LE-SBCC: Loss-Event Oriented Source-based Multicast Congestion Control

Puneet Thapliyal, Sidhartha, Jiang Li, Shivkumar Kalyanaraman

Abstract—

We propose an end-to-end *single-rate* source-based multicast congestion control scheme (LE-SBCC) for reliable or unreliable multicast transport protocols. It addresses key issues such as *drop-to-zero* issues [27], *TCP friendliness* [1] and RTT estimation. The scheme consists of a cascaded set of filters and a rate adaptation policy module (AIMD or TFRC [9]) which transform the multicast tree to appear like a unicast path for the purposes of congestion control. The scheme is not self-clocked but acts upon a stream of loss indications (LIs), which are filtered to get a stream of loss events (LEs) [9] (at most one per RTT per receiver). This LE stream is further filtered to extract the maximum LEs from any one receiver, which results in at most one rate-reduction per RTT. A range of results (simulation and experimental) is presented and compared against the mathematical model of the scheme components.

Keywords—**Multicast, congestion control, multimedia, drop-to-zero, TCP friendliness, loss event**

I. Introduction

THIS paper presents an end-to-end *single-rate* congestion control scheme (LE-SBCC) for reliable or unreliable multicast transport protocols. The scheme is *purely source-based* because it operates on a stream of loss-indications (LIs) from the receivers and does not require support from other elements. This feature distinguishes it from recent single-rate schemes like PGMCC [19], Equation-based TFMCC [14], [27], MTCP [18], Kasera et al [21] and Golestani et al[10]. These latter schemes require some form of special support at receivers (eg: acker capability, RTT/loss estimation capability), network elements (eg: scheme-specific aggregation capability, active services) or protocol packet-formats (eg: new header fields). A *purely* source-based scheme can be implemented by a simple upgrade of the source of a multicast transport protocol. This scheme is modular and allows a variety of rate-based control policies like AIMD or TFRC[9]. It also does not incur congestion-related (S,G) state in receivers, a property which can be leveraged to support multi-source multicast applications. Although at the source we maintain some per-receiver state in the worst case, efficient implementations can reduce the average case memory requirement to below 1MB for 10,000-50,000 receivers.

Rensselaer Polytechnic Institute, Troy, NY 12180. Contact Email: shivkuma@ecse.rpi.edu

This work was funded in part by NSF ANIR grants: ANI9806660, ANI9819112

The LE-SBCC scheme builds upon our earlier work GSC [16] and Bhattacharya et al's LPRF [3], [5]. It differs from these prior schemes in that it tackles the drop-to-zero problem under high degrees of multiplexing, addresses several implementation issues and has a much faster transient response. We believe that it is the first purely source-based scheme scheme to fully address all components of the single-rate multicast congestion control.

LE-SBCC is a *single-rate* scheme like PGMCC [19] and TFMCC [14], which operates at the rate dictated by the worst loss path on the tree and is targeted at small-medium sized trees (up to 10,000 receivers). In contrast, there exist *multi-rate and receiver-based* schemes where the sender organizes data as multiple layers, with each layer being sent to a separate multicast group address. Receivers measure congestion and determine the number of groups they join. These schemes could be further classified as end-to-end (eg: RLM, RLC, and FLID-DL [15], [26], [6], [8], [4]) or network-feedback driven (eg: [13], [20]).

II. LE-SBCC: Scheme Description

The LE-SBCC scheme consists of a *purely source-based* cascaded set of filters and RTT estimation modules feeding into a rate-adaptation module (which could be additive increase/multiplicative decrease (AIMD) or TFRC[9]) (Figure 1). We use AIMD in a majority of our results. These filters (see section II-A) transform the multicast tree to appear like a unicast path for the purposes of congestion control. Unlike TCP, the

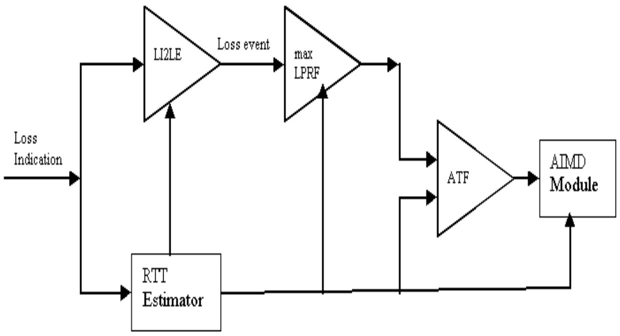


Fig. 1. CASCADED FILTER MODEL OF LE-SBCC

scheme is not self-clocked but acts upon a stream of loss indications (LIs) from receivers. The filters address the following key problems:

• DROP-TO-ZERO: is the problem of reacting to *more loss indications (LIs) than necessary* leading to a beat-down of the multicast flow's rate[27], [5]. This occurs because the multicast flow receives LIs from multiple paths and may not filter LIs sufficiently. In an earlier work [16], we showed that *without* flow multiplexing on paths, the problem of independent multi-path loss can be addressed through a simple adaptive time filter and a TCP-like RTT/RTO estimator. A single flow's congestion control actions directly impacts the loss rate and loss patterns in this case. However, this solution is insufficient if paths display a high degree of flow multiplexing, because then any *single* flow's congestion actions do not appreciably impact the path loss rates.

• TCP-UNFRIENDLINESS: is the problem of reacting to *less LIs than a hypothetical TCP flow would on the worst loss path* [7], [27]. TCP-friendliness is a subject of current debate[9], [19]. In particular, Padhye et al's TCP throughput formula [17] requires the definitions and estimations of "loss rate" and "RTT" of the paths. The unicast scheme, TFRC [9] introduced the concept of loss events (LEs) and uses the LE rate instead of the LI rate as the "loss rate." *An LE is a binary number defined every RTT per receiver which is 1 when one or more LIs are generated in that RTT by the receiver, and is 0 otherwise*. Like TFRC, our paper will also first derive LEs from LIs before responding to congestion.

• RTT ESTIMATION: is the problem of determining the round-trip-time (RTT) of the tree for use by congestion control. PGMCC and TFMCC [19], [14] have receivers estimate path RTT and drive the control based upon paths having the maximum value of $(RTT\sqrt{p_{le}})$, where p_{le} is the *per-packet LE probability*. Our proposed scheme LE-SBCC estimates the maximum RTT of the congested sub-tree. This estimate $\max RTT$, with $\max p_{le}$ is used to drive the various filters and rate-adaptation module. This decoupled estimation and use of $\max RTT$ and $\max p_{le}$ represents a sub-optimal approach in general[19], [14], [27]. However, since our RTT-estimation approach tends to measure the $\max RTT$ from the *currently congested sub-tree*, it reduces this suboptimality in practice.

