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

Sidhartha, Puneet Thapliyal, Jiang Li,

Shivkumar Kalyanaraman[1]

Rensselaer Polytechnic Institute, Troy, NY 12180

Contact Email: lij6@cs.rpi.edu

Abstract

We propose an end-to-end single-rate multicast congestion control scheme (LE-SBCC) for reliable or unreliable multicast transport protocols. A distinguishing feature of the scheme is that it is *purely* source-based, i.e., it requires no support from multicast receivers or network elements, which makes it very easy to deploy for a range of protocols. It addresses all the pieces of the single-rate multicast congestion control problem including *drop-to-zero* issues, *TCP friendliness* and RTT estimation. The scheme design consists of a cascaded set of filters and a rate-based additive-increase multiplicative-decrease (AIMD) module. These filters together transform the multicast tree to appear like a unicast path for the purposes of congestion control. The first problem is that, unlike TCP, the scheme is not self-clocked but acts upon a stream of loss indications (LIs) from receivers. These LIs are filtered to get a stream of loss events (LEs) [8] (at most one per RTT per receiver). This LE stream is further filtered to extract the maximum LEs from any one receiver. Then the scheme effects at most one rate-reduction per round trip time (RTT). A range of results (numerical, simulation and experimental) is presented and compared against the mathematical model of the scheme components. The scheme achieves all its goals in the common case of *partial LI aggregation*[2] [17], though the rare worst case of *complete LI aggregation* could be aggressive towards TCP.

1 Introduction

This paper presents a 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 any other support from network elements, receivers or in the packet format of underlying multicast transport protocols. This feature distinguishes it from recent

[1]We acknowledge the help of Jye-Young Song to implement this scheme

[2]Assuming that network elements aggregate the reverse LI flow to avoid implosion e.g. PGM model [21]

single-rate schemes like PGMCC [19], Equation-based TFMCC [11, 23], MTCP [18], Kasera et al [13] and Golestani et al[9]. 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 without touching the receivers as long as they generate minimal congestion feedback. The source-based nature of the scheme does not incur extra congestion-related (S,G) state in network elements and receivers, which can be directly exploited to support multi-source multicast applications (like interactive multi-player games). In this sense, our approach is very lightweight in terms of implementation, upgrade and management compared to the earlier schemes, while achieving comparable performance levels.

The LE-SBCC scheme builds upon our earlier work GSC [15] and Bhattacharya et al's LPRF [2, 4]. It differs from these prior schemes in that it tackles the drop-to-zero problem under high degrees of multiplexing, addresses several implementation issues [17] and has a much faster transient response. We believe that it is the first scheme to fully address all components of single-rate source-based multicast congestion control, i.e. *drop-to-zero* issues, *TCP friendliness*, RTT estimation, robustness and scalability up to 10,000 receivers.

LE-SBCC belongs to the class of single-rate schemes, i.e., which operate at the rate dictated by the worst path on the tree. Similar to PGMCC [19], it is aimed at small-medium scale receiver sets (up to 10,000 receivers). There exist another distinct class of congestion control schemes which are *multi-rate and receiver-based*. In such schemes, 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 [14, 22, 5, 7, 3]) or network-feedback driven (eg: [12, 20]).

In contrast to our work, the network-feedback driven schemes aim to achieve max-min fairness, but require complex support at all nodes in the multicast tree. The end-to-end receiver-based approach is attractive because of its scalability to very large group sizes. But recent analysis by Gopalakrishnan et al [10] suggests that the end-to-

end receiver-based approaches are coarse-grained (in terms of join/leave latency and traffic on each group) and may have stability/fairness problems. Our scheme does not fall into this class, and is fine-grained in terms of control and traffic granularity. The combination of source-based and receiver-based approaches is largely an open problem.

