TFRC with Self-Limiting Sources

Tom Phelan
Sonus Networks
tphelan@sonusnet.com
October 2004

(Addendum added 20-Oct-2004)

Table of Contents

1 INTRODUCTION 3

2 SIMULATOR 3

2.1 SIMULATION ENVIRONMENT 3
2.2 NETWORK ENVIRONMENT 3

3 CLIENT-SIDE SIMULATIONS 4

3.1 INBOUND CLIENT-SIDE 4
3.1.1 Simulation 1 4
3.1.2 Simulation 2 5
3.1.3 Simulation 3 7
3.2 OUTBOUND CLIENT SIDE 8
3.2.1 Simulation 4 8

4 SERVER SIDE SIMULATIONS 9

4.1 SIMULATION 5 9
4.2 SIMULATION 6 11

5 CBR SOURCES 14

5.1 SIMULATION 7 14

6 CONCLUSIONS 15

7 CONTACT INFORMATION 15

8 ADDENDUM – THE EFFECTS OF PACKET SIZE 15

Table of Figures

Figure 1: Network topology 4
Figure 2: Sim1 results. 5
Figure 3: Sim2 results. 6
Figure 4:Sim2, TFRC flow only. 6
Figure 5: Sim3 results. 7
Figure 6: Sim3, TFRC flow only. 8
Figure 7: Sim4 results. 9
Figure 8: Simulation 5 results. 10
Figure 9: Simulation 5 – representative TCP flows. 10
Figure 10: Simulation 5 – representative TFRC flows. 11
Figure 11: Topology for simulation 6. 12
Figure 12: Simulation 6 results. 12
Figure 13: Simulation 6 – representative TCP flows. 13
Figure 14: Simulation 6 – representative TFRC flows. 13
Figure 15:Simulation 7 results. 14
Figure 16: Sim7, CBR only. 15
Figure 17: Sim8, Sim3 redone with 100ms packetization (TFRC stream only). 16

1 Introduction

TCP-Friendly Rate Control (TFRC) is a congestion control algorithm that is meant as an alternative to the widely deployed Additive-Increase Multiplicative-Decrease (AIMD) algorithm used in TCP. The basic characteristic of TFRC is that it gradually diminishes its maximum allowed rate in response to congestion events, as opposed to the halving of the rate used by AIMD. In order to not overwhelm competing AIMD flows, the price for this smooth decrease is a less aggressive increase when more capacity is available. TFRC implementations are currently being worked on for the Datagram Congestion Control Protocol (DCCP) and as a profile for the Real Time Protocol (RTP).

Applications that value continuity of transmission/delivery rates over maximum throughput should find TFRC useful. The application that immediately jumps to mind when these characteristics are discussed is streaming media, and that has been presented as one of the driving forces for TFRC. Interestingly, even though streaming media has always been assumed to be a major candidate for the use of TFRC, no simulation studies that I know of have used TFRC sources that resemble streaming media sources. Instead, the simulations have focused on sources that resemble file transfer operations. These simulations have shown a reasonable long-term "fairness" between TFRC and TCP flows competing for network resources.

There is a marked difference between file transfer applications and streaming media, though, and the difference is greediness. There is a finite throughput that is enough for a streaming media application. Even if more throughput is available, the streaming media application will not use it. File transfer applications, on the other hand, are not satisfied by any finite throughput. In the real world, network resources are not always the limiting factor (disk transfer rate sometimes gets in the way), but, for the most part, file transfer applications will consume all available network resources.

The main question for this paper is: Can a TFRC source that self-limits to a throughput less than its "fair share" maintain that throughput in the face of competing TCP applications? Examination of the TFRC algorithm is inconclusive on this question. Some factors indicate "yes", others "no", and the balance is difficult to determine. To further the discussion, this paper presents the results of a simulation study that uses self-limiting TFRC sources competing with TCP-based file transfer applications.

2 Simulator

2.1 Simulation Environment

The study uses the "ns" network simulator, version 2.27, available from http://www.isi.edu/nsnam/ns/. It uses the simulator's TCP, TCPSink, TFRC and TFRCSink "Agents" and FTP and CBR "Applications".

Some minor modifications were made to the TFRC and TFRCSink agents. The original TFRC agent maintains a virtually infinite queue of pending transmissions. In some circumstances, this resulted in delivery delays exceeding one or two seconds. Since this is typically unacceptable to streaming media applications, the ability to limit the transmit queue length was added. For the simulations, the maximum transmit queue length was two packets. Experimentation has indicated that this does not have a qualitative effect on the simulation results.

The TFRCSink agent was modified to record a received bytes count. This count was used to create the throughput rates shown for the simulations. Note that all of the throughput rates are bits received at the endpoint.

The ns source modifications, simulation scripts, and raw simulation results are available at http://www.phelan-4.com/dccp/tfrc-self-limit.tgz.

2.2 Network Environment

The simulations use the five-node network shown in Figure 1. Various link speeds are used, depending on the simulation scenario, but the link between N2 and N3 always represented the scarce resource. TCP flows are from N0 to N4 and TFRC flows are from N1 to N5.

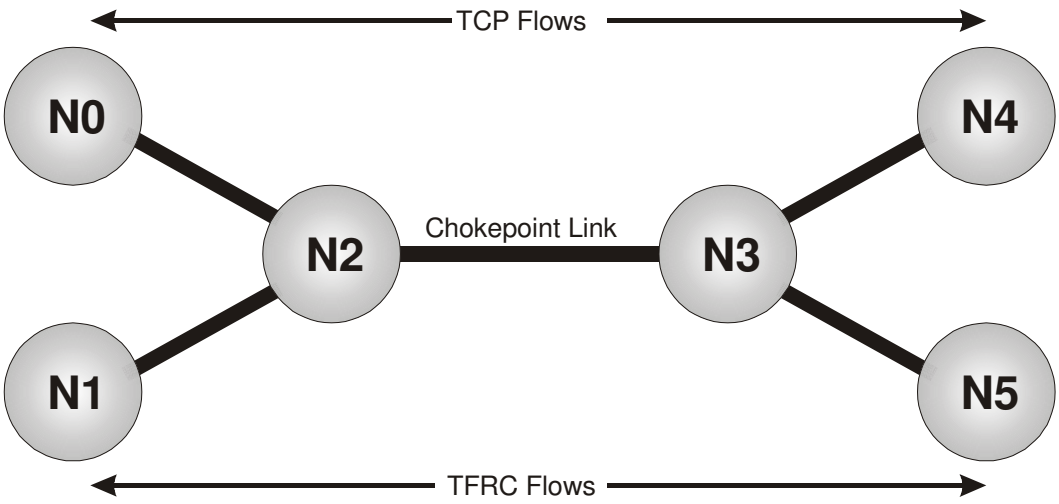

Figure 1: Network topology.