A. Cascaded Filter Model

The cascade of filters shown in Figure 1 converts a stream of loss indications (LIs) into a stream of rate-reduction indications (RRs) to the rate-based policy module. Assuming AIMD policy, the rate-adaptation module performs multiplicative decrease (MD) upon receipt of a RR and performs additive increase (AI) each time a RR is not received within an increase interval. Observe that this closely models the behavior of AIMD control on a *single path* where RRs are triggered by LEs [9]. The RRs could also be used to trigger other policies like an adapted version of TFRC[9]. The modules work as follows:

• LI2LE FILTER: This filter converts per-receiver loss indications (LIs) into per-receiver loss events (LEs). Recall that an LE is a per-receiver binary number which is 1 when one or more LIs are generated per RTT per receiver, and 0 otherwise. The source stores a *per-receiver timestamp* ($T_{LastPassed}$) which records the time when the last LI was converted and passed as an LE. If a new LI arrives from the receiver *after* a period $SRTT + 2\sigma$ of[1] $T_{LastPassed}$, it is converted into an LE and passed, and the timestamp $T_{LastPassed}$ is updated to the current time. Else the LIs are filtered. This filter is critical because the rate-based AIMD module is not self-clocked (no acks like TCP, PGMCC [19] or Golestani [10]); which leads to large burst losses with drop-tail queues (see section III-A.3).

• MAX-LPRF: The goal of this filter is to pass, on the average, the number of LEs corresponding to what the

[1] *SRTT* = average of RTT samples similar to the variable used in TCP

source would have received from the worst case receiver. This worst-case loss rate estimation problem has been studied earlier (eg: see [3]). Our filter is an extension of Bhattacharya et al's Linear Proportional Response Filter[3] (LPRF). The *LPRF* is a filter which passes loss indications (LIs) with a probability $\frac{L_i}{\Sigma_i L_i}$ where L_i is the number of LIs from receiver i.
The LPRF filter has some practical issues which preclude its direct use. Firstly, it was designed under Markovian model assumptions [3]. Second, *LPRF* did not quickly adapt to sudden increases in worst-case loss rates. For example, consider a tree with a 1000 receivers, each with a loss rate of 1%. If one of the receivers suddenly sees a loss rate of 10%, then *LPRF* convergence is very slow. Third, *LPRF* works with LIs instead of LEs. Our subsequent arguments show that using LEs is much superior to using LIs in non-self-clocked systems. Fourth, it does not specify an RTT estimation procedure.
Our *Max-LPRF* works as follows: Assuming X_i is the count of LEs from receiver i, this filter takes as input all the LEs from receivers ($\Sigma_i X_i$) and on an average passes the maximum number of LEs from any one receiver (i.e. $\max_i X_i$). In particular, it *passes each LE with a probability* $\frac{\max_i X_i}{\Sigma_i X_i}$. The LE counts per receiver (X_i) are decayed periodically by 10% every 100 $SRTTs$.

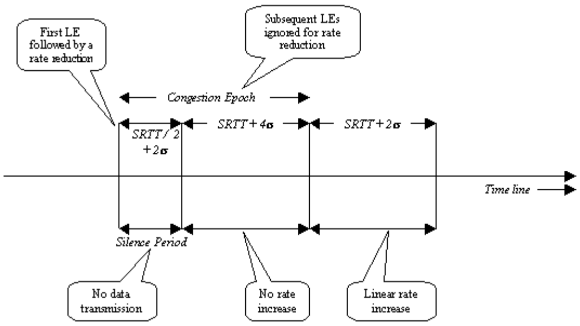


Fig. 2. ADAPTIVE TIME FILTER (ATF)

• ADAPTIVE TIME FILTER (ATF): This filter shown in Figure 2 is a modification of the time-filter introduced in our earlier work [16]. This filter simply drops excess LEs passed by Max-LPRF in any RTT to enforce *at most one rate reduction (RR) per* $SRTT + 4\sigma$. In addition, the filter also imposes a *"silence period"* of $\frac{1}{2}(SRTT + 4\sigma)$ when no packets are sent[16] to reduce the probability of losing control traffic/retransmissions during this phase.

• RTT ESTIMATION: We use the LI-based RTT estimation module developed in our earlier work [16], which works similar to the TCP timeout procedure. It uses a smoothed RTT ($SRTT$) and a mean deviation σ. The RTT sample set is pruned to largely exclude small samples ($< SRTT/2$) to bias the $SRTT$ estimate higher. The procedure is LI-driven, and hence tends to estimate the worst RTT from the *currently congested sub-tree*.

In summary, for ideal operation, the scheme expects: (a) at least one LI per receiver seeing loss per RTT if packets are lost in that RTT (i.e. at least one LE per RTT) and (b) timely generation/forwarding of feedback by receivers/network elements to allow reliable RTT sampling [2]. If (a) and (b) are not satisfied, the scheme performance will degrade. Eg. LI aggregation may affect (a) (section III-C), and delays introduced by receivers may affect (b). The pseudo code for the scheme is presented in Appendix A.

III. Performance Evaluation

We have evaluated the performance of LE-SBCC to test for Drop-to-Zero avoidance, TCP-friendliness and LI aggregation effects. We evaluate our scheme using simulations (for up to 10000 receivers), simple stochastic modeling (to demonstrate drop-to-zero avoidance) and linux-based implementation (to understand implementation complexity).

A. Evaluation: Drop-to-Zero Avoidance

Drop-to-Zero Avoidance is the problem of reacting to more loss indications (LIs) than necessary leading to a beat-down of the multicast flow's rate. The test cases for drop-to-zero we use are:
a) a multicast tree where branches each see non-trivial degree of multiplexing with other multicast or TCP flows.
b) a multicast tree with large number of receivers, each experiencing different loss rates.