2 LE-SBCC: Scheme Description

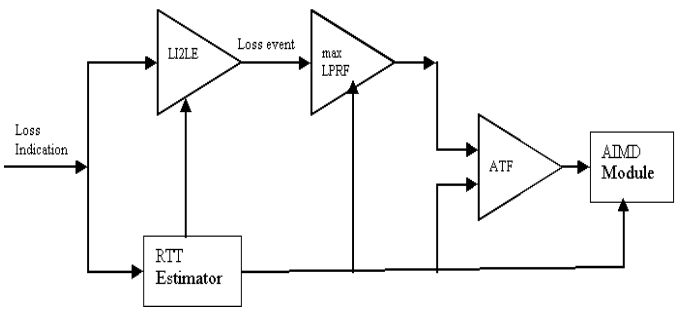


Figure 1: Cascaded Filter Model of LE-SBCC implemented at the multicast source

The LE-SBCC scheme consists of a *purely source-based* cascaded set of filters and RTT estimation modules feeding into a rate-based additive increase/multiplicative decrease (AIMD) module (Figure 1). These filters together transform the multicast tree to appear like a unicast path for the purposes of congestion control. Also, unlike TCP, the scheme is not self-clocked but acts upon a stream of loss indications (LIs) from receivers. The filters are then designed to 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[23, 4]. This occurs because the multicast flow receives LIs from multiple paths and may not filter LIs sufficiently.

TCP-unfriendliness: is the problem of reacting to *less LIs than a hypothetical TCP flow would on the worst loss path* [6, 23, 11]. TCP-friendliness is a subject of current debate[8, 19, 11]. In particular, Padhye et al's TCP throughput formula [16] requires the definition and estimation of "loss rate" and "RTT" of the paths. The unicast scheme, TFRC[8] 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, this paper will also first derive LEs from LIs before responding to congestion.

2.1 Cascaded Filter Model

Figure 1 illustrates the outline of the building blocks of the scheme implemented at the multicast source. The cascade of filters converts a stream of loss indications (LIs) from receivers into per-receiver LEs, filter the LEs further, and finally outputs a stream of rate-reduction indications (RRs) to the rate-based AIMD module (at most one RR per RTT). The AIMD module performs multiplicative rate decrease upon receipt of a RR and performs rate increase 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 [8].

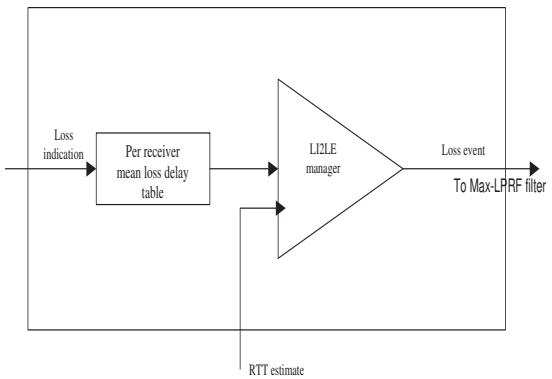


Figure 2: Loss Indication to Loss Event Filter (LI2LE)

All filters and the AIMD module need RTT estimates which is fed by the RTT estimation module developed in our earlier work [15]. It works similar to the TCP timeout procedure i.e. it calculates a smoothed RTT ($SRTT$) and a mean deviation which approximates the standard deviation σ. However the set of samples is pruned to exclude a large fraction of samples which are smaller than $SRTT/2$ (i.e. smaller by an order of magnitude) to bias the average RTT higher. The rate increase uses intervals of length $SRTT + 2\sigma$ (see [15]). Observe that this LI-driven RTT estimation procedure will tend to opportunistically measure the worst RTT of paths which are generating more LIs. That is, the procedure tends to estimate the worst RTT from the *currently congested subtree*. Therefore, the de-coupling of RTT estimation from worst-case loss rate estimation does not lead to significant suboptimality in practice (unlike what a naive analysis of the TCP-throughput formula [16] would predict). Note that *$SRTT$ is a global value, not maintained on a per receiver basis.*

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 $SRTT$[3] $+ 2\sigma$ of $T_{LastPassed}$, 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 [9]); which leads to large burst losses with drop-tail queues (see section 3.1.3).

