62. Thus the total bandwidth consumed by non-cumulative layers across ℓ is at most $1.62 \max_j b_j$. Plugging these derived quantities into the formula in Definition 2 yields:

$$\text{Dilation} \le \frac{\hat{a} + 1.62 \max_j b_j}{\hat{a} + \hat{b}} < \frac{\hat{a} + 1.62 \frac{(\alpha-1)\hat{a}}{\alpha}}{\hat{a} + \hat{b}} \le 1 + 1.62 \frac{(\alpha-1)}{\alpha}. \qquad \square$$

Applying this lemma to a hybrid scheme with a geometric increase rate of $\alpha = 1.2$ on the cumulative layers realizes the benefits of a non-cumulative scheme, reduces the worst-case dilation in the limit from 1.62 to 1.27 (a 22% bandwidth savings) and requires only a modest increase in the number of groups. Figure 1 (right) shows the maximal dilation at a link as the link bandwidth varies as a function of n_0 for FGLM and the hybrid scheme for two different values of α. Recall that in FGLM, there are bandwidth *transition points*, when clients will subscribe to a new maximum layer j and unsubscribe from layers $j-1$ and $j-2$ across the bottleneck. At these transition points (spikes in the plot), worst-case dilation can be large due to the bandwidth consumed by this new layer. While STAIR with $\alpha = 2$ has comparable dilation to FGLM, STAIR with $\alpha = 1.2$ has substantially smaller worst-case dilation.

3.2 Introducing Stair Layers

Our next contribution is stair layers, so named because the rates on these layers change dynamically over time, and in so doing resemble a staircase. This third layer that a sender maintains is used to automatically emulate the additive-increase portion of AIMD congestion control, without the need for IGMP control traffic. Different stair layers are used to accomodate additive increase for receivers with heterogeneous RTT's from the source. These layers also "smooth" discontinuities between subscription levels of the underlying CLs and NCLs, which provide rather coarse granularity (in the subsequent discussion, we assume that these underlying layers have base rates $c_0 = n_0 = 1$Mbps). Finally, we note that the addition of stair layers increases the dilation beyond that proven in Lemma 1, but only by a small additive term, which we quantify in the full version of the paper.

- *Stair Layers (SL)*: Every SL has two parameters: 1) a round-trip time t in ms that it is designed to emulate and 2) a maximum rate R, measured in packets per t ms. The rate transmitted on each SL is a cyclic step function with a minimum bandwidth of 1 packet per t ms, a maximum of R, a step size of

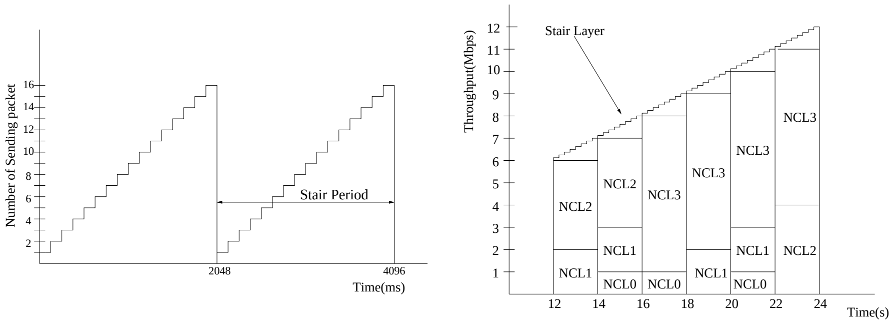

Fig. 2. Use of a stair layer with $t = 128$ms, $R = 1$Mbps, packet size $S = 1$KB. (left) Rate of SL_{128} in isolation, (right) SL_{128} used in conjunction with underlying non-cumulative layers.

one packet, and a stepping rate of one per emulated RTT. Upon reaching the maximum attainable rate, the SL recycles to a rate of one packet per RTT.

Unlike CLs and NCLs, SLs are *dynamic* layers whose rates change over time. Dynamic layers were first used by [11] to probe for available bandwidth and later defined as such and used in [1] to avoid large IGMP leave latencies. Figure 2(left) shows the transmission pattern of SL_{128} (a stair layer for 128ms RTT) with maximum rate $R = 16$ packets per RTT. Also depicted in Figure 2 is a third useful parameter of a stair layer:

Definition 3. *The* stair period *of a given stair layer is the duration of time that it takes the layer to iterate through one full cycle of rates.*

Given a stair layer with an emulated RTT t and a maximum rate R the stair period p satisfies $p = Rt^2$. Typically, we will set the maximum rate R of a stair layer to be the base rate of the standard cumulative scheme c_0 (in Mbps), in which case we substitute for R and perform the appropriate conversions, assuming a fixed packet size S in bytes: $p = \left(\frac{c_0}{8S}\right) t^2$.