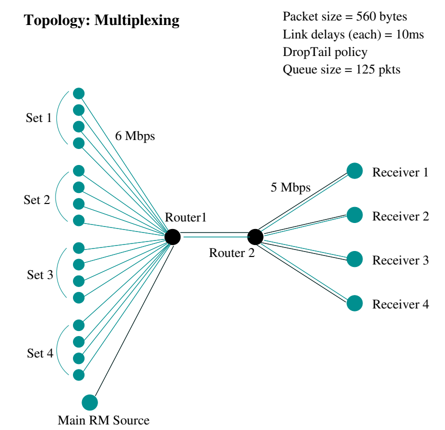


Fig. 3. TOPOLOGY TO TEST DROP-TO-ZERO AVOIDANCE AND TCP-FRIENDLINESS

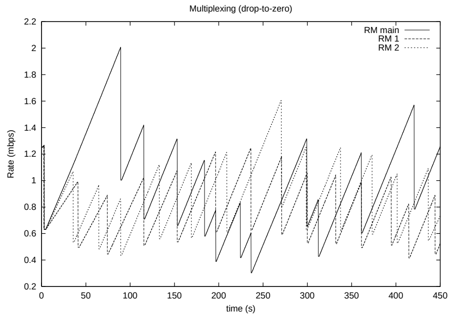


Fig. 4. DROP-TO-ZERO AVOIDANCE RESULTS

A.1 Effect of Multiplexing on Independent Branches

Figure 3 shows a single multicast (main RM) flow to four receivers. On each branch (5 Mbps bottleneck) it competes with four other unicast flows implementing the same algorithm. Observe that Max-LPRF and ATF do not affect LE feedback for these unicast flows. The fair rate for all flows is therefore 1 Mbps each. The buffer size (125 pkts) is roughly twice the bandwidth times fixed delays. The packet size is 560 bytes. This topology tests the performance of the multicast congestion control scheme under independent loss rates on paths, and non-trivial multiplexing on each path. Figure 4 (rate graph) shows that the multicast flow rate (solid line) competes fairly with a selected subset of unicast background flows (dotted lines) and the average rate is close to 1 Mbps, the fair share. This illustrates drop-to-zero avoidance.

The analysis of the number of packets transmitted, the number of LIs and LEs and rate reductions (RRs) also illustrates the impact of the cascaded set of filters. In particular, the multicast RM flow transmitted a total of 167762 pkts in 450s and received 949 LIs (i.e. a low average LI rate). After LI2LE filtering, there were only 32 LEs, a dramatic reduction! The Max-LPRF and ATF further filter LEs to result in a total of 12 rate-reductions which is the same number of RRs effected by unicast flows.

[2]Within these constraints the scheme may be successfully applied to unreliable multicast transport protocols

A.2 Effects of Removing a Filter From the Cascade

A natural question is the relative importance of each filter in the LE-SBCC's cascade of filters. To further demonstrate this, we repeat the simulations for the configuration shown in Figure 3 for three different cases:

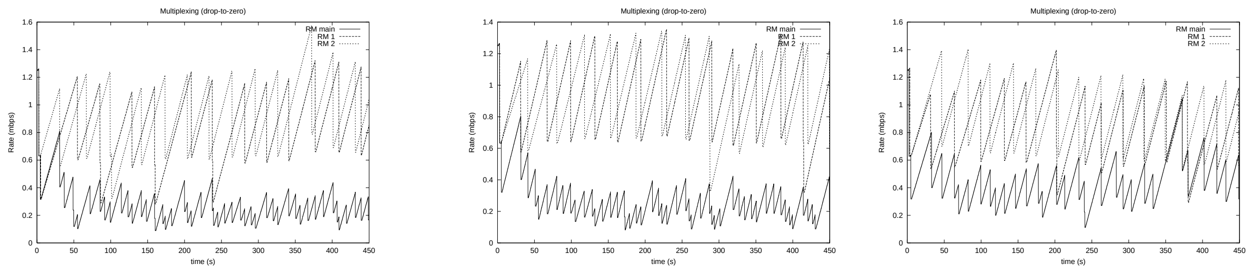
(a) Removing both *LI2LE* and *maxLPRF* filters
(b) Removing only the *maxLPRF* filter, and
(c) Removing only the *LI2LE* filter.

All other parameters are kept the same as before. The results in Figure 5 show "drop-to-zero" problem to some degree in all cases (solid curves always below dotted curves). The results are also affected by the fact that in some cases the max-LPRF operates on LIs. In these cases, the max-LPRF processes a larger absolute population of samples, and deals with a different relative distribution of counts from receivers (which affects the critical ratio: $\frac{\max_i X_i}{\Sigma_i X_i}$). Thus each filter in the cascade has a key role in filtering of LIs and avoiding the drop-to-zero problem.

A.3 Theoretical Analysis of LE vs LI Probability

This section presents a simple Markov chain model to show that using LEs in general leads to a reduced probability of drop-to-zero, especially under bursty loss conditions.

Assume that the system moves in cycles of RTT, which is constant. In each RTT, the source transmits with a constant rate, and may experience one or more packet losses. At the end of each cycle, it may receive zero or one LEs which triggers either an additive increase (AI) or a multiplicative decrease (MD) respectively. This behavior is modeled as a Markov chain with N discrete states (from 1 to N), shown in Figure 1. State x corresponds to the number of packets sent during that RTT. Therefore, $x = Rate(x) \times RTT$. The process starts in state 1, and may go up to the maximum allowable rate, which corresponds to state N. A loss event receipt triggers a transition to state $\lfloor x/2 \rfloor$. No loss event receipt triggers a transition to state $x + 1$. Our goal is



(a) Removing both LI2LE and maxLPRF (b) Removing maxLPRF (c) Removing LI2LE

Fig. 5. EFFECTS OF REMOVING A SUBSET OF FILTERS FROM THE LE-SBCC CASCADE

to estimate the ratio of per-packet LI probability (p_{li}) to per-packet LE probability (p_{le}), i.e., $E(p_{li})/E(p_{le})$. We analyze this chain for two cases: the non-bursty and bursty packet loss case.