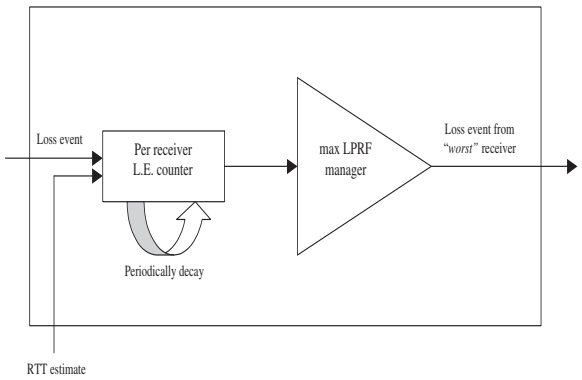


Figure 3: Max-LPRF: max-Linear Proportional Rate Filter

Max-LPRF: The goal of this filter is to pass, on the average, the number of LEs corresponding to what the source would have received from the worst case receiver. This worst-case loss rate estimation problem has been studied earlier in the literature (see [2] and references therein). Our filter is a extension of Bhattacharya et al's Linear Proportional Response Filter[2] (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.

We found that we could not use the LPRF filter directly because of a number of issues. Firstly, *LPRF* did not quickly adapt to sudden increases in worst-case loss rates. Second, *LPRF* works with LIs instead of LEs. Our subsequent arguments show that using LEs is much superior to using LIs.

The *Max-LPRF* works as follows: Assuming X_i is the count of LEs from receiver i, this probabilistic 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 $SRTTs$. These numbers were determined empirically by running several *ns* simulations. The optimal choice of these parameters is still open to

[3] $SRTT$ = average of RTT samples similar to the variable used in TCP

research. *Max-LPRF tracks the worst path better than LPRF and is the crucial building block for drop-to-zero avoidance.*

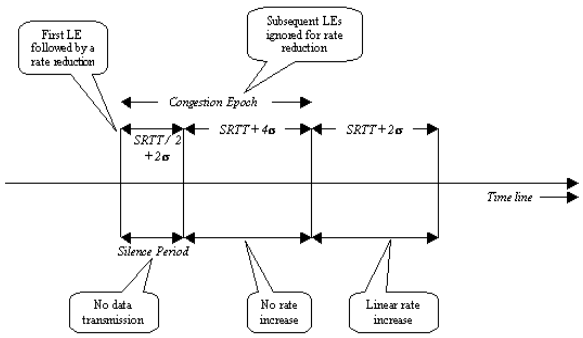


Figure 4: Adaptive Time Filter (ATF)

Adaptive Time Filter (ATF): This filter shown in Figure 4 is a modification of the time-filter introduced in our earlier work [15]. 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. As suggested in [15], the goal is to reduce the probability of losing any control traffic or retransmissions during this phase.

In summary, for ideal operation, the scheme expects: (a) at least one LI per receiver per RTT if packets are lost in that RTT (i.e. at least one LE per RTT) and (b) timely generation of feedback by receivers/network elements to allow reliable RTT sampling [4]. In general, if expectations (a) and (b) are not satisfied completely or satisfied in an unreliable manner, the scheme performance will degrade. We examine such performance effects in the following sections.

3 Performance Evaluation

We have evaluated the performance of LE-SBCC to test for Drop-to-Zero avoidance, TCP-friendliness and LI aggregation effects. We use *ns-2* simulations, Simple Markov chain based modeling and Linux-based implememtation to test our scheme. We have also developed a C-program to numerically analyze our scheme for Drop-to-Zero avoidance in medium to large scale trees [17]. Detailed LI aggregation effects were also analyzed similarly [17].

3.1 Drop-to-Zero Avoidance

Recall that 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. We have analyzed the drop-to-zero issue using *ns-2*, numerical simulations and Markov Chain based theoretical analysis.