Three simulation scenarios are used. Two scenarios use low degrees of connection multiplexing and roughly emulate client devices on an asymmetric broadband connection. The third scenario uses a high degree of connection multiplexing and roughly emulates a server in a server farm with shared high-speed access.

The TFRC source emulates one leg of a voice call using G.711 encoding with 10 ms packetization intervals. G.711 takes an 8-bit sample of the voice signal once every 125 µs (8,000 times per second, 64K bps). The encoder collects 10 ms worth of samples into a packet payload (80 bytes) and adds 40 bytes of IP/UDP/RTP header to make a 120-byte packet. Ignoring MAC headers, that makes a 96K bps stream. The simulation uses the CBR application attached to the TFRC agent to send one 120-byte packet every 10 ms. The TFRC agent sends the packet if the current allowed rate allows it, queues it if the current transmit queue length is less than two packets, and otherwise discards it.

3 Client-Side Simulations

3.1 Inbound Client-Side

In the Inbound Client Side scenario, the access links are 10M bps (N0 and N1 to N2 and N3 to N4 and N5) and the chokepoint link is 1.5M bps (N2 to N3).

3.1.1 Simulation 1

In the first simulation run, there is one TFRC stream that starts at 1.0 seconds and one TCP stream that starts at 2.0 seconds and ends at 19.0 seconds.

Figure 2 shows the throughput for each flow. The TFRC flow takes a hit in throughput when the TCP flow starts up, but returns to a fairly steady 96K bps about a second-and-a-half later. These results are fairly encouraging – the TFRC flow limits itself to substantially less than its nominal 750K bps "fair share" and is able to hold onto it. Unfortunately, this is the last encouraging result we will see.

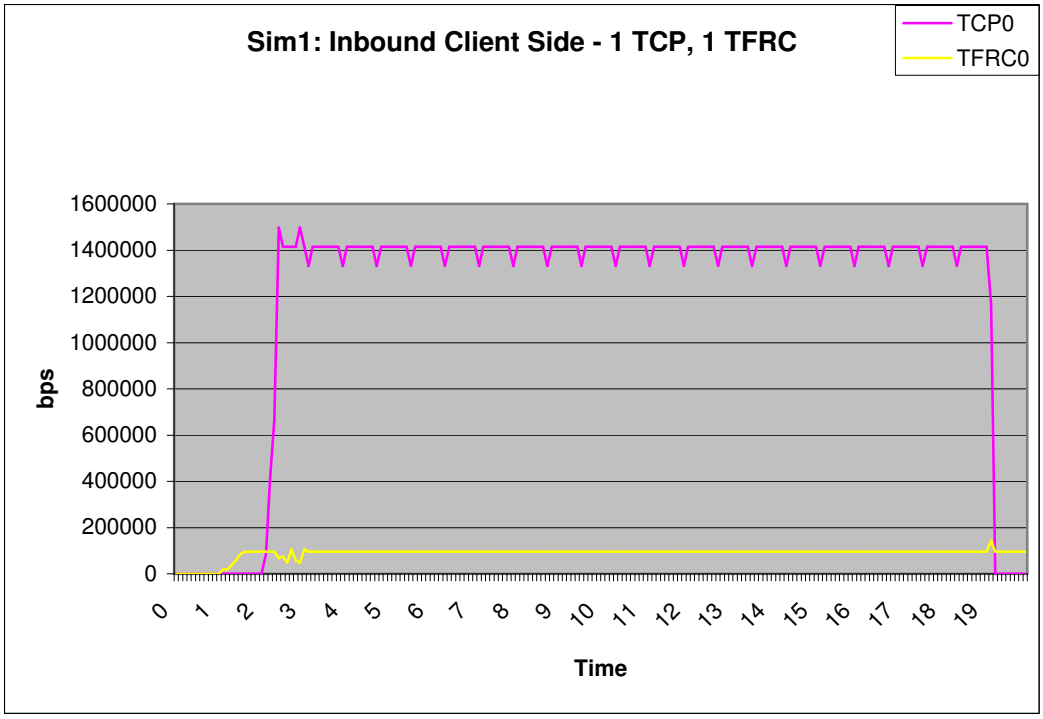

Figure 2: Sim1 results.

3.1.2 Simulation 2

The second simulation roughly emulates a high-graphic-content web page download concurrent with the voice stream by using four TCP flows, starting at 2.0 seconds with 0.2 seconds separation. Figure 3 shows the aggregate throughput for the TCP flows and the throughput for the TFRC flow. The magnitude of the TCP throughput makes it difficult to see the TFRC flow, so the TFRC throughput is shown by itself in Figure 4.

Even though the 96K bps required by the TFRC flow is still well below its nominal 300K bps "fair share" (1.5M bps evenly divided among the five competing flows), the TFRC flow does not fare well. Between seconds 2 and 4 of the simulation, the TFRC flow is driven down to roughly 25K bps, where it remains until about second 6. After that it gradually rises over the next 3 or so seconds to about 75K bps, where it more or less remains until the TCP flows stop in second 19.