1. NON-BURSTY LOSS CASE

Assume that the packet loss probability is uniform. Define p: the probability that a packet is lost, and a: 1 - p. In this case $p_{li} = p$. In state x, the probability of having at least one loss is $1 - a^x$. The LE probability (p_{le}) is thus $\sum s(x)(1 - a^x)$ where $s(x)$ is the probability of being in state x. Let $m = \lfloor N/2 \rfloor$. This leads to five steady state equations (see Figure 1).

$$
\begin{aligned}
s(1) \times a &= s(2) \times (1 - a^2) + s(3) \times (1 - a^3); \;\; x = 1 \\
s(k) &= s(k-1) \times a^{k-1} + s(2k) \times (1 - a^{2k}) \\
&\quad + s(2k+1)(1 - a^{2k+1}); \;\; 2 \leq x \leq m - 1 \\
s(m) &= s(m-1) \times (a^{m-1}) + s(N) \times (1 - a^N); \;\; x = m \\
s(k) &= s(k-1) \times a^{k-1}; \;\; (m+1) \leq x \leq (N-1) \\
s(N) &\times (1 - a^N) = s(N-1) \times a^{N-1}; \;\; x = N
\end{aligned}
$$

We numerically derived the solution for this Markov chain. The solutions yield the steady state probabilities $s(x)$ and hence the probability $p_{le} = \sum s(x)(1 - a^x)$. We then plot $\frac{p_{li}}{p_{le}}$ Figure 8(a), which has a maxima of 1.25. In other words, the number of LEs is at most 25% smaller than the number of LIs for $p = 0.3$. However, as we shall see the bursty loss case (seen in drop-tail cases), leads to larger values of $\frac{p_{li}}{p_{le}}$ thus justifying our filters.

2. BURSTY LOSS CASE

Our scheme uses a rate-based AIMD policy, which is *not self-clocked like TCP since we depend upon a stream of LIs, not acks*. A simple analysis[22] shows that such schemes lead to a quadratic rate of bottleneck queue growth. Therefore, given larger buffers, when the buffer is full, the sending rate differs significantly from the optimal transmission rate. So, assuming drop-tail policy the number of packets dropped corresponds to $(R - C)T$ where R is the aggregate transmission rate and C is the bottleneck capacity, and T is the round trip time (RTT) including the queueing delay. This implies a large burst drop of packets. Under such conditions, the choice of LEs over LIs is even more justified.

Consider the prior Markov chain model as used earlier, albeit with some changes. To approximate the bursty loss behavior, we assume that if a packet is lost during any instant within the RTT, all remaining packets transmitted during that RTT are also lost. This assumption is also made by Padhye et al [17] for TCP throughput modeling. Under this assumption, p assumes different semantics. p is defined as the probability that a packet is lost, given that either it is the first packet in the RTT or the preceding packet is not lost. Assume like before that $a = 1 - p$.

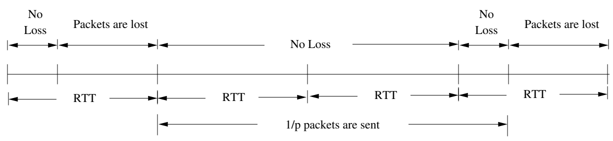
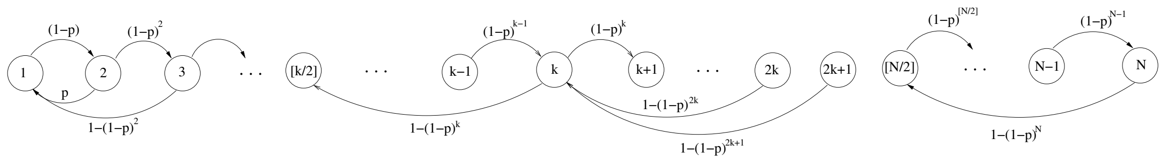


Fig. 7. p's SEMANTICS IN BURSTY LOSS CASE

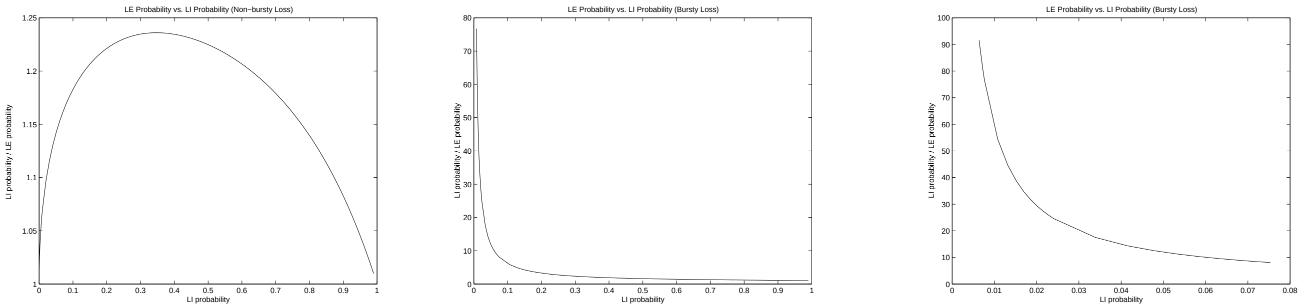
In this case, $p \neq p_{li}$ (see Figure 7) because p_{li} is affected by every packet loss unlike p. However, the p_{li} can be derived from p as follows. First, recall that the state x specifies the number of packets transmitted during an RTT. Assuming that the first i packets are not lost, $x - i$ is the number of lost packets in that RTT. Therefore, P(The first i packets are not lost and the sub-



For non-bursty loss, $p_{li} = p$.

For bursty loss, p is the probability that a packet is lost and it is the first loss after the most recent RTT with a loss.

Fig. 6. STATE TRANSITION OF PACKETS SENT PER RTT



(a) Non-bursty loss (b) Bursty loss (All loss probabilities) (c) Bursty loss (Low loss probabilities)

Fig. 8. RATIO OF LI AND LE PROBABILITIES

sequent packets in the round are lost) $= (1-p)^i p$. Therefore, at state x, the expected number of lost packets is $\sum_{i=0}^{x}(x-i)(1-p)^i p$. The LI probability p_{li} is the expected number of lost packets divided by the expected total number of packets transmitted:

$$p_{li} = \frac{\sum_{x=0}^{\infty} s(x) \sum_{i=0}^{x} (x-i)(1-p)^i p}{\sum_{x=0}^{\infty} x \cdot s(x)} \tag{1}$$

At state x, the expected number of loss events (LE) is the probability that there is at least one loss, that is $1 - (1-p)^x$. Therefore,

$$p_{le} = \sum_{x=0}^{\infty} s(x) \cdot (1 - (1-p)^x) \tag{2}$$

From equations 1 and 2, we see that ratio of LI and LE probabilities ($\frac{p_{li}}{p_{le}}$) is substantially different compared to the uniform loss case. The plot of $\frac{p_{li}}{p_{le}}$ for the bursty model is shown in Figure 8 (b) and (c). The ratio $\frac{p_{li}}{p_{le}}$ is large when p_{li} is small (which is the common case of network operations). This is because the small overall LI probability leads to concentrated burst loss in few RTTs leading to a small number of LEs in comparison to LIs. We observe high p_{li}/p_{le} ratios (9-90). Comparing these numbers to the numbers of LIs and LEs observed in simulation (see section III-A.1), we observe that the model gives the same order of magnitude of results as the simulation results. In that case $p_{li}/p_{le} = 949/32 = 29.7$. Therefore, we can conclude that the LI2LE filtering is critical in our cascade filter design.