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

3.1.1 *ns*-2 Simulations Illustrating Drop-to-Zero Avoidance

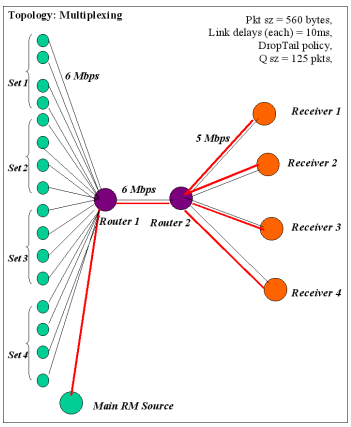


Figure 5: Topology to Test Drop-to-Zero Avoidance and TCP-friendliness

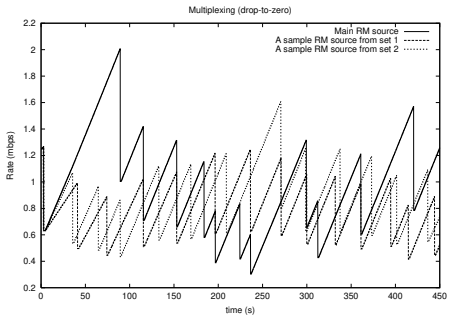


Figure 6: Drop-to-zero Avoidance Results

Consider a set of reliable multicast (RM) flows that goes to receiver 1 through receiver 4 as shown in Figure 5. The capacity of the links connecting the sources (on the left) to "Router 1" are 6 Mbps each. The link from "Router 1" to "Router 2" has 6 Mbps for each flow. There is a *Main RM* source which sends data to receivers 1 through 4. The links from "Router 2" to each of the receivers are bottlenecks at 5 Mbps each. A set of 4 background unicast flows implementing the LE-SBCC algorithm compete on each of the 4 bottlenecks links. The *unicast* flows in "Set 1" through "Set 4" compete on the bottleneck links to "Router 2-Receiver 1" through "Router 2-Receiver 4" respectively. The fair rate for all flows is therefore 1 Mbps. 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 reasonable degrees of multiplexing on each path. In effect, the loss rate on such congested paths is less dependent on the rate change of any single flow on the path (especially the multicast flow).

Figure 6 (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. More importantly, the multicast flow does not get beaten down because of the independent loss indications received from each path. This is a non-trivial illustration of drop-to-zero avoidance. Observe that the same topology also illustrates the basic applicability of LE-SBCC to multi-sender multicast, because the competing unicast flows use the same congestion control scheme. Each sender's multicast traffic is con-

trolled as if it were the only sender.

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 main 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. When we go back to the graph in Figure 6 and count the number of rate-reductions in multicast as well as unicast flows, *we observe that all flows have 12 rate-reductions.* This is another measure of the fairness of the scheme to AIMD-based background traffic. The issue of TCP fairness is further explored in section 3.2.

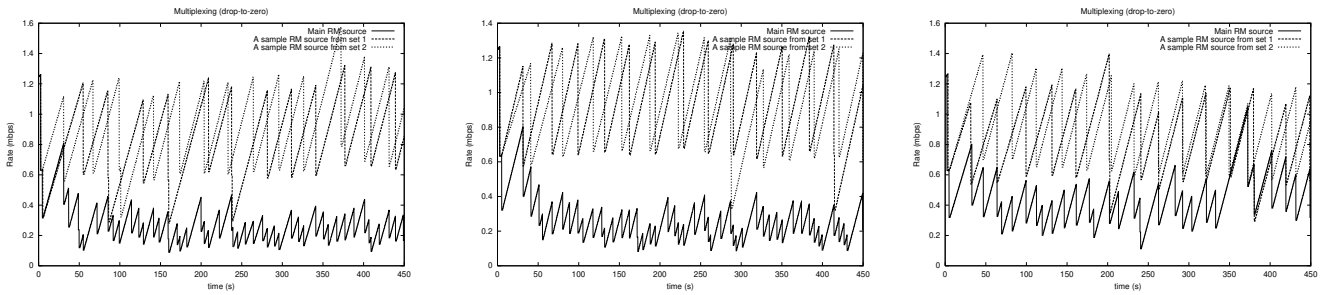
As discussed in section 3.1.3, the dramatic difference between the number of LIs and LEs is because of bursty losses caused by the lack of self-clocking (unlike TCP), rate-based nature of control and the use of sizable drop-tail buffers.

3.1.2 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 the important role that each filter plays in our scheme, we repeat the simulations for the configuration shown in Figure 5 for three different cases mentioned in Figure 7. In all these experiments all other parameters are kept constant as before. The results are illustrated in Figure 7. Clearly, we observe the 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 LI's, and their correct combination avoids the drop-to-zero problem (Figure 6).

