More Thoughts on Reference Simulations for Reliable Multicast Congestion Control Schemes

John Byers* Mark Handley† Gavin Horn‡ Michael Luby§ Lorenzo Vicisano¶

August 15, 2000

Abstract

This document summarizes the output of a meeting held at Digital Fountain, Inc. on August 8, 2000 to expand upon the work initiated by Mark Handley to develop a set of reference simulations for reliable multicast congestion control schemes [1]. This outline sketches a set of additional experiments which the meeting participants generally deemed useful, and provides guidelines for setting up appropriate *ns* simulations in which to perform these experiments. This work is not a standalone document; our plan is to integrate the suggestions it contains into an updated version of the reference simulations draft.

1 Methodological Considerations

Before a successful experiment can be undertaken in *ns* to measure the performance of a multicast congestion control strategy, there are a substantial number of parameters which must be set carefully to ensure that the experiment is measuring something useful. We enumerate these here, beginning with principles that apply to congestion control studies in general, then continuing with principles specific to multicast:

- Correct queue sizing. The minimum queue size for which TCP congestion control behaves according to ideal sawtooth behavior is one in which the bottleneck gateway has a queue sized equal to the bandwidth-delay product of the connection. For drop-tail gateways, we recommend testing at that setting, and with a queue size four times larger, i.e. four times the bandwidth-delay product. RED gateway settings are discussed in more detail below

- TCP type and settings. In order of importance, we recommend running tests with TCP SACK and TCP Reno; the former providing robustness to multiple losses per window, the latter being the most widely deployed version of TCP today. The clock granularity of TCP must be set appropriately. The TCP receiver window should be set to be large enough to avoid impacting congestion control algorithms (10000 packets). Unless the goal of the experiment is to explore interaction with TCP slow start dynamics, sufficient time (15 seconds) should elapse after a TCP connection is established before measurements are taken.

- Bottleneck bandwidth. It is imperative to test congestion control strategies over a wide range of bottleneck bandwidths. We recommend testing with bottleneck bandwidths of 200Kbps, 500Kbps, 1Mbps, 8Mbps and 64Mbps.

*Boston University and Digital Fountain, Inc.
†AT&T Center for Internet Research at ICSI
‡Digital Fountain, Inc.
§Digital Fountain, Inc.
¶Cisco Systems, Inc.

- Randomization. *ns* and other event-driven simulators are prone to simulation artifacts such as phase effects that may not be present in the real world. We recommend the use of randomization in simulations to remove these effects. Some tactics for doing so include the use of RED gateways, adding a random error term to packet transmission times, or injecting a small amount of random cross-traffic on the forward path.

- Queueing policies. The impact of both drop-tail and RED gateways on the congestion control protocol should be studied, compliance with drop-tail being the more important given its wider deployment at present. When RED gateways are used, a number of parameters must be set carefully to provide a meaningful simulation. Recommended settings include: sizing the queue to be twice the bandwidth-delay product, setting *minthresh* to be 5% of the queue size and *maxthresh* to be 50% of the queue size, setting *maxp* to be approximately equal to the standing queue size, or roughly (minthresh + maxthresh) / 2, and turning the gentle setting on.

- Multicast routing protocol setup. *ns* provides a number of possible multicast routing options, none of which faithfully reproduce the behavior of these protocols on real networks. Particularly problematic is the failure of *ns* to simulate control traffic correctly. We recommend using the PIM-SM centralized routing setup, on top of which control message latency is simulated (see next item).

- Correct modeling of join and leave latencies. *ns* does not provide an accurate model for propagation of IGMP join and leave requests, nor does it provide an accurate model of latencies of PIM control messages. Leave latencies can be especially substantial and must be tested. We recommend tests which involve a four to six second IGMP leave latency and PIM prune messages which take three seconds to traverse a LAN. These latencies are in addition to hop-by-hop propagation delay.

- Pruning traffic back to the source. Protocols using layered multicast for congestion control must be careful to prune unwanted traffic all the way back to the source to avoid unintended side effects. In some situations, co-locating a router with the source is a simple way to perform this pruning step.

2 Experiments

We now provide a list of potentially interesting and useful experiments with which to test a congestion control strategy. There are several principles which loosely guided the makeup and ordering of these tests. First, we intended the test suite to begin with tests of very basic principles, and to have later tests become richer in scope, often with the assumption that earlier tests had completed correctly. Second, virtually every experiment was designed with a specific point in mind, and the names we attribute to the experiments attempt to reflect this. Finally, for the advanced tests, we made some effort to classify them into coarse classifications, which were tests of 1) scaling behavior, 2) heterogeneity and 3) fairness.