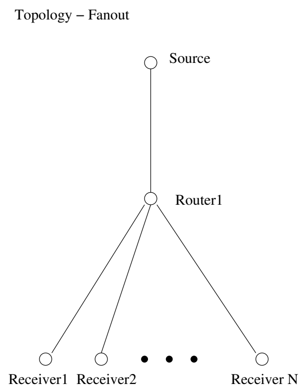
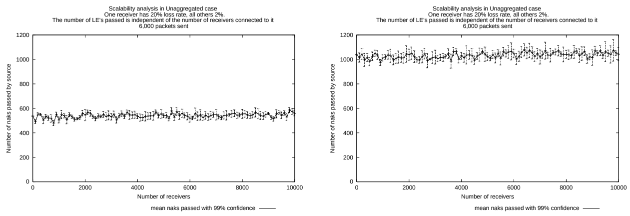


Fig. 9. TOPOLOGY : LARGE RECEIVER SETS

A.4 Drop-to-Zero Avoidance with Large Receiver Sets

Consider the topology shown in Figure 9, where one of the receivers in the session has the worst loss rate, and the rest have lower loss probabilities. Irrespective of the total number of receivers, we expect that the number of LEs passed to be constant, and equal to the number of LEs generated by the worst loss receiver. For simplicity, we assume constant RTTs and synchronous feedback. We vary the number of receivers from 1 to 10000, with the worst-loss receiver seeing 10% loss, and the others seeing 2% loss.

The number of LEs passed for rate reduction by the source is plotted against the number of receivers(Figure 10(a)), with 99% confidence intervals. As expected, the number of LEs passed by the cascade of filters is in the range 500-520. This is consistent with the 10% loss rate and total of 6000 packets transmitted (i.e. 600 expected LIs); and the non-bursty model where the ratio: $\frac{p_{li}}{p_{le}}$ for 10% loss is 1.2 (i.e. expected 500 LEs). The experiment was repeated with a worst-loss of 20% (see Figure 10(b)), again with results as expected (around 1000 LEs passed). This illustrates the correctness of our filter cascade for larger trees with heterogeneous, independent loss paths.



(a) 10% Max loss receiver (b) 20% Max loss receiver

Fig. 10. SCALABILITY ANALYSIS

B. *Evaluation: TCP friendliness*

TCP-friendliness is a subject of current debate [9], [19], [14]. A good measure of TCP-friendliness would be the average throughputs of the competing TCP and RM flows. To illustrate TCP-friendliness, we use two topologies, a) where a common bottleneck is shared by many RM flows and a single TCP flow and, b) many TCP flows and a single RM flow. In Figure 3, consider two cases: case (a) when *one* of the flows of "Set 4" is replaced by a TCP Reno flow while the rest are LE-SBCC flows; and case (b) when *all* the flows in "Set 1" through "Set 4") are replaced by TCP flows.

SIMULATION RESULTS: The simulation results for these two cases is shown in Figures 11(a) and 11(b) [3]. The dips by the TCP flow close to zero represent time-outs. Again observe that the main RM flow (solid line in both graphs) shares the bottleneck(s) fairly with the competing TCP flow(s). This becomes clearer by observing that the rate graphs for the RM and TCP flows oscillate about roughly the same mean and with the same variance. The horizontal dotted line is the expected TCP throughput plotted post-simulation using the simplified TCP equation ($1.22MSS/RTT\sqrt{q}$). Recall that q is the LE probability and its value is measured here by counting the number of LIs passed by the LI2LE filter. Also recall that MSS is 560 bytes and the RTT is the smoothed RTT measured by the *RTT estimator module* of our scheme.

[3]To avoid cluttering, the TCP throughput is sampled every 2s

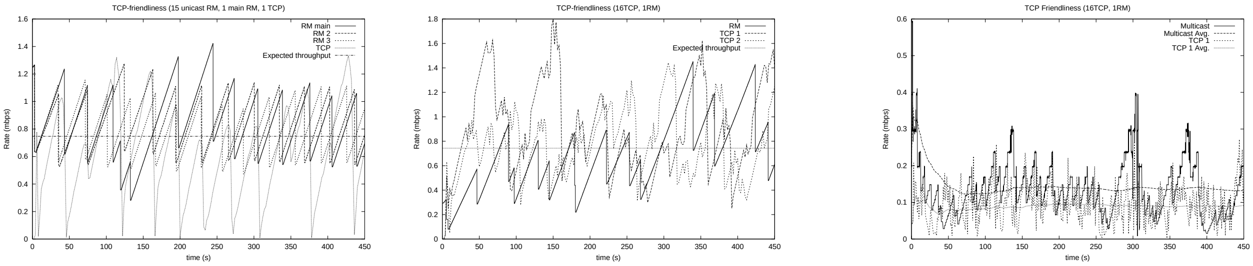
EXPERIMENTATION RESULTS: We implemented the scheme in Redhat 7.0 over a simplified version of PGM [24]. The scheme runs as a user process, and hence is subject to OS scheduling delays. Moreover, since timers are coarse-grained in the user-plane, we introduce approximations in the source rate-controller which lead to somewhat bursty transmissions. The transmissions were also bursty because a limited number of machines were used, and multiple sources executed on each machine. The details of implementation issues are discussed in a technical report [25]. Further, due to experimental setup limitations (especially due to the Linux diffserv *tc* package used), we tested the scheme on topology shown in Figure 3, but with different parameters:

- Round trip link latency: 500 *ms*
- Bottleneck bandwidth: 1Mbps
- Source initial transmission rate: 200Kbps
- Source initial RTT estimate: 200 *ms*
- Receiver maximum random backoff before sending NAK's: 100 *ms*

The results are illustrated in Figure 11(c). Observe, that the multicast (RM) flows are roughly fair to the competing TCP SACK flows. The burstiness of transmission and delayed response due to several factors mentioned above lead to a somewhat higher throughput for multicast (RM) sources.