In practice, the sender will maintain several SLs to emulate a range of different RTTs. However, the fixed packet size and the maximum rate R of a stair layer give a lower bound on the range of RTT's that can be accomodated. The height of the staircase in steps directly corresponds to the factor in control traffic savings that will be achieved. Denoting this minimum desired height by h, we require that: $t \geq \frac{S*8*h}{R}$. For example, with a packet size of 512 bytes, $R = 1$ Mbps, and a desired value of $h = 8$, then the smallest allowable RTT in an SL is 32ms.

4 The STAIR Congestion Control Algorithm

We now describe how the techniques we have described come together into a unified multirate congestion control algorithm. We employ a hybrid scheme as described in Section 3.1, from which each receiver selects an appropriate subset of layers, used in concert with *one* stair layer, appropriate for its RTT. The two most significant challenges to address are providing the algorithms to performing additive increase and multiplicative decrease, respectively. Two additional challenges we address are 1) incorporating methods for estimation of multicast RTTs and 2) establishing a set of appropriate stair layers.

4.1 Additive Increase, Multiplicative Decrease

In order for a set of stair layers to complement a set of CLs and NCLs, the maximum rate of the stair layer must be calibrated to the base rate of the CLs and NCLs. The effect of appropriately calibrated rates can be seen in Figure 2: at exactly those instants when the stair layer recycles, the subscription rate on the NCL's increases by n_0, to compensate for the identical decrease on the stair layer. Now in order to conduct AIMD congestion control, the receiver measures packet loss over each stair period (during which additive increase takes place automatically). If there is no loss, then the receiver performs an increase of n_0, the base bandwidth of the NCLs as described earlier (1 join and 2 leaves or k leaves when the stair period is an exact power of α). As an aside, we note that it may be much more efficient for a last-hop router to handle such a *batch* of IGMP leave requests, rather than handling them as k separate requests.

Conversely, if there is a packet loss event in a stair period (of one or more losses), then one round of multiplicative decrease is performed. Approximately decreasing the rate by half is straightforward – it is necessary to drop the top cumulative layer as well as the top non-cumulative layer. While existing non-cumulative layering schemes do not easily admit dropping rates by exactly a factor of two, the consequences are mitigated substantially in a hybrid scheme; moreover, our experimental results indicate that the level of TCP-friendliness which can be achieved using our approach remains very high. We also note that there is no particular reason to wait until a stair period terminates before conducting multiplicative decrease – it can be done any time. Since the STAIR receiver unsubscribes and subscribes frequently to increase the rate, IGMP leave latency could be problematic. One solution is to perform joins and leaves in advance of when those operations need to take effect; we are also hopeful that subsequent versions of IGMP will accommodate fast IGMP leaves so that we can use them directly to respond to congestion in a timely fashion.

4.2 Configuration of Stair Layers

As motivated earlier, to accomodate a wide variety of receivers, stair layers must be configured carefully. We choose to space the RTTs across the available stair layers exponentially. Let RTT in the base Stair Layer be $2^i ms$. The base

Stair layer increases its sending rate every $2^i ms$ and all the other stair layers j will increase the sending rate in every 2^{j+i} ms. The TCP throughput rate R, in units of packets per second, can be approximated by the formula in [10]: $R = \frac{1}{RTT\sqrt{q}(\sqrt{2/3}+6\sqrt{3/2}q(1+32q^2))}$, where R is a function of the packet loss rate q, the TCP round trip time RTT, and the round trip time out value RTO,where $RTO = 4RTT$ according to [5].

Since the throughput is inversely proportional to RTT, the receiver with a small RTT is more sensitive to the throughput than the receiver with large RTT, thus we recommend that RTTs provided by stair layers be exponentially spaced. Note that with an exponential spacing of stair layers, a receiver may subscribe to a different SL if its measured RTT changes significantly: it can can subscribe to a faster layer at the end of its current stair period, or drop down to a slower stair layer every other stair period.