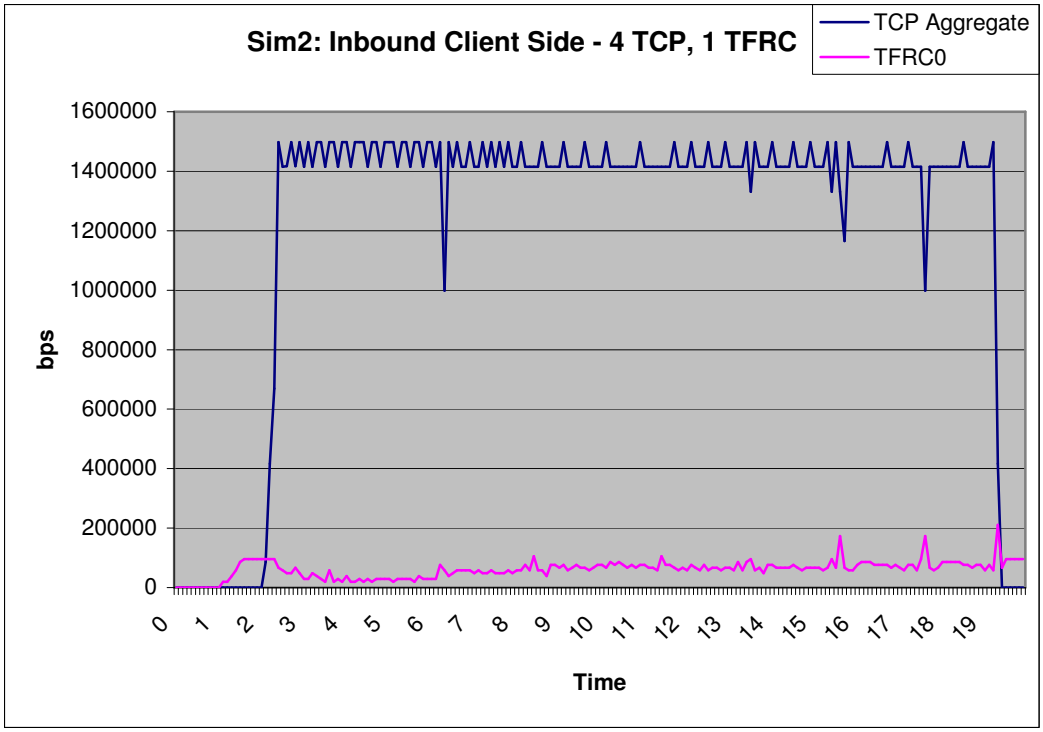

Figure 3: Sim2 results.

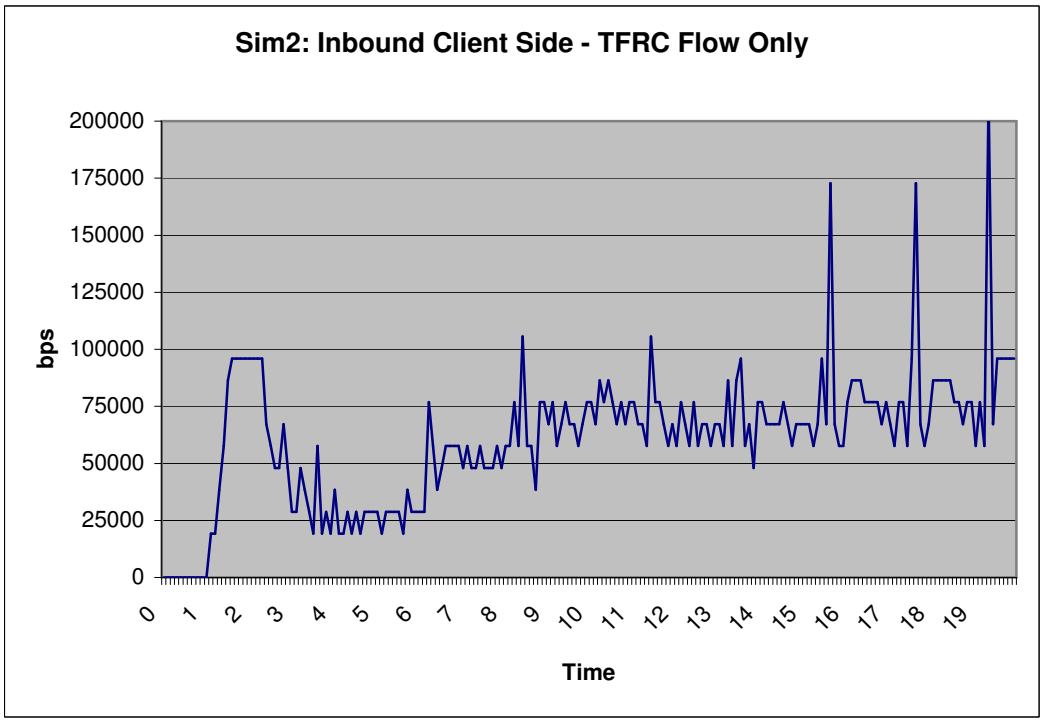

Figure 4:Sim2, TFRC flow only.

3.1.3 Simulation 3

The third simulation adds two more TCP flows, for a total of six, plus the one TFRC flow. Figure 5 shows the aggregate TCP throughput and the TFRC throughput. Figure 6 shows the TFRC throughput only.

Again, even though the TFRC's 96K bps second is less than its fair share (now a little over 200K bps), it is unable to maintain its throughput, and is quickly driven down below 20K bps. This time, however it's unable to rise from there until the TCP flows stop at 19 seconds.

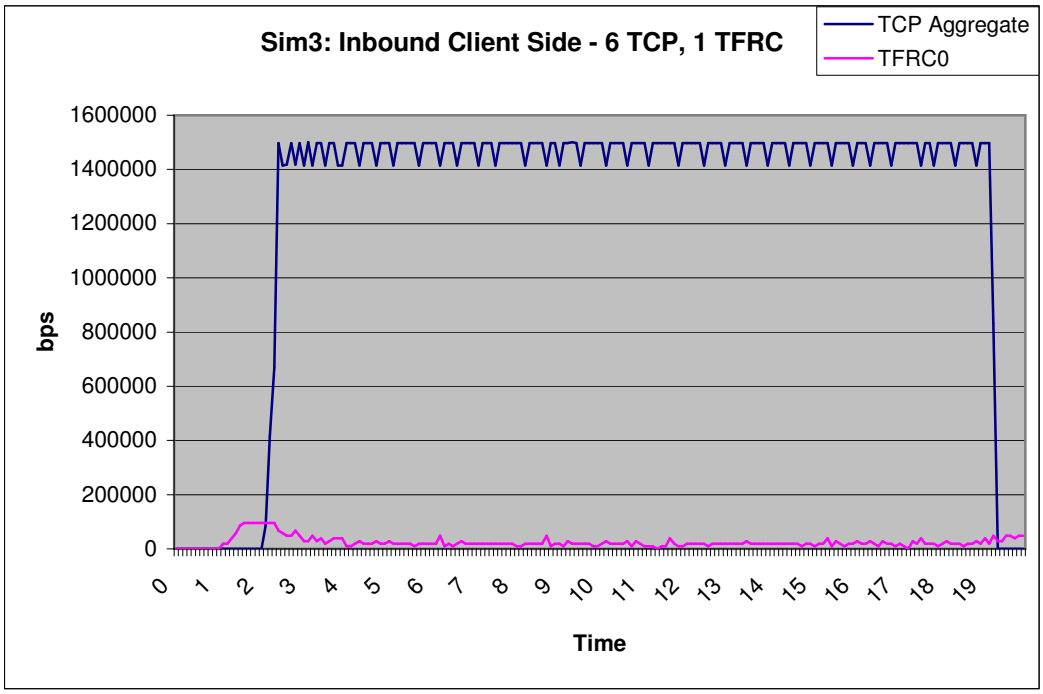

Figure 5: Sim3 results.