3.1.3 Theoretical Analysis of LE vs LI Probability

Recall that our scheme uses LEs instead of LIs for the max-LPRF filter and ultimately in rate reduction decisions. Therefore, our scheme is critically dependent on the number of LE's passed by filters at the source. Thus it becomes important to study the relationship between LI's and LE's to get an



(a) Removing LI2LE & maxLPRF (b) Removing only maxLPRF (c) Removing only LI2LE

Figure 7: Effects of Removing a Subset of Filters from the LE-SBCC Cascade

understanding of when it is really important to use LEs instead of LIs.

Consider the following Markov Chain model for unicast transmission. Assume that the RTT is constant and that the system moves in cycles of RTT. In each RTT, the source transmits with a constant rate, and may experience one or more packet losses. At the end of each RTT, it changes its rate based upon the AIMD policy and depending upon the receipt of LEs. The system may be modeled as a markov chain with N discrete states (from 1 to N), where the state x specifies the number of packets sent during that RTT. The process starts in state 1. When a loss event (LE) is encountered within an RTT, the system transitions to state $\lfloor x/2 \rfloor$ but no smaller than 1. When there is no loss event in an RTT, the system moves to state $x + 1$. This markov chain model is illustrated in Figure 9. We analyze this chain for bursty packet loss case and estimate the ratio of per-packet LI probability (p_{li}) to per-packet LE probability (p_{le}) . An analysis for non-bursty case can be found at [17].

We define p as the probability that a packet is lost given that either it is the first packet in the RTT or the preceding packet in the RTT is not lost [16] (Figure 8), and a as $1-p$. The LE probability (p_{le}) is thus $\sum s(x)(1-a^x)$ where $s(x)$ is the probability of being in state x. In the general case, under steady state,

$s(k) = s(k-1) \times a^{k-1} + s(2k) \times (1-a^{2k}) + s(2k+1)(1-a^{2k+1})$

More detailed analysis considering boundary and special cases can be found in [17].

This markov chain cannot be solved for an exact solution. However, we did a Matlab analysis to numerically solve the chain for different large N's. The solutions yield the steady state probabilities

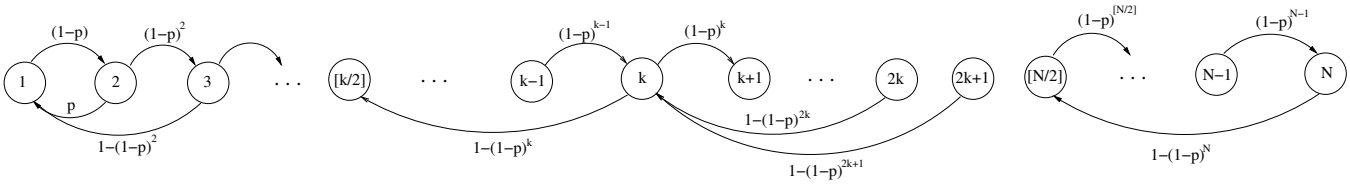


Figure 9: State transition of packets sent per RTT

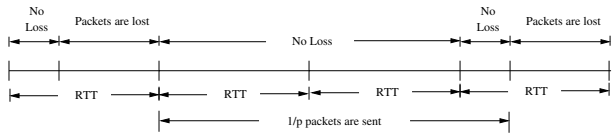


Figure 8: p's semantics

$s(x)$ and hence the LE probability p_{le}. Furthermore, we can use the following formula to derive the LI probability p_{li} from p:

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

Figure 10 (a) and (b) plot the ratio of p_{li} and p_{le} for p_{li} from 0 to 1 and for low p_{li} respectively. The ratio $\frac{p_{li}}{p_{le}}$ is huge (9-90) 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.

