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

Puneet Thapliyal, Sidhartha, Jiang Li, Shivkumar Kalyanaraman[1]
Rensselaer Polytechnic Institute, Troy, NY 12180
Contact Email: shivkuma@ecse.rpi.edu

Abstract

We propose a *loss-event oriented purely source-based single-rate* multicast congestion control scheme (LE-SBCC) for reliable or unreliable multicast transport protocols. The scheme is easy to deploy and addresses the *drop-to-zero* and *TCP friendliness* problems [15, 19]. It consists of a simple cascaded set of filters and a rate-based AIMD module. Loss indications (LIs) from receivers are converted into loss events (LEs) [6]; the maximum LEs from any path are chosen, and at most one rate-reduction per round trip time (RTT) is effected. The scheme performs well in the common case of *partial LI aggregation*, though the worst case in the rare instance of *complete LI aggregation* is aggressive towards TCP. A range of results (numerical, simulation and experimental) is presented. The scheme has been implemented using PGM as the underlying RMT protocol over a Redhat 7.0 Linux testbed.

1 Introduction

This paper presents a single-rate loss-event oriented *purely source-based* congestion control scheme (LE-SBCC) for reliable or unreliable multicast transport protocols. The scheme can function without any *new* support from receivers, network elements or packet-header support and can leverage the underlying loss indications (LI) provided by protocols like PGM or RMTP [16, 20]. LE-SBCC is a single-rate scheme like PGMCC [15], LPRF [2, 3], Equation-based TFMCC [9, 19], MTCP [14] and Golestani [7]. In contrast to LE-SBCC, these schemes require *new* support at the source, receivers and transport packet-format level. Examples of multi-rate schemes (which are not source-based) include RLM, RLC, FLID-DL etc [11, 18, 5, 10].

Our 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). The filters are designed to address the following key problems:

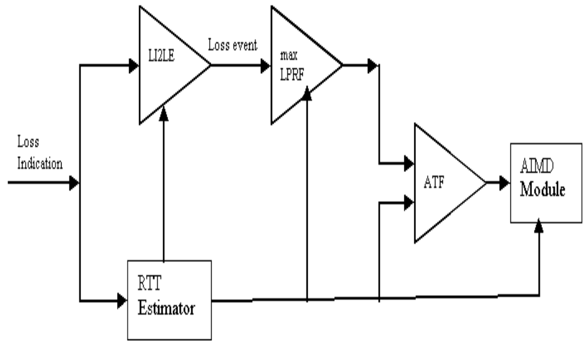


Figure 1: Cascaded Filter Model of LE-SBCC

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[19, 3]. This occurs because the multicast flow receives LIs from multiple paths and may not filter LIs sufficiently. In an earlier work [12], 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* [4, 19, 9]. TCP-friendliness is a subject of current debate[6, 15, 9]. In particular, Padhye et al's TCP throughput formula [13] requires the definition and estimation of "loss rate" and "RTT" of the paths. The unicast scheme, TFRC 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.

On the subject of RTT estimation, PGMCC and TFMCC [15, 9] have receivers estimate RTT and drive the control based upon paths having the maximum value of $(RTT\sqrt{q})$, where q is the *per-packet LE probability.* Our proposed scheme LE-SBCC computes $\max RTT$ and $\max q$ concurently and uses both these estimates to drive the AIMD module. This decoupled use of $\max RTT$ and $\max q$ represents a sub-optimal approach in general[15, 9, 19, 17].