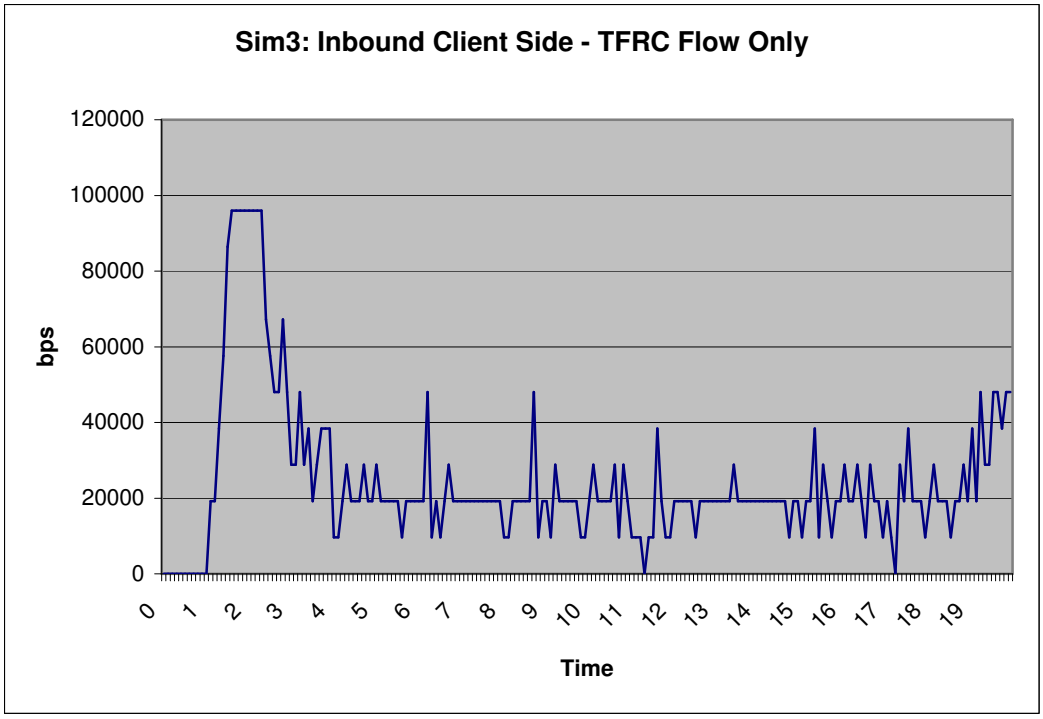

Figure 6: Sim3, TFRC flow only.

3.2 Outbound Client Side

In the Outbound Client Side scenario, the access links are still 10M bps, but the chokepoint link is 256K bps.

3.2.1 Simulation 4

This simulation uses one TCP flow and one TFRC flow. The results are shown in Figure 7. Even though the TFRC flow's 96K bps requirement is less than its nominal fair share (128K bps), it is quickly driven down to less than 10K bps by the TCP flow. Eventually it does manage to recover to around 30K bps, but it is unable to rise from there until the TCP flow stops.

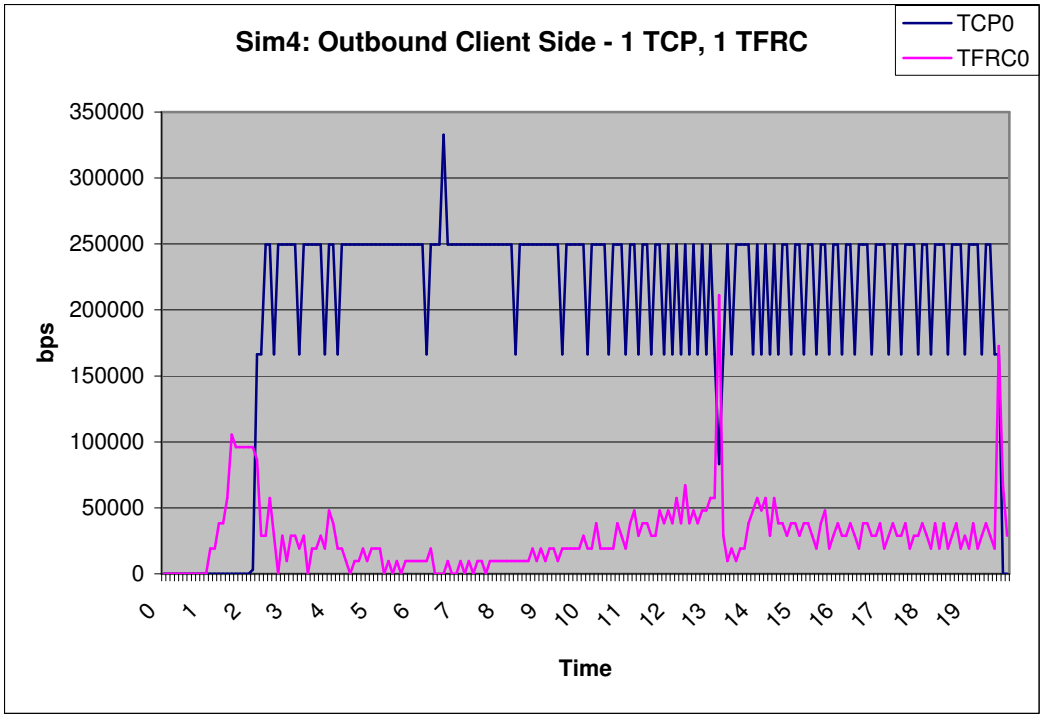

Figure 7: Sim4 results.

4 Server Side Simulations

4.1 Simulation 5

The server side scenario emulates a server in a server farm with a high-speed access connection. For simulation 5, the access links are 100M bps and the chokepoint link is 45M bps. There are 80 TFRC flows and 120 TCP flows giving a fair share throughput of 225K bps. The first TFRC flow begins at second 1 and another is added each 299 µs after that (the non-round number is to avoid synchronizing the sources). The first TCP flow begins at second 2 and others are added each 50 ms after.

Figure 8 shows the aggregate results. At about second 3 of the simulation, when 20 or so TCP flows are active, the TFRC flows begin to be squeezed out. The TFRC aggregate throughput declines inexorably throughout the simulation (as the TCP throughput climbs) from the initial 7.68M bps to less than 2M bps with a momentary low of 1.6M bps in second 17.