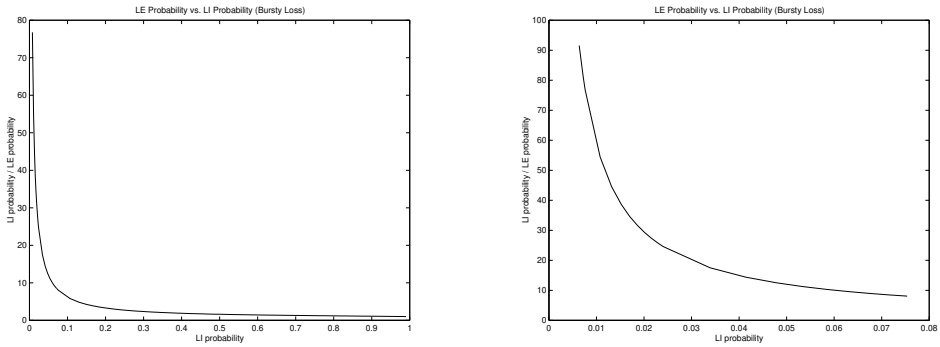
Comparing the theorectical $\frac{p_{li}}{p_{le}}$ ratio to that observed in simulation (see section 3.1.1), we observe that the model gives the same order of magnitude as the simulation results. Therefore, we can conclude that the LI2LE filtering is critical in our cascade filter design.

3.2 TCP-friendliness

TCP-friendliness is a subject of current debate [8, 19, 11]. 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. LE-SBCC would be TCP-unfriendly if either of the TCP or RM flow is beaten down. In Figure 5, 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.

***ns* simulation results:** The simulation results for these two cases is shown in Figures 11(a) and 11(b) [5]. The dips by the TCP flow close to zero represent timeouts. Again observe that

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



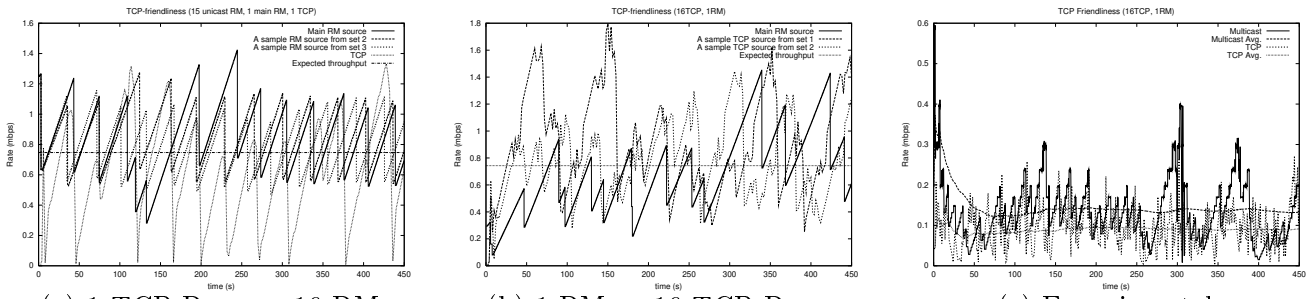
(a) Bursty loss (All loss probabilities) (b) Bursty loss (Low loss probabilities)

Figure 10: Ratio of LI and LE probabilities

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 ($1.22MSS/RTT\sqrt{p_{le}}$). Recall that p_{le} 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.

Experimentation results: The scheme is implemented in a Redhat Linux 7.0 testbed. The topology used is identical to the one used for the *ns* simulations as described above. Again the RM flows are fair to the competing TCP flow (Figure 11(c)). More details on this are presented in the section on implementation issues.

For the purpose of having a measure of TCP friendliness, we aim to reduce the multicast problem to a unicast one with the unicast receiver having the worst LE probability p_{le} among the multicast group. The RTT of the hypothetical unicast receiver is that of the most congested subtree. The hypothetical unicast receiver is probabilistically determined, and in general, may change with time. Then using the simple throughput formula for unicast proposed by Mathis et al ($1.22MSS/RTT\sqrt{p_{le}}$), we do a post-processing of the simulation data to compare the throughput of our scheme with the theoretical throughput. Clearly from the Figures 11(a) and 11(b), the rate values oscillate about their mean shown by horizontal dotted line. Their mean rate is close to the theoretical mean. The fact that the average



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

Figure 11: TCP Friendliness (Simulation and Experiment)

throughput of the RM flows is equal to the unicast flows is a good measure of the TCP friendliness of our scheme. This further demonstrates the validity of choosing LE rather than LI probabilities in our scheme.