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

2 Cascaded Filter Model

Figure 1 illustrates the outline of the building blocks of the scheme implemented at the multicast source. The cascade of filters convert a stream of loss indications (LIs) from receivers into per-receiver LEs, filter the LEs further, and finally output 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 an RR and performs rate increase each time an 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 [6]. 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. It 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[2] $T_{LastPassed}$, 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 [15] or Golestani [7]); which leads to large burst losses with drop-tail queues[3]

Max-LPRF: This filter is a extension of Bhattacharya et al's Linear Proportional Response Filter[2]. 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 the 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 $SRTTs$. The O(N) state requirements of both LI2LE and Max-LPRF are not a big issue because single-rate schemes are typically targeted for a small-medium scale (< 10000 receivers). *Max-LPRF tracks the worst path better than LPRF and is the crucial building block for drop-to-zero avoidance.* It operates on per-receiver LE counts since they differ dramatically from LI counts in drop-tail networks with no self-clocking.

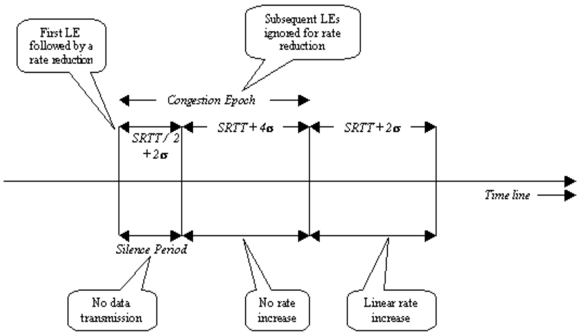


Figure 2: Adaptive Time Filter (ATF)

Adaptive Time Filter (ATF): This filter shown in Figure 2 is a modification of the time-filter introduced in our earlier work [12]. 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 an optional *"silence period"* of $SRTT/2 + 2\sigma$ when no packets are sent. As suggested in [12], the goal is to reduce the probability of losing any control traffic or retransmissions during this phase.

All filters and the AIMD module need RTT estimates which is fed by the RTT estimation module developed in our earlier work [12]. 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 [12]) while other functions differ in their use of σ as described earlier.

In summary, the scheme expects: (a) at least one LI per receiver per RTT if packets are lost in that RTT and (b) timely generation of feedback by receivers/network elements to allow reliable RTT sampling[4]. The scheme is sensitive to arbitrary delays introduced by receivers/network-elements in generating an LI corresponding to a lost packet. Further *LI aggregation* like in PGM [16] itself is a form of filtering which suppresses receiver IDs, timing information and reduces LI or LE counts, thus conflicting with (a) and (b) above[5]. We examine the performance effects in the following sections.

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

[3] During overload the queue increases quadratically with time in rate-based AIMD.

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

[5] LIs carry RTT information which affects the RTT estimation, and LEs carry receiver IDs which affects the max-LPRF filter.

3 Performance Evaluation

We have evaluated the performance of LE-SBCC using simple modeling, numerical simulation (upto 400 receivers), *ns-2* simulation, linux-based implementation/experimentation (high multiplexing degrees and upto 10s of receivers). More detailed performance analysis is available in a technical report [17].

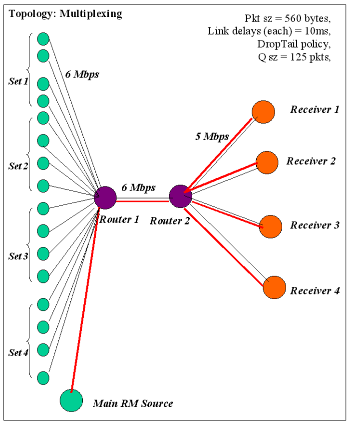


Figure 3: Topology to Test Drop-to-Zero Avoidance and TCP Friendliness

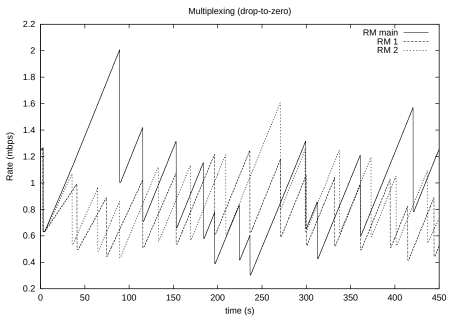


Figure 4: Drop-to-zero Avoidance Results

3.1 Drop-to-Zero Avoidance

Consider a multicast flow to 4 receivers as shown in Figure 3. 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. In Figure 3, 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, and packet size is 560 bytes. Figure 4 shows that the multicast flow (solid line) competes fairly with a selected subset of unicast background flows (dotted lines) and the average rate is close to 1 Mbps, which is a non-trivial illustration of drop-to-zero avoidance.