Figure 9 shows the throughput for three representative TCP flows. Their throughput varies quite a bit, but seems to average well over the 96K bps the TFRC flows would like to get. TCP60 does seem to take some pretty large throughput hits though. I suspect this has nothing to do with the competing TFRC flows.

Figure 10 shows three representative TFRC flows. All are driven significantly below their desired 96K bps, even though their theoretical fair share is 225K bps.

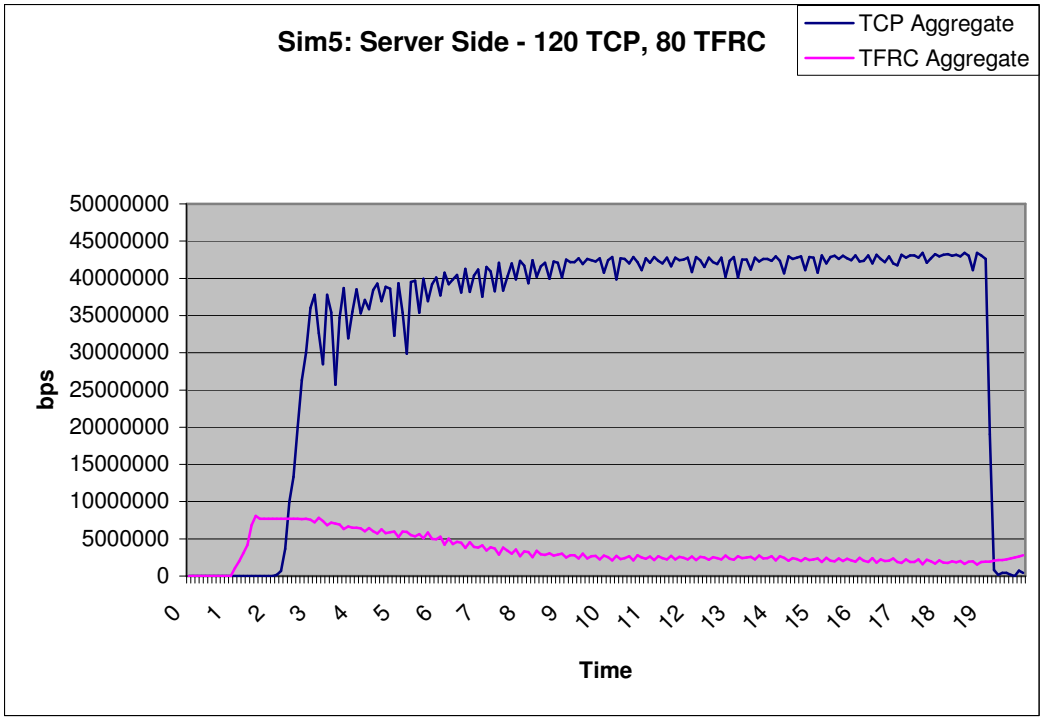

Figure 8: Simulation 5 results.

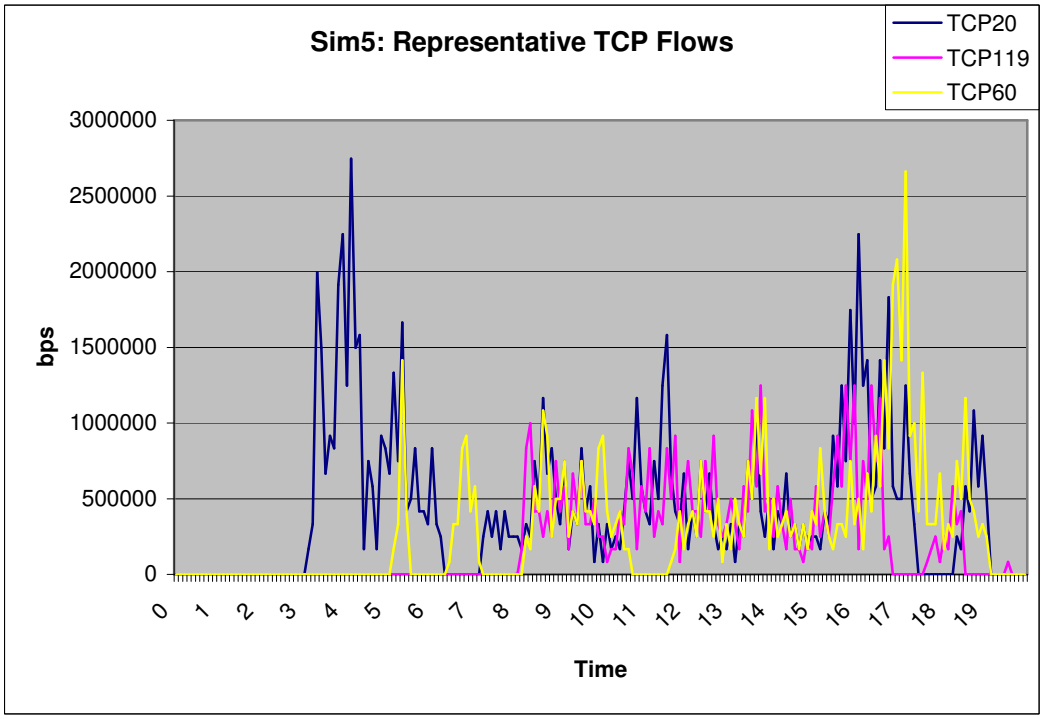

Figure 9: Simulation 5 – representative TCP flows.