C. *Evaluation: LI Aggregation Effects on Performance*

Aggregation of loss indications (LIs) has an effect on the number of LEs and timing information reaching the source, and therefore affects the performance of LE-SBCC. To evaluate these effects, consider the simple multi-level topology in Figure 12. As the number of



(a) 1 TCP Reno vs 16 RM (b) 1 RM vs 16 TCP Reno (c) Experimental (1RM vs 16 TCP SACK)

Fig. 11. TCP FRIENDLINESS (SIMULATION AND EXPERIMENT)

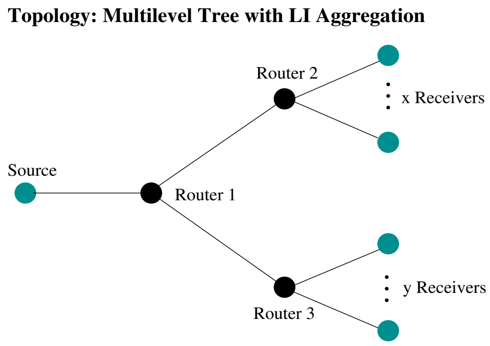


Fig. 12. TOPOLOGY TO ILLUSTRATE LI AGGREGATION EFFECTS

receivers and their loss rates are varied, consider three cases which we simulated:

1. No LI Aggregation performed at the routers.

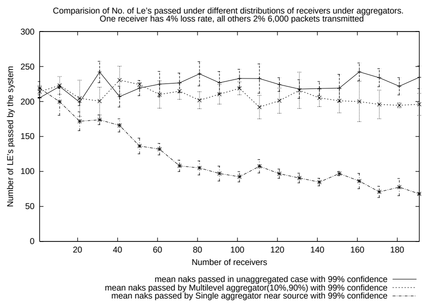
2. Partial LI Aggregation: LI aggregation performed at router 2 and router 3 only (and not in router 1).

3. Complete LI Aggregation: LI aggregation performed at all routers.

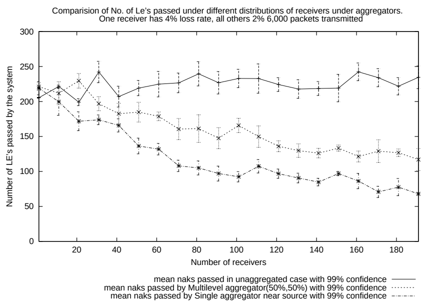
Consider cases 2 and 3. Assume one receiver X has a fairly high loss rate, and all others have very low loss rates. Ideally, the number of LEs reaching through the source (after filtering) should be equal to the number of LEs generated by worst-loss receiver X. If a larger fraction of the LIs sent by X are suppressed by those sent from other low-loss receivers, then fewer and fewer LIs generated by X might reach the source. We call this the partial aggregation case (case 2). This could occur because X has a longer RTT, and as the number of receivers increase, we expect more overlaps between sequence numbers lost by X and other receivers. In the worst case, no LIs generated by X would reach the source. Then even after LI2LE filtering, the maximum number of LEs generated by *any single receiver* would now become much lower, and thus affects the critical ratio $\frac{\max_i X_i}{\Sigma_i X_i}$ used by Max-LPRF. Hence MaxL-PRF would pass through lesser number of LEs. The source hence would make fewer rate reductions, becoming *more aggressive compared to TCP flows on the worst loss path*. Observe that fewer rate reductions also means that the multicast flow even under cases of full (worst-case) aggregation *does not experience the drop-to-zero problem*.

The graphs shown in Figures 13 (a), (b), (c) illustrate the effects of no aggregation, partial and worst-case aggregation. The graphs plot the number of LEs passed by the filter cascade against the total number of receivers. The topology is same as in Figure 12. One receiver has loss probability of 4% (and has a longer RTT) while all others have 2%. This would bias the aggregation process against the worst-loss receiver's LIs. We vary the distribution of receivers under the two aggregators in the multicast tree. The notation (x,y) indicates that x% of all the receivers (including the receiver with 4% loss probability) are attached to Router2 and remaining y% are attached to Router3. The experiments were repeated for different random seeds and confidence intervals plotted. In each experiment, 6000 packets were sent which leads to an expectation of 240 LEs (given a worst-loss receiver having 4% loss rate) after the cascade of filters is applied. If fewer than 240 LEs are passed, the multicast rate would be higher than competing TCP on the worst path.

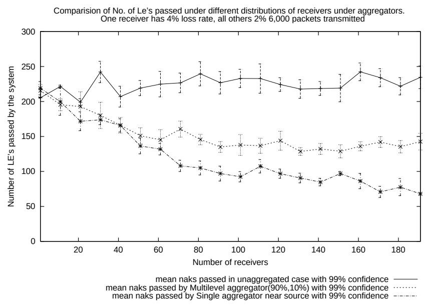
In all graphs of Figures 13, the no-aggregation case passes just under 240 LEs (99% confidence intervals), which implies that the scheme works very well without aggregation. This is also similar to the results seen in Figure 10 where we used different loss probabilities and examined a larger number of receivers (up to 10,000 receivers). Now, the performance of the intermediate curve (partial aggregation) in Figures 13 varies



(a) (10,90)



(b) (50,50)



(c) (90,10)

Fig. 13. PARTIAL AGGREGATION EFFECTS

depending upon the pattern of aggregation. Observe that if more LIs from the max loss receiver get through without begin aggregated, the intermediate (partial aggregation) curve is almost comparable to the scenario with no aggregation (eg: Figure 13 (a)). However, in the case of Figures 13 (b) & (c) the max loss receiver is present in the subtree with a larger number of receivers. Combined with the effect of longer RTT, these configurations bias the aggregation process against the worst-loss receiver because of the increased probability of overlap between the ranges of lost packets seen by the worst-loss receiver and other receivers connected to Router2. Hence Figures 13 (b) & (c) shows the partial aggregation curve closer to the worst-case curve. The worst-case curve in all the three graphs depicts the effect of complete aggregation done at Router1, which suppresses all the LIs from the worst-loss receiver.