The main RM flow transmitted a total of 167762 pkts in 450s and received 949 LIs (i.e. a low loss rate). After LI2LE filtering, there were only 32 LEs, a dramatic reduction! As discussed earlier, the bursty losses are caused by the lack of AIMD self-clocking in combination with drop-tail buffers. The Max-LPRF and ATF further filter LEs to result in a total of 12 rate-reductions which is the same as a unicast flow as shown in Figure 4. This illustrates the important role of the LI2LE and Max-LPRF filters in the drop-to-zero avoidance process.

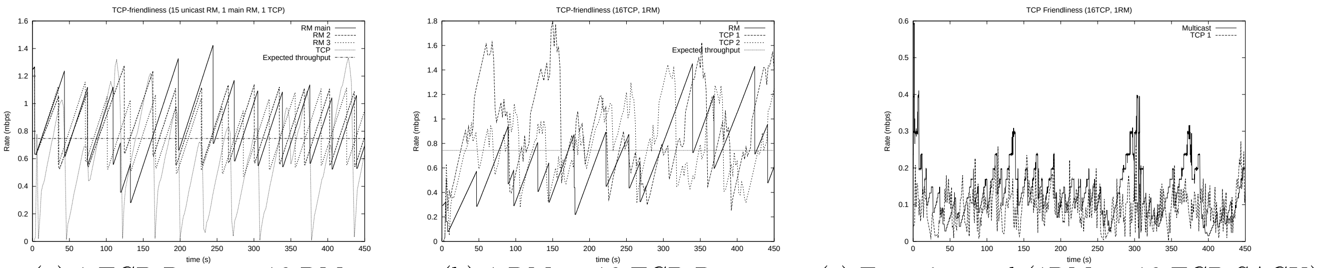
3.2 TCP-friendliness

In the topology of 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. This topology is used to illustrate TCP friendliness. The simulation results for these two cases is shown in Figures 5(a) and 5(b) [6]. The dips by TCP close to zero represent timeouts. Again, the main RM flow (solid line in both graphs) shares the bottleneck(s) fairly with the competing TCP flow(s). The horizontal dotted line is the expected TCP throughput plotted post-simulation using the simplified TCP equation ($1.22 MSS/RTT\sqrt{q}$).

3.3 Implementation Results

The scheme was integrated into a simple user-level PGM [16] implementation. 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 3, 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 [16]. The Linux socket library allows only coarse-grained timers (10 ms) and we made changes for parts of the code needing fine-grained timing. The TCP version used in Redhat 7.0 is TCP SACK. A sample experimental result is shown in Figure 5(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

[6] To avoid cluttering, the TCP thruput is sampled every 2s



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

Figure 5: TCP Friendliness (Simulation and Experiment)

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.

3.4 LI Aggregation Effects

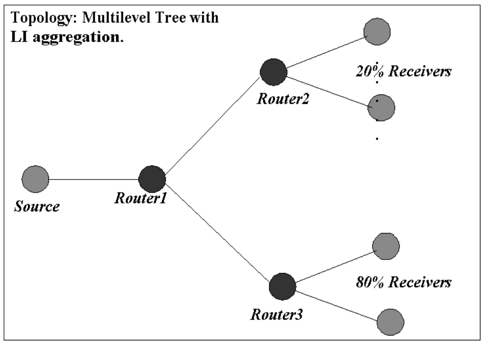


Figure 6: Topology to Illustrate LI Aggregation Effects

As explained earlier, LI aggregation has an effect on the number of LEs and timing information reaching the source. To evaluate these effects, consider the simple multi-level topology in Figure 6. Observe that the number of receivers and their loss rates can be varied. Consider three cases: (a) *no LI aggregation* performed at the routers (b) LI aggregation performed at router 2 and router 3 only (i.e. *partial LI aggregation*) (c) LI aggregation performed at all routers (i.e. *complete LI aggregation*).