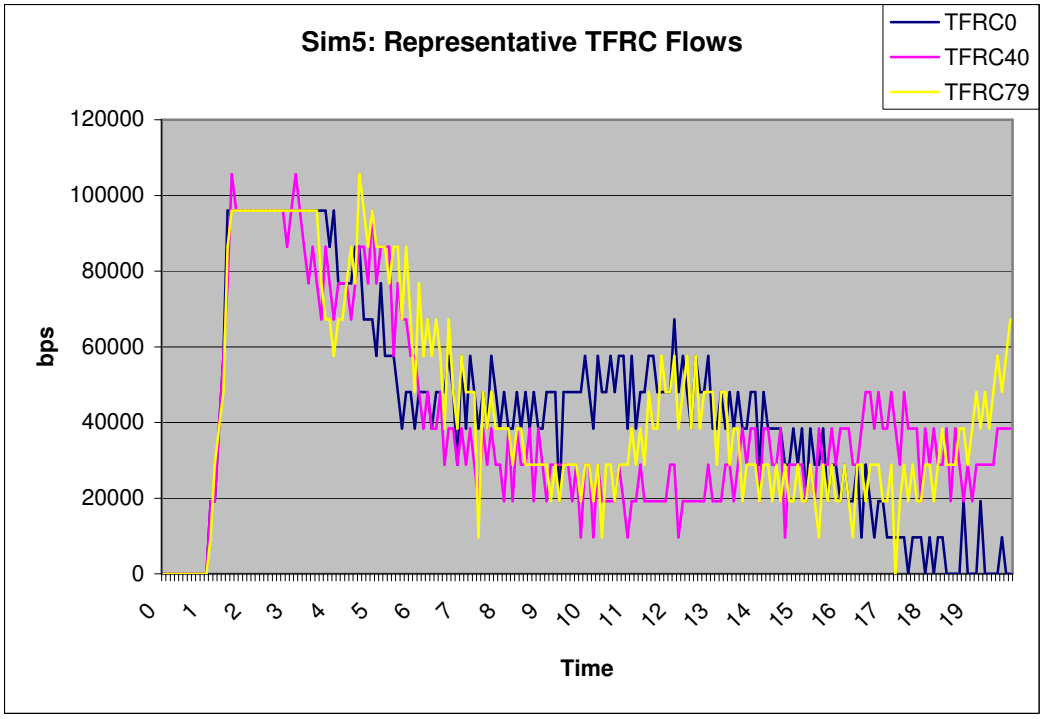

Figure 10: Simulation 5 – representative TFRC flows.

4.2 Simulation 6

The topology of simulation 5 is perhaps not a reasonable approximation of the normal client-server model – after all, the simulated client devices sit on a 100M bps access link. Simulation 6 modifies the topology so that each TCP flow terminates on a separate node with its own 1.5M bps access link (see Figure 11). The number of TFRC and TCP flows and the start and ramp rates are the same as for simulation 5 (80 TFRC, 120 TCP).

Figure 12 shows the aggregate results for simulation 6. Although the rate of the decline in the TFRC throughput is less that in simulation 5, it is still pronounced, dropping from its initial 7.68M bps to about 3M bps, with a low of about 2.9M bps in second 17. Figure 13 and Figure 14 show throughput for representative TCP and TFRC flows.

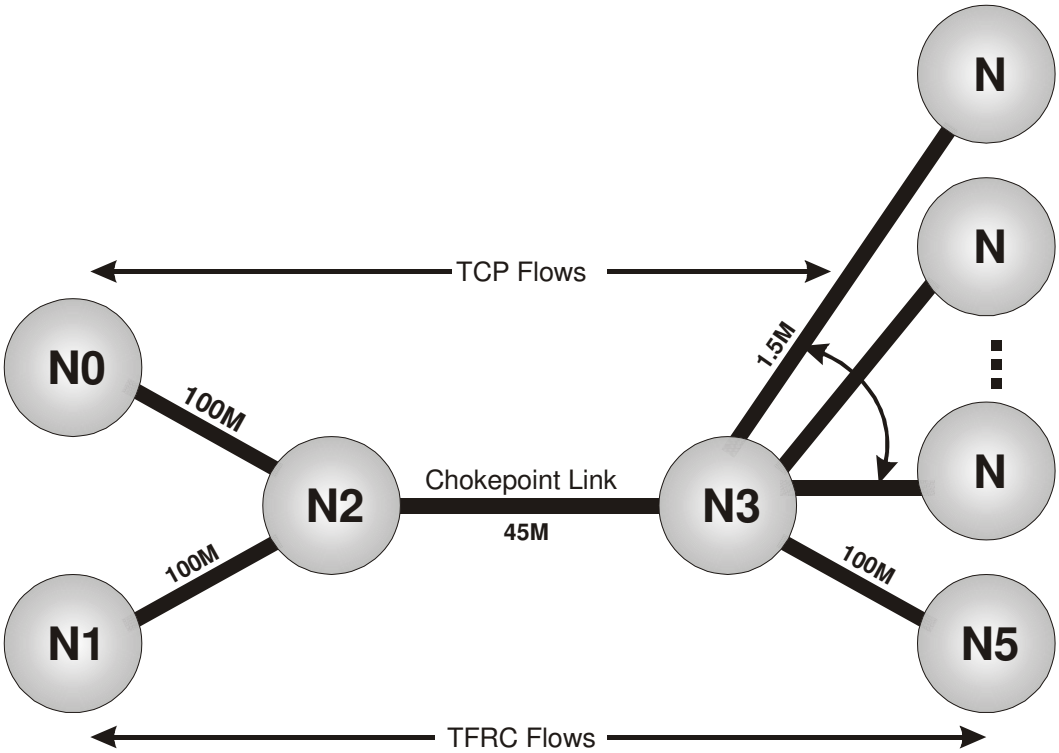

Figure 11: Topology for simulation 6.

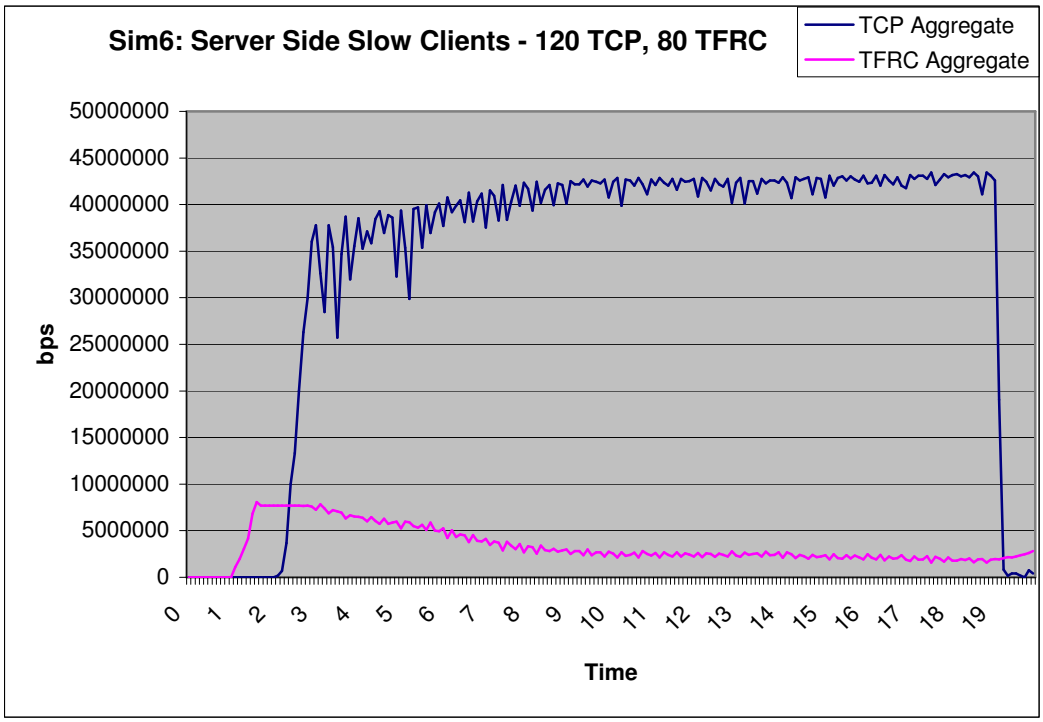

Figure 12: Simulation 6 results.