4.3 RTT Estimation and STAIR Subscription

Each receiver must measure or estimate its RTT to subscribe to an appropriate stair layer. A variety of methods can be employed to do so; we describe two such possibilities, with the expectation that any scalable method can be employed in parallel with our approach. Golestani et al. provide an effective mechanism to measure RTT in multicast using a hierarchical approach [6]. However, their approach requires clock synchronization among the sender and receivers and depends on some router support which is not widely available. Another simple way to estimate RTT is to use one of various `ping`-like utilities. However, one cost associated with use of ping is that as the number of receivers increase the sender faces a "ping implosion" problem. We leave efficient RTT estimation for future work, noting the need for careful study of the tradeoff between frequency of measurement and accuracy of estimation.

Assuming that the receiver has an estimate of its RTT, its next challenge is to subscribe to the appropriate stair layers. Let RTT_i be the RTT in SL_i and RTT_m be the measured RTT. The receiver can subscribe to appropriate stair layers based on the measured RTT in the following way. If the RTT_m is within a $(1+\epsilon)$ factor from RTT_i for some i, simply subscribe to SL_i. A reasonable choice for ϵ which we argue for in the full paper is $\epsilon = 1/3$. Otherwise, to decrease the error bound in certain cases, a receiver should subscribe to the *two* smaller stair layers SL_i and SL_{i-1} for which $RTT_i < RTT_m$.

5 Experimental Evaluation

We have tested the behaviour of STAIR extensively using the NS simulator [9]. The simulation results show that STAIR exhibits good inter-path fairness when competing with TCP traffic in a wide variety of scenarios. Our initial topology is the "dumbbell", with all non-bottleneck links set to 10ms delay and 100 Mbps bandwidth. In this topology, we vary the cross-traffic multiplexing level by varying the number of TCP flows, vary the bottleneck bandwidth, and scale the queue size. We then consider the impact of richer topologies, including multiple bottleneck links, and TCP cross traffic with both short and long RTTs. Throughout our experiments, we set $c_0 = n_0 = 512$ Kbps and set $\alpha = 2$, i.e. the rate $c_i = 2^{i-1} * 512$ Kbps for $i > 0$. We employ a fixed packet size of 512B throughout. Also, while there is theoretical justification for smaller settings of α, we did not observe worst-case dilation often in our simulations.

In most 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 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.

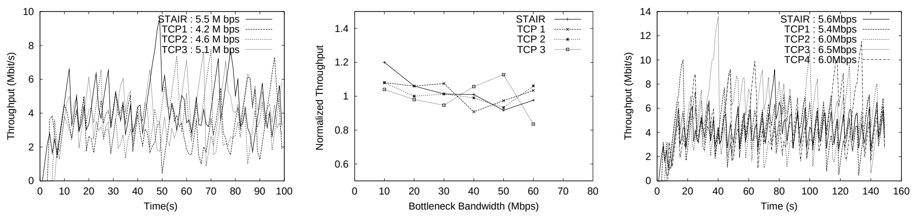

Fig. 3. (a)(b) : TCP flows and One STAIR with RED, (c) DropTail

Figure 3 shows the throughput of a one receiver STAIR flow competing with three TCP Reno flows on RED. Figure 3(a) shows the throughput of STAIR competing with three TCP flows across a dumbbell with a 12ms/20Mbps bottleneck link. In this environment, STAIR fairly shares the link with TCP flows. We next vary the bandwidth of the bottleneck link to assess the impact of changing the range of subscribed NCLs. Figure 3(b) shows the average throughput trends achieved by three long-running TCP flows and one STAIR flow on various bottleneck bandwidths.

Figure 3(c) shows the throughput of STAIR competing with four TCP Reno flows on drop-tail gateways competing for 30Mbps bottleneck bandwidth. Dynamics across a bottleneck drop-tail gateway tended to be more dramatic, but overall fairness remained high. Here, the throughputs of TCP receivers ranged between [5.4Mbps, 6.5Mbps] with a mean per-connection throughput of 6.0 Mbps, while the STAIR receiver had average throughput of 5.6Mbps.