Since *ns-2* simulation does not scale beyond tens of nodes, we wrote a C program to numerically simulate the scheme using a model based on fixed loss probabilities of receivers, fixed RTT and packets assumed to be sent in rounds of length RTT. The LIs for a round are computed and informed to the source at the end of the round, and the control decision is enforced in the next round. In Figure 6, we had 20% of receivers in the subtree headed by router 2, and 80% of receivers in the subtree headed by router 3. One receiver in the first subtree has a loss (LI) rate of 4% and all other receivers (in both subtrees) have a loss rate of 2%. A total of 6000 packets were transmitted which translates to a max-LI count of 240 and the observed max-LE count is roughly 150-160 (without aggregation)[7].

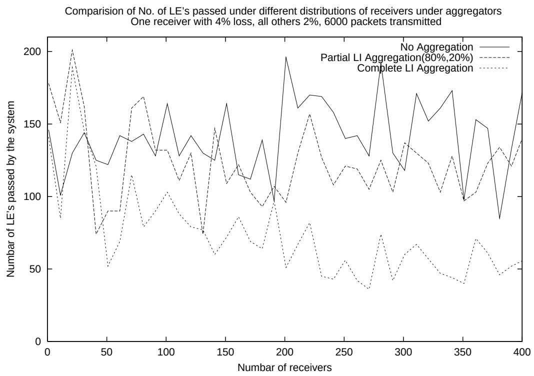


Figure 7: LI Aggregation Effects: LEs Passed by Cascade of Filters

The numerical simulation for cases (a), (b), (c) above are shown in Figure 7. The figure illustrates the LEs passed by the cascade of filters (one curve for each case), plotted against the number of receivers (upto 400 receivers). Except for the completely aggregated cases, each curve is roughly flat which means that the Max-LPRF passes roughly the same number of LEs irrespective the number of receivers or the total number of LEs generated. Two curves (no aggregation and partial aggregation) are nearly identical which shows that the scheme works well even with non-trivial partial aggregation. We have verified these results for confidence with many randomized variations of the simulation [17]. This is because the max-LPRF scheme is *more sensitive to LE counts and*

[7] LI vs LE issues with uniform and bursty loss models are discussed in [17]

receiver IDs rather than LI counts/IDs. However, for the case of complete aggregation, the number of LEs passed is lower, which implies that the scheme will be more aggressive than an equivalent TCP. *In all cases, it does not incur the drop-to-zero problem*, i.e. passing more than 240 LEs.

4 Summary and Conclusions

We have presented an *LE-oriented, purely source-based 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 [6] 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[8].

References

[1] Bansal D. and Balakrishnan H., "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 (to appear).

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

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

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

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

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

[7] Golestani J., "Fundamental observations on Multicast Congestion Control in the Internet," *INFOCOM 1999*,

[8] Handley M., et al, "The Reliable Multicast Design Space for Bulk Data Transfer," *RFC 2887*, Aug '00.

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

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

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

[12] 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).

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

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

[15] Rizzo L., "PGMCC: A TCP-friendly Single-Rate Multicast Congestion Control Scheme," *SIGCOMM '00*, Aug '00.

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

[17] Thapliyal P. et al, "LE-SBCC: Loss-Event oriented Source-based Multicast Congestion Control," *Technical Report*, 2001. Available from: http://www.ecse.rpi.edu/Homepages/shivkuma/

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

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

[20] Whetten B. and Taskale G., "An Overview of Reliable Multicast Transport Protocol II," *IEEE Network Magazine*, Jan/Feb '00.