3.3 Implementation Results

The scheme was integrated into a simple user-level PGM [21] implementation. PGM was chosen as it is one of the standard multicast transport protocols. *However, LE-SBCC is not dependent on PGM.* PGM is emulated at the user-level using UDP sockets with multicast enabled on a Redhat 7.0 Linux-based testbed. Routers were configured with *mrouted* over an Ethernet, a delay component was written for controlling delays, and link speeds were throttled down using the traffic control package *tc* (available with the diffserv-linux distribution). The configuration used is the same as in Figure 5, but with RTT of 0.5 ms; minimal buffers, bottleneck rates of 1 Mbps (for *tc* shaper configuration and Ethernet-related reasons) and average NAK backoff timer as 100 ms at receivers [21]. The Linux socket library allows only coarse-grained timers (10 ms) and we made changes for parts of the code needing fine-grained timing [17]. The TCP version used in Redhat 7.0 is TCP SACK. A sample experimental result is shown in Figure 11(c) which illustrates both TCP friendliness and drop-to-zero avoidance of the scheme. The graph plots the case for 1 RM (main RM) vs 16 TCP flows, with a selected TCP flow. The RM rate is reasonably TCP friendly, but gains somewhat more bandwidth in part due to a) the lack of self-clocking in the rate-based AIMD b) TCP timeouts triggered due to small window values and c) user-level implementation of the LE-SBCC scheme.

4 Summary and Conclusions

We have presented an *LE-oriented, purely source-based multi-sender multicast congestion control scheme (LE-SBCC)* for reliable and unreliable multicast transport . The model is a simple cascade of filters followed by an AIMD module. In general the AIMD module may also be replaced by a source-based TFRC scheme [8] or rate-based binomial scheme [1]. With partial or no LI aggregation, the scheme correctly filters the LIs for AIMD; avoids the drop-to-zero problem while being TCP-friendly. With substantial 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. The scheme is simple to implement and deploy since it is completely contained at the source[6]. The pseudo code for LE-SBCC may be found at [17].

References

[1] Bansal D. et al, "Binomial Congestion Control Algorithms," *INFOCOM 2001*, Apr 2001.

[2] Bhattacharyya S. et al, "A Novel Loss Indication Filtering Approach for Multicast Congestion Control," *J. of Comp. Commns*, Feb '01.

[3] 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.

[4] Bhattacharya S., et al, "The Loss Path Multiplicity Problem in Multicast Congestion Control," *INFOCOM '99*, March '99.

[5] Bolot J.C., et al, "Scalable Feedback Control for Multicast Video Distribution in the Internet," *SIGCOMM '94*, Aug '94.

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

[7] Byers J.W., et al, "FLID-DL Congestion Control for Layered Multicast," *NGC 2000*, Nov '00.

[8] Floyd S. et al, "Equation-Based Congestion Control for Unicast Applications," *SIGCOMM '00*, Aug '00.

[6]Thanks to Supratik Bhattacharyya, Sonia Fahmy, Luigi Rizzo, Mark Handley, Lorenzo Vicisano, Jon Crowcroft and Brian Whetten for extensive comments on various versions of this work.

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

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

[11] Handley M., Floyd S., Whetten B., "TCP Friendly Multicast Congestion Control," *RMRG meeting*, June '99.

[12] Kar K., Sarkar S. and Tassiulas L., "Optimization Based Rate Control for Multirate Multicast Sessions," *INFOCOM '01*, Apr '01.

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

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

[15] Natu N., Rajagopal P., Kalyanaraman S., "GSC: A Generic Source-based Congestion Control Algorithm for Reliable Multicast," *J. of Comp. Commns*, Feb '01 (to appear).

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

[17] Puneet T. et al, "Technical Report: LE-SBCC: Loss-Event Oriented Source-based Multicast Congestion Control," Available at http://www.cs.rpi.edu/ lij6/papers/le-sbcc-tr.ps.gz April 2001

[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 '00*, Aug '00.

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

[21] Speakman T. et al, "PGM reliable transport protocol specification," *Internet Draft*, March '00.

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

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