In most cases, the tests are equally valid for single-rate and multi-rate congestion control schemes, although the expected outcomes of the experiments in those two domains may vary. Unless otherwise specified, all of the methodological considerations described in the previous section apply. When not otherwise specified, we use a default setting of 10ms latency on all links, 1KB packet size including headers and 100Mbps bandwidth on links which are not bottleneck links.

The experiments we employ have two very different sources of packet loss. In some experiments, we have a large amount of available bandwidth to support the flows and employ random dropping of packets at links to achieve loss. In other experiments, we study the queuing behavior of the system, dropping packets only when queues overflow (or in the case of RED, only when queues elect to drop packets).

1. Sanity check. A single multicast sender and a single multicast receiver begin to communicate over a three hop path with bottleneck bandwidth 1Mbps. After 30 seconds, a CBR flow emits cross traffic over the bottleneck link at rate 500Kbps. After 60 seconds, the CBR flow decreases its rate to 250Kbps. Expected behavior: The multicast flow should adapt to the available bottleneck bandwidth smoothly and efficiently.

2. Correlated loss. Referring to Figure 1 in Mark's original Reference Simulations writeup, a sender transmits to three receivers over a binary tree of depth 3 (the link nearest the sender is shared by all three paths). Then we run three different experiments. In experiment A, the shared link experiences random loss of 10%. In experiment B, the shared experiences 0% packet loss, but the binary branches at the next level **both** experience 10% packet loss. In experiment C, the three links closest to the receivers experience 10% packet loss[1]. Expected behavior: The multicast flows should produce similar mean bandwidth in all three experiments whether the loss is correlated (A) or uncorrelated (B and C). This mean bandwidth should be approximately equal to that of a TCP connection along the same path (tested later).

3. Heterogeneous loss. Referring to the same figure, reproduce experiment (B) from the loss correlation experiment, but reduce the link loss rate en route to receiver 3 to 5%. Then, rerun the experiment after separating the remaining 10% loss segment into two 5% loss segments, the second of which is used only by receiver 1. In the latter configuration, receiver 1 has two 5% loss segments, receiver 2 has a single 5% loss segment shared with receiver 1, and receiver 3 has an uncorrelated loss segment of 5%. Expected behavior: For single-rate multicast congestion control, the mean throughput should be similar to that of a TCP connection to the receiver experiencing 10% loss in both experiments. For multi-rate congestion control, paths to receivers with 5% loss rates should ideally receive commensurately better throughput than paths to those receivers experiencing 10% loss rates.

4. Heterogeneous latency (hard). Take a topology with a single multicast sender and two receivers. A 100 Mbps link with 10% packet loss is placed along the shared segment of the tree. The independent segments of the tree each have 100Mbps bandwidth, 1% packet loss rates, but have latencies which differ by an order of magnitude. The link to receiver 1 has 20ms latency and the link to receiver 2 has 200ms latency. Measure scenarios in which receiver 1 arrives first and in which receiver 2 arrives first. Desired behavior: Deliver mean throughput to both receivers which is approximately equal to that of a TCP connection along the higher latency path. For multi-rate congestion control, simultaneously deliver better performance to receiver 1.

5. Simple TCP fairness. Construct a double star topology with senders on the left and receivers on the right sharing a bottleneck link. For each of the bottleneck bandwidths suggested, initiate two TCP flows and ten seconds later, initiate a multicast flow to two receivers. Expected behavior: An equilibrium should quickly be reached in which the multicast flow receives approximately one-third of the available bandwidth.

6. TCP fairness with high multiplexing. Same topology and general set-up as in the previous experiment, but scale up the number of TCP flows as the bottleneck bandwidth increases, and verify that the multicast session still receives (only) its fair share.

7. Inter-session fairness. Same topology and general set-up as in the previous two experiments, but 1) use multicast flows exclusively and 2) scale up the number of multicast flows and TCP flows concurrently.

[1]Note that the link bandwidths should be set to be large enough to avoid queuing effects, i.e. 100Mbps

8. Heterogeneous fairness. Begin with the double star topology used in the simple TCP fairness experiment. Now make the round-trip times that flows experience heterogeneous by varying the latencies on the last hops to the receivers. With two TCP flows and one multicast flow to two receivers, allow one of each type of receiver to have a short last hop (10ms) and one to have a long last hop (100ms). Expected behavior: For single-rate congestion control, the multicast throughput should be comparable to the mean throughput achieved by the TCP flow with the long last hop. For multi-rate congestion control, the two short receivers and the two long receivers should ideally have comparable mean throughput values.