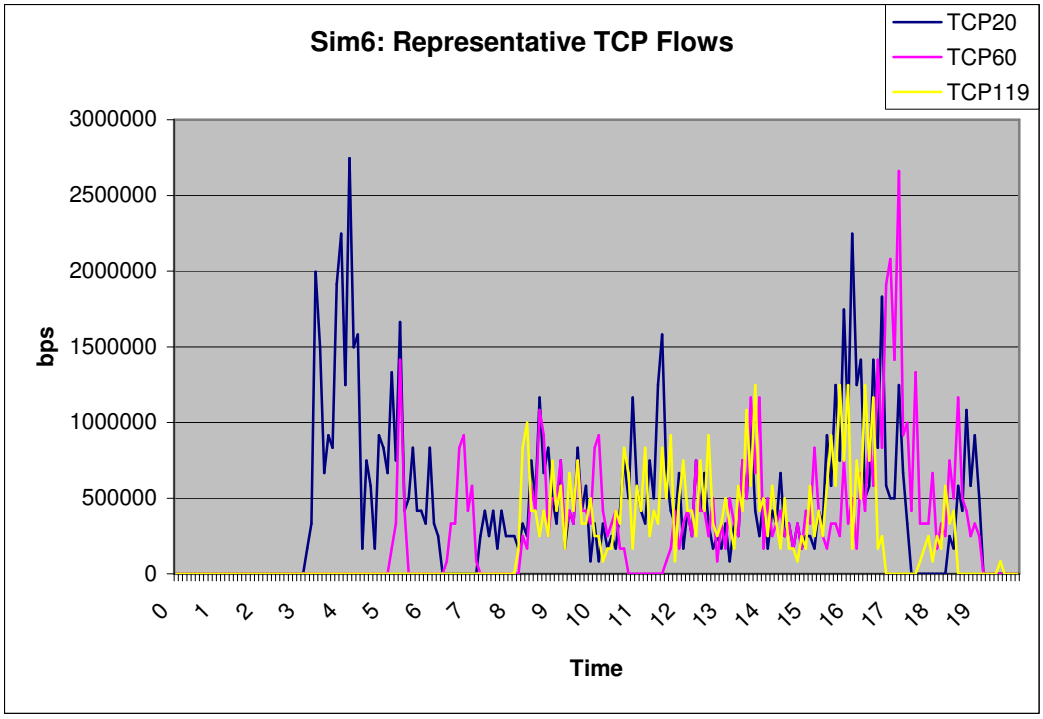

Figure 13: Simulation 6 – representative TCP flows.

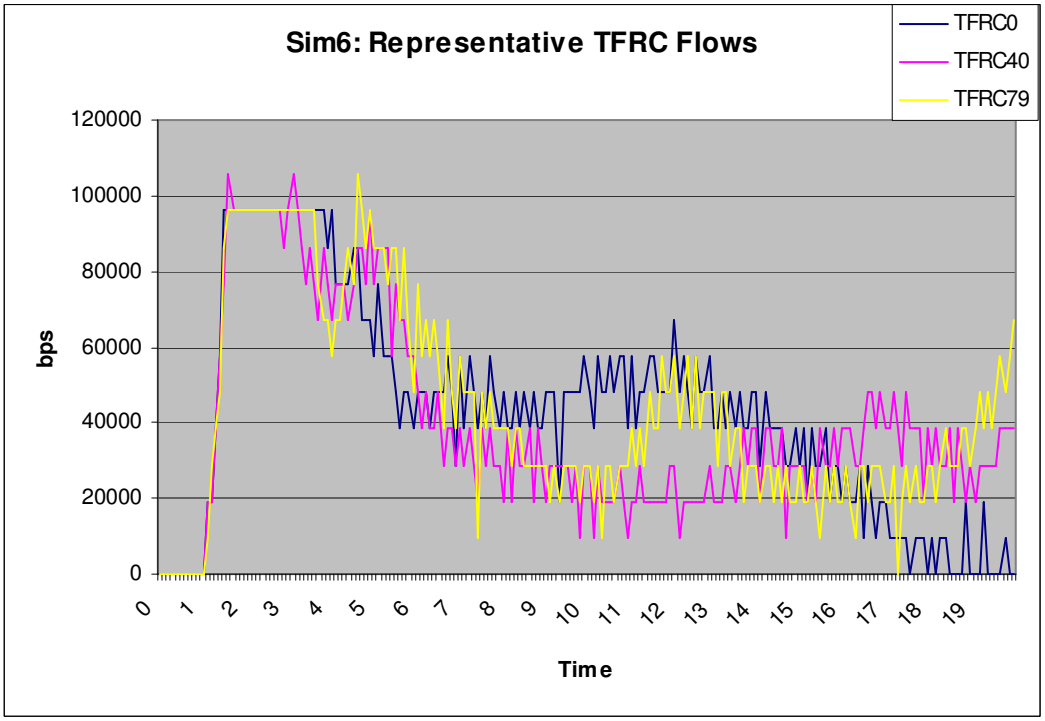

Figure 14: Simulation 6 – representative TFRC flows.

5 CBR Sources

In the simulations above, the CBR/TFRC packets are discarded both by the network and at the transmitter when the TFRC rate falls below the CBR rate. It should be interesting to see the results of a traditional CBR application that does not adjust its rate in response to network congestion.

5.1 Simulation 7

The seventh simulation uses the same scenario as simulation 3 (six TCP sources, one CBR source, 1.5M bps chokepoint link), but replaces the CBR/TFRC source with a CBR/UDP source. Figure 15 shows the simulation results and Figure 16 shows the CBR throughput only.

While the CBR flow doesn't completely achieve its desired 96K bps, it does come close, at least compared to the results the TFRC flows achieved above. From second 1 to 19 it averages 92K bps. By comparison, the TFRC flow in simulation 3 averaged 25K bps over the same period.