We used a second topology to test heterogeneous fairness (see Figure 4) under different RTTs. We consider a single STAIR session with two STAIR receivers and two parallel TCP flows sharing the same bottleneck link. The RTT of STAIR receiver 1, R_{s1}, is 60ms, while the RTT of STAIR receiver 2, R_{s2}, is 120ms. In our

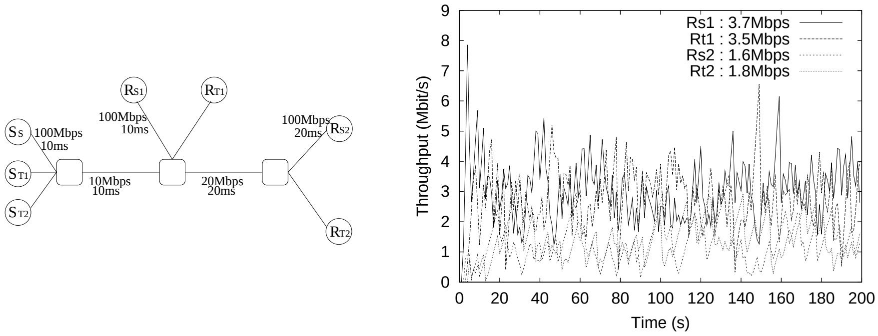

Fig. 4. Throughput of STAIR and TCP flows sharing bottleneck link with different RTT,R_{S1}, R_{S2} : STAIR Receiver, R_{T1}, R_{T2} : TCP receiver

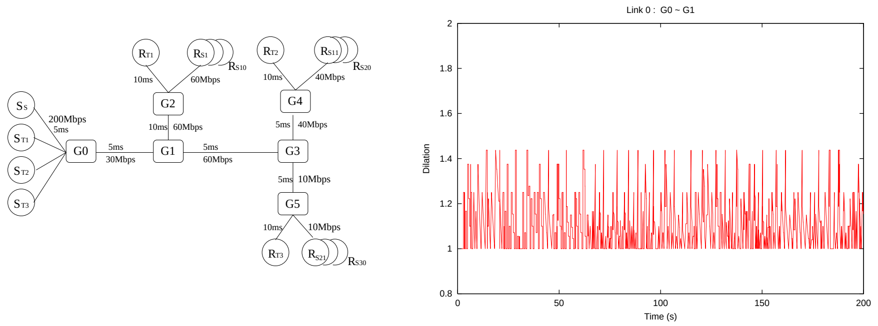

Fig. 5. Dilation on the bottleneck link(G0 - G1)

experimental set up, each receiver periodically samples the RTT using `ping`. R_{s1} subscribes to SL_{64}, which is the closest stair layer based on the measured RTT, while R_{s2} subscribes to SL_{128}. The throughput of each of the flows is plotted in Figure 4. Both of the STAIR flows share fairly with the parallel TCP flows with the same RTT. Since the throughput of TCP is inversely proportional to RTT, the receiver R_{s2} should have approximately half of R_{s1}'s average throughput. In this experiment, the average throughput attained by R_{s1} was 3.7Mbps and the average throughput attained by R_{s2} was 1.6Mbps.

We then increased the number of STAIR receivers using the topology in Figure 5. The starting time of 30 STAIR receivers are uniformly distributed from 1 second to 10 second. The average throughput attained were : $R_{s1} \ldots R_{s10}$: 8.6 Mbps , R_{t1}: 7.3 Mbps, $R_{s11} \ldots R_{s20}$: 8.7 Mbps, R_{t2}: 8.27Mbps, $R_{s21} \ldots R_{s30}$: 4.5Mbps, R_{t3}: 2.3 Mbps. The discrepancy between receiver R_{s30} and R_{t3} points to an aspect of TCP behavior we have not yet captured accurately. When a STAIR receiver subscribes a new maximum j and unsubscribes from layers $j-1$ and $j-2$, it can cause a significant increase in bandwidth consumption. Although the measured dilation on the link(G3 - G5) is less than the dilation in Figure 5, the increases were substantial enough to drive TCP flows into timeout. Since TCP timeout behavior is not yet accurately reflected in STAIR, some unfairness can result.