9. Augmented sanity check (scalability). Build a binary tree with receivers at the leaves, scaling the set of receivers up to hundreds of receivers. Place the source one hop away from the root of this tree on a 1Mbps link. Now initiate a multicast session to these receivers. After 30 seconds, inject 500Kbps CBR cross traffic on the shared link adjoining the source. After 60 seconds, reduce this cross traffic to 250Kbps. Expected behavior: Ideally the mean throughput seen by the receivers should be the same as in the basic sanity check. Be sure to measure the difference in responsiveness from the basic sanity check, and demonstrate the scalability of the reverse path traffic (ACKs, NAKs, etc.).

10. Throughput-loss curve. With a single sender and receiver setup as in the basic sanity check, set the bottleneck bandwidth to be large (100 Mbps). Then vary the random loss rate and the round-trip time on the bottleneck link and plot the throughput-loss curve of the congestion control protocol.

11. Bandwidth absorption. Never fully specified. We had a half-baked idea of using the basic sanity check set-up and measuring the mean throughput achieved by a multicast session vs. an unspecified deterministic VBR flow. The main point of this experiment would be to demonstrate reasonable responsiveness to changing network circumstances.

12. Correct loss aggregation. Build a tree with a sequence of lossy links falling from 5% to 1% at 1% intervals. Place a single client downstream of each lossy link (five in all). Expected behavior: The worst-case client experiences 15% packet loss and should be accorded an appropriate throughput.

13. Drop-to-zero. Build a star topology for a multicast session with 50 receivers, each experiencing 5% packet loss over uncorrelated links. In a second experiment, let the 50th receiver experience 20% loss rather than 5%. Compare the throughput to the throughput that a TCP flow experiencing identical conditions would receive. Expected behavior: The session rate should not drop to zero in either case; in the second experiment, single-rate congestion control schemes should reduce their rate to compensate for the lossy receiver.

14. Anticorrelated loss. Build a star topology for a multicast session with 20 receivers. At any instant in time, 19 of the receivers will experience 1% packet loss, while the twentieth receiver will experience 5% packet loss. The location of the lossy receiver *rotates* every k seconds, where k is either 0.5, 5, or 15. This particular protocol must be tested with accurate modeling of IGMP leave latency and throughput should again be compared with TCP flows experiencing identical conditions. Expected behavior: With single-rate congestion control, the session rate must cater to the lossy receiver; with multi-rate, the experiment tests tolerance to IGMP leave latency.

15. Multiple bottlenecks. Build a binary tree with 8 leaves and label the links of the tree with their loss rates. The four lossy links include the two links adjoining the root, with loss rate 4% and 1% respectively, a link with 2% loss immediately below the link with 4% loss, and a link with 9% adjoining a client under the 1% subtree. Initiate a multicast session with receivers located at all eight leaves, and run four TCP sessions from the source to a client in each of the four distinct loss regions (totaling 4%,

6%, 1%, and 10%). Expected behavior for multi-rate sessions: comparable mean throughput at each receiver for their multicast session and their TCP flow.

16. PIM/IGMP issues. We suggested a topology which is beyond the scope of a textual writeup. The main point of this complex scenario was to verify TCP-friendliness even when issues of IGMP and PIM latency are apparent. The topology we considered included three nasty beasts in succession: a LAN which PIM messages needed to traverse (3 seconds), a high latency link (100ms) and a last-hop router which processed IGMP messages (3-9 seconds). We advocated putting a multicast receiver behind each of these obstacles, then running a multicast session to these receivers alongside individual TCP flows. It is of course necessary to perform hop-by-hop modeling of control messages to run this simulation. Expected behavior: good luck.

3 Outcomes and Topics We Didn't Cover

In most cases, the experiments we have enumerated here have been designed with a primary goal of attaining an appropriate mean throughput value over reasonable time scales at each receiver. We often implicitly make the assumption that compatibility with a TCP flow operating in the same environment is equivalent to a successful outcome. However, we emphasize that we do not intend for this objective to be attained by incurring performance penalties which were not explicitly mentioned in the writeup. When evaluating a multicast congestion control protocol, it is essential to measure its other impacts, such as the amount of traffic it generates on the reverse path, or the amount of control traffic it generates. We also recommend quantifying the protocol's responsiveness, especially the time it takes to react to congestion.

While this list of experiments is incomplete and should be viewed as a work in progress, there are some omissions which are intentional. We did not consider simulations which cover failure cases, including link failure, receiver failure and aggregation node failure. We also did not consider simulations which include changes in the topology during simulation. Finally, we did not consider simulations which studied rich network topologies. Any successful multicast congestion control protocols will be expected to address all of these cases.

References

[1] M. Handley. Reference simulations for multicast congestion control, 1999. Draft presented at Reliable Multicast Research Group meetings in 1998-1999.