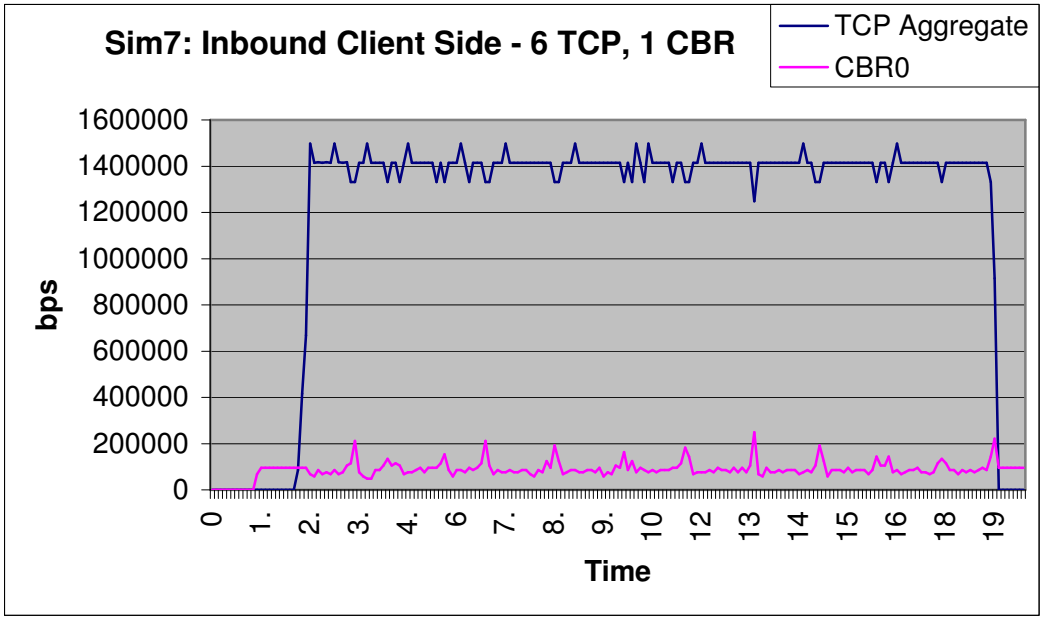

Figure 15:Simulation 7 results.

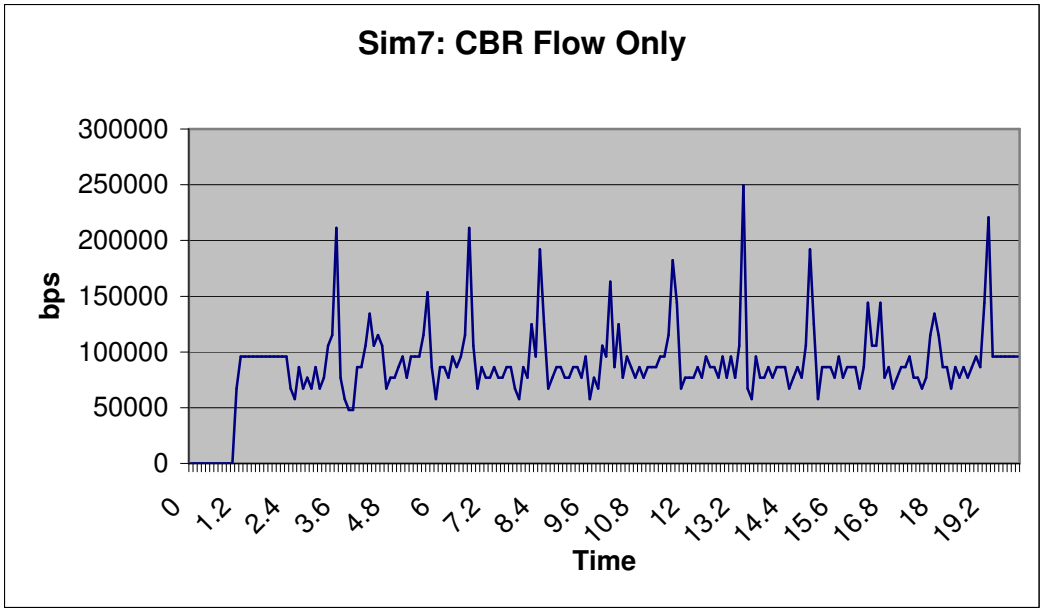

Figure 16: Sim7, CBR only.

6 Conclusions

Assuming my methods are reasonable (and the TCP and TFRC implementations in ns work correctly), the simulations paint a pretty grim picture about the usefulness of TFRC for self-limiting sources. A source that is limiting itself to less than its theoretical "fair share" will, in many common situations, likely be driven substantially below its desired rate.

The results are far more fair when the self-limiting source uses traditional UDP methods and doesn't adjust to packet losses. Of course this assumes that the self-limit is less than the fair share. When it is more that the fair share, it's the TCP flows that will likely suffer. How to tell the difference between these two situations will take some research.

7 Contact Information

Tom Phelan
Principal Architect
Sonus Networks
250 Apollo Dr.
Chelmsford, MA USA 01824
+1-978-614-8456
tphelan@sonusnet.com

8 Addendum – The Effects of Packet Size

In discussions on the DCCP working group mailing list subsequent to the initial release of this paper, it was pointed out that since TFRC works on packet rate, not byte rate, it is converging on a fair share in packets per second. Sure enough, if you look at the data in that light (1000-byte TCP packets and 120-byte TFRC packets), you'll see that the TFRC sources roughly get a fair share packet rate (with things seeming perhaps a little "fairer" to TFRC).

This is refreshing, in that it seems that the "fault" was in my expectations of TFRC, not in its design. It's possible that a self-limiting source that uses large packets would get a fair share of bits per second as well as

packets per second. Unfortunately, for the throughput rates involved in telephony applications, this doesn't seem to hold up. Forgetting the problems of end-to-end delay for two-way telephony applications, I tried the simulation 3 case (6 TCP streams and one TFRC stream) using a 100ms packetization interval for the TFRC stream. This gives 840 byte packets, and a resulting bit rate of 67.2K bps (the lower ratio of packet header overhead to payload reduces the overall bit rate).

Figure 17 shows the results for the TFRC stream. The stream achieves an overall throughput (from second 1 to 19) of about 31K bps, far below the nominal 67.2K. It's difficult to tell just which factors predominate here – in addition to possible TFRC rate limiting, each dropped packet represents 10% of the flow – but whatever the causes, the results aren't good.