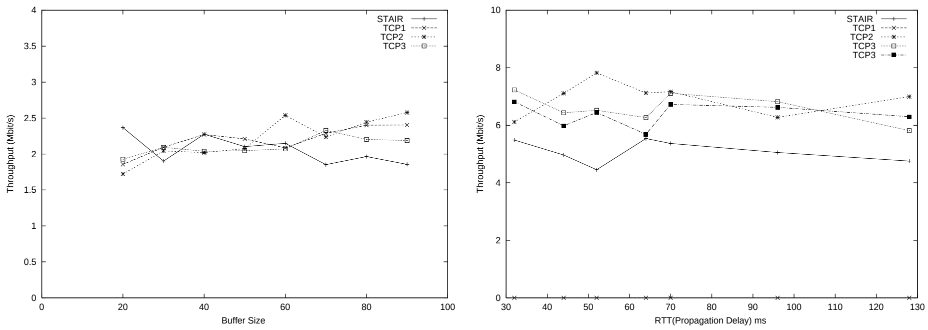

Fig. 6. Throughput on different queue size(left) and different RTT(right), with Drop-Tail

We then considered varying the queue size of the bottleneck link router in our baseline topology, holding all other parameters constant. To make this simulation more interesting, we used drop-tail gateways to magnify the negative performance impact of large queues. Note that for these simulations, when the queue size is large (overprovisioned with respect to the bandwidth-delay product of the link), the RTT is affected by queueing delay. STAIR receivers adapt by changing the stair layer depending on the measured RTT. Figure 6 shows the throughput of STAIR and TCP as we vary the queue size. When the RTT varies over time, the throughput is affected by the error bound of RTT. Even though the average throughput is reduced as the queue size increases, STAIR is not especially sensitive to queue sizes, unlike some other schemes. Finally, we consider varying the link delay on the bottleneck link. Figure 6 (right) shows that as the estimated RTT increases, STAIR becomes less aggressive in accordance with TCP. Additional experiments which we conducted are available in the full version of this paper [2].

6 Conclusions

We have presented STAIR: a hybrid of cumulative, non-cumulative and stair layers to facilitate receiver-driven multicast AIMD congestion control. Our approach has the appealing scalability advantage that it allows receivers to operate asynchronously with no need for coordination; moreover, receivers with widely differing RTTs may simulate different TCP-friendly rates of additive increase. While asynchronous joining and leaving of groups at first appears to run the risk of consuming excessive bandwidth through a shared bottleneck, in fact judicious layering can limit the harmful impact of this issue.

Our approach does have several limitations, which we plan to address in future work. First, while our congestion control scheme tolerates heterogeneous audiences, it is primarily designed for users with high end-to-end bandwidth rates in the hundreds of Kbps range or higher. We expect that slower users would wish to employ a different congestion control strategy than the one we advocate here. Second, congestion control approaches which use non-cumulative layering and dynamic layers cannot be considered general purpose (just as TCP's congestion control mechanism is not general-purpose) since not all applications can take full advantage of highly layer-adaptive congestion control techniques. For now, the only application which integrates cleanly with our congestion control methods is reliable multicast of encoded content. We hope to develop scalable STAIR methods compatible with other applications, such as real-time streaming.

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 2000*, pages 71–81, November 2000.
2. J. Byers and G. Kwon. STAIR: Practical AIMD Multirate Multicast Congestion Control, Technical Report BUCS-TR-2001-018, Boston University, Sept. 2001.
3. J. Byers, M. Luby, and M. Mitzenmacher. Fine-Grained Layered Multicast. In *Proc. of IEEE INFOCOM '01*, 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*, pages 56–67, 1998.
5. S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast application . In *Proc. of ACM SIGCOMM*, 2000.
6. S. Golestani. Fundamental observations on multicast congestion control in the Internet. In *Proc. of IEEE INFOCOM '99*, New York, NY, March 1999.
7. A. Legout and E. Biersack. PLM: Fast convergence for cumulative layered multicast transmission schemes. In *Proc. of ACM SIGMETRICS'2000*, pages 13–22, Santa Clara, CA, 2000.
8. S. McCanne, V. Jacobson, and M. Vetterli. Receiver-Driven Layered Multicast. In *Proc. of ACM SIGCOMM '96*, pages 1–14, 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 throughput: a simple model and its empirical validation. In *Proc. of ACM SIGCOMM*, 1998.
11. L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like Congestion Control for Layered Multicast Data Transfer. In *Proc. of IEEE INFOCOM '98*, April 1998.
12. J. Widmer, R. Denda, and M. Mauve. A Survey on TCP-Friendly Congestion Control. *IEEE Network*, May 2001.
13. K. Yano and S. McCanne. The Breadcrumb Forwarding Service: A Synthesis of PGM and EXPRESS to Improve and Simplify Global IP Multicast. In *Proc. of ACM SIGCOMM Computer Communication Review (CCR), 30 (2), April*, 2000.