In summary, aggregation poses a performance problem for LE-SBCC in the cases where the worst-loss receiver's feedback is significantly suppressed. It should be noted that aggregation issues pose a problem for PGMCC too (see [23]). Any randomization or technique to break the bias against such worst-loss receivers in the aggregation process (eg: see [23]) would solve this performance degradation issue both for LE-SBCC and PGMCC.

IV. TFRC MODULE

TFRC [9] is a congestion control policy which provides smooth rate changes. Our modular approach allows the use of TFRC instead of AIMD, with very few changes.

A. *TFRC Module Design*

For AIMD, we used loss events (LEs) to trigger rate reductions. However, for TFRC we need to convert them into a LE rate to update the data transfer rate. We modify the receiver algorithm of TFRC so that the LE rate can be calculated at the source. Since the output of the cascaded filters are loss events (LEs), the source can count the loss intervals (the number of packets between consecutive LEs) and calculate the LE rate.

Assume:

- s_0: the interval since the most recent LE.
- s_i: the i-th most recent loss interval. $(i \geq 1)$
- N: the total number of loss intervals from the beginning of the data transfer.
- n: an integer greater than 0.
- α: $0 \leq \alpha \leq 1$.

The average loss interval $\hat{s}_i$ [4] is calculated as:

$$\hat{s}_i = \frac{\sum\limits_{k=1}^{n} s_{i+k-1} w_k + \sum\limits_{k=i+n}^{N} s_k \alpha^{k-i-n+1}}{\sum\limits_{k=1}^{n} w_k + \sum\limits_{k=i+n}^{N} \alpha^{k-i-n+1}}$$

In [9], $n = 8$, $\alpha = 0$, w_1 to w_8 have values of 1, 1, 1, 1, 0.8, 0.6, 0.4, 0.2 respectively. In our scheme, the randomness of the MaxLPRF filter can cause larger variance of the average loss interval and therefore more oscillation of the LE rate. To counter this effect, we chose different values for these parameters. In our experiments, $n = 8$, w_1 to w_8 are all 1, and $\alpha = 0.95$. (When there are not enough loss intervals, (1) $N < n$, set w_{N+1}, ..., w_n and α to 0, (2) $N = n$, set α to 0.) Whenever a LE is detected, $\hat{s}_1$ is calculated, the reported LE rate p is $1/\hat{s}_1$. Then the data transfer rate

[4] Only $\hat{s}_0$ and $\hat{s}_1$ are used for our TFRC adaption.

T (bytes/sec) is calculated with the following formula [9] and enforced.

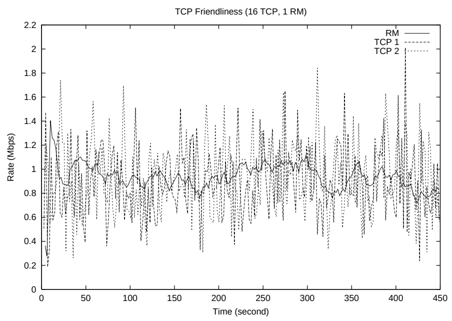
$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)}$$

where s is the packet size, R is round-trip time, p is LE rate, $t_{RTO} = SRTT + 4 * \sigma$.

During absence of congestion, after each RTT, $\hat{s}_0$ is re-calculated. The final average loss interval $\hat{s}$ is $\max(\hat{s}_0, \hat{s}_1)$. If it is different from the previous value of $\hat{s}$, the rate T will be updated with the reported LE rate $p = 1/\hat{s}$. If congestion is absent for a long time, $\hat{s}_0$ will become larger and larger, so is $\hat{s}$, and the reported LE rate will drop. Consequently, the rate T will increase, given that the RTT does not vary much. Note that our adaptation of TFRC for multicast differs from the recent work in TFMCC [14] where each receiver measures the RTT and loss event (LE) rate.

B. Simulation Results

We tested the scheme with the TFRC module on the topology shown in Figure 3. The instantaneous rates of the multicast flow and two randomly selected TCP flows are shown in Figure 14 (a). Compared to the TCP flows, the TFRC rate-changes are much smoother, without degrading fairness or drop-to-zero avoidance. This shows that a variety of rate-adaptation schemes can be installed into our modular framework (eg: AIMD, binomial [2] schemes, TFRC [9] etc).



(a) Instant rates

Fig. 14. TFRC MODULE SIMULATION RESULTS (1 TFRC-BASED RM VS 16 TCP RENO)

V. SUMMARY AND CONCLUSIONS

We have presented an *LE-oriented, purely source-based congestion control scheme (LE-SBCC)* for reliable and unreliable multicast transport. Since the scheme is completely contained at the source, it is simple to implement and deploy. The model is a simple cascade of filters followed by a rate-adaptation module (AIMD or TFRC). The filters transform the multicast tree to appear like a unicast path for the purposes of congestion control. In the section of theoretical analysis of loss events (LEs) probability versus loss indications (LIs) probability, we justified our choice of LEs over LIs. By studying the effects of removing a filter from the cascade, we showed the importance of all filters. Several simulations and Linux-based implementation experiments confirm the effectiveness of our scheme.

Without LI aggregation, the scheme correctly filters the LEs (converted from LIs); avoids the drop-to-zero problem while being TCP-friendly. With aggregation, the scheme is likely to be more aggressive towards TCP, but never incurs the drop-to-zero problem. Like all single-rate schemes, its scalability is limited by the heterogeneity of the tree which may depress the source rate. In general, the AIMD module in our scheme may also be replaced by a source-based TFRC scheme [9] or rate-based binomial scheme [2].

VI. ACKNOWLEDGEMENTS

We acknowledge the help of Jye-Young Song to implement this scheme. We also thank Supratik Bhattacharyya, Sonia Fahmy, Luigi Rizzo, Mark Handley, Lorenzo Vicisano, Jon Crowcroft and Brian Whetten for extensive comments.

REFERENCES

[1] A. Mankin, A. Romanow, S. Bradner, and V. Paxson., *RFC 2357: IETF criteria for evaluating reliable multicast transport and application protocols*, June 1998. Obsoletes RFC 1650. Status: INFORMATIONAL.

[2] Bansal D. and Balakrishnan H., *Binomial Congestion Control Algorithms*, INFOCOM 2001, Apr 2001.

[3] Bhattacharyya S. et al, *A Novel Loss Indication Filtering Approach for Multicast Congestion Control*, J. of Comp. Commns, Feb 2001.

[4] Bhattacharyya S., Towsley D. and Kurose J., *Efficient Multicast Flow Control using Multiple Multicast Groups*, U.Mass, Amherst, CMPCSI Technical Report TR 97-15, 1997.

[5] Bhattacharya S., Towsley D. and Kurose J., *The Loss Path Multiplicity Problem in Multicast Congestion Control*, INFOCOM '99, March '99.

[6] Bolot J.C., Turletti T., Wakeman I., *Scalable Feedback Control for Multicast Video Distribution in the Internet*, SIGCOMM '94, Aug '94.

[7] Bradner S. et al, *IETF criteria for evaluating reliable multicast transport and application protocols*, RFC 2357, June '98.

[8] Byers J.W., et al, *FLID-DL Congestion Control for Layered Multicast*, NGC 2000, Nov 2000.

[9] Floyd S. et al, *Equation-Based Congestion Control for Unicast Applications*, SIGCOMM 2000, Aug 2000.

[10] Golestani J., *Fundamental Observations on Multicast Congestion Control in the Internet*, INFOCOM 1999, March '99.

[11] Gopalkrishnan R. et al, *Stability and Fairness Issues in Layered Multicast*, NOSSDAV 1999, June '99.

[12] Handley M., et al, *The Reliable Multicast Design Space for Bulk Data Transfer*, RFC 2887, Aug 2000.

[13] Kar K., Sarkar S. and Tassiulas L., *Optimization Based Rate Control for Multirate Multicast Sessions*, INFOCOM 2001, Apr 2001.

[14] Jorg Widmer, Mark Handley, *Extending Equation-based Congestion Control to Multicast Appliations*, SIGCOMM 2001, Aug 2001.

[15] McCanne S., Jacobson V. and Vetterli M., *Receiver-driven Layered Multicast*, SIGCOMM '96, Aug '96, pp. 117-130.

[16] Natu N., Rajagopal P., Kalyanaraman S., *GSC: A Generic Source-based Congestion Control Algorithm for Reliable Multicast*, J. of Comp. Commns, Feb 2001.

[17] Padhye J. et al, *Modeling TCP Throughput: A Simple Model and its Empirical Validation*, SIGCOMM '98, Aug '98.

[18] Rhee I. et al, *MTCP: Scalable TCP-like Congestion Control for Reliable Multicast*, INFOCOM '99, March '99.

[19] Rizzo L., *PGMCC: A TCP-friendly Single-Rate Multicast Congestion Control Scheme*, SIGCOMM 2000, Aug 2000.

[20] Rubenstein D., Kurose J., and Towsley D., *The Impact of Multicast Layering on Network Fairness*, SIGCOMM'99, Sept 1999.

[21] S. Kasera et al, *Scalable Fair Reliable Multicast Using Active Services*, IEEE Network Magazine, January/February 2000.

[22] S. Ramakrishnan, S. Kalyanaraman, J. Wen, H. Ozbay, *Effect of Time Delay in Network Traffic Control*, Short Paper, Automatic Controls Conference (ACC), 2001.

[23] K. Seada, A. Helmy, *Fairness Analysis of Multicast Congestion Control: A Case Study on pgmcc,* Technical Report 01-743, University of Southern California, CS Department, April 2001.

[24] Speakman T. et al, *PGM reliable transport protocol specification*, Internet Draft, March 2000.

[25] P. Thapliyal, Sidhartha, J. Li, S. Kalyanaraman, *LE-SBCC: Loss-Event Oriented Source-based Multicast Congestion Control*, RPI-ECSE Technical Report, 2001. Available from: http://www.ecse.rpi.edu/Homepages/shivkuma/research/papers-rpi.html#mult

[26] Vicisano L., Rizzo L. and Crowcroft J., *TCP-like congestion control for layered multicast data transfer*, INFOCOM '98, Apr '98.

[27] Whetten B. and Conlan J., *A Rate Based Congestion Control scheme for Reliable Multicast*, RMRG meeting, Jul '98.

[28] Whetten B. and Taskale G., *An Overview of Reliable Multicast Transport Protocol II*, IEEE Network Magazine, Jan/Feb 2000.

APPENDIX

I. PSEUDO CODE

event: Send packet:
    Record $T_{send}[j]$ for seq j
event: LI[i][j] received for from receiver i for seq j:
    call **estimateRTT()**
    call **LI2LEfilter()**
    if passed
      call **maxLPRFilter()**
      if passed
        call **ATFilter()**
        if passed
          $rate = rate/2$
        endif
      endif
    endif

estimateRTT() {
  $RTT_{current} = T_{current} - T_{send}[j]$
  $\delta = SRTT - RTT_{current}$
  $SRTT = RTT_{current} + 0.125 * \delta$
  $\sigma = \sigma + (0.125 * (|\delta| - \sigma)))$
}
LI2LEfilter() {
  if $((t - T_{lastPass}[i]) > SRTT + 2\sigma)$
    $T_{lastPass}[i] = t$
    pass LI as LE
  else
    filter LI
  endif
}
maxLPRFilter() {
  update $X_i, \max X_i, \Sigma X_i$
  $P(accept) = \frac{\max X_i}{\Sigma X_i}$
  Accept LE w/ probability $P(accept)$
}
ATFilter() {
  if $(CongestionFlag == TRUE)$
    filter LE
  else {
    $SilenceFlag = TRUE$
    $SilencePeriodTimer = RTT/2 + 2\sigma$
    $CongestionFlag = TRUE$
    $CongestionEpochTimer = SilencePeriod + SRTT + 4\sigma$
    Accept LE
  }
}

event: $CongestionEpochTimer$ expires:
    $CongestionFlag = FALSE$

event: $SilencePeriodTimer$ expires:
    $SilenceFlag = FALSE$

event: $RateIncreaseTimer$ expires:
    if $(CongestionFlag == FALSE)$
      $rate+ = \frac{MSS}{SRTT+2\sigma}$
    else {
      if $(SilenceFlag == TRUE)$
        no data transfer
      else
        no increase in rate
    }
    $RateIncreaseTimer = RTT + 2\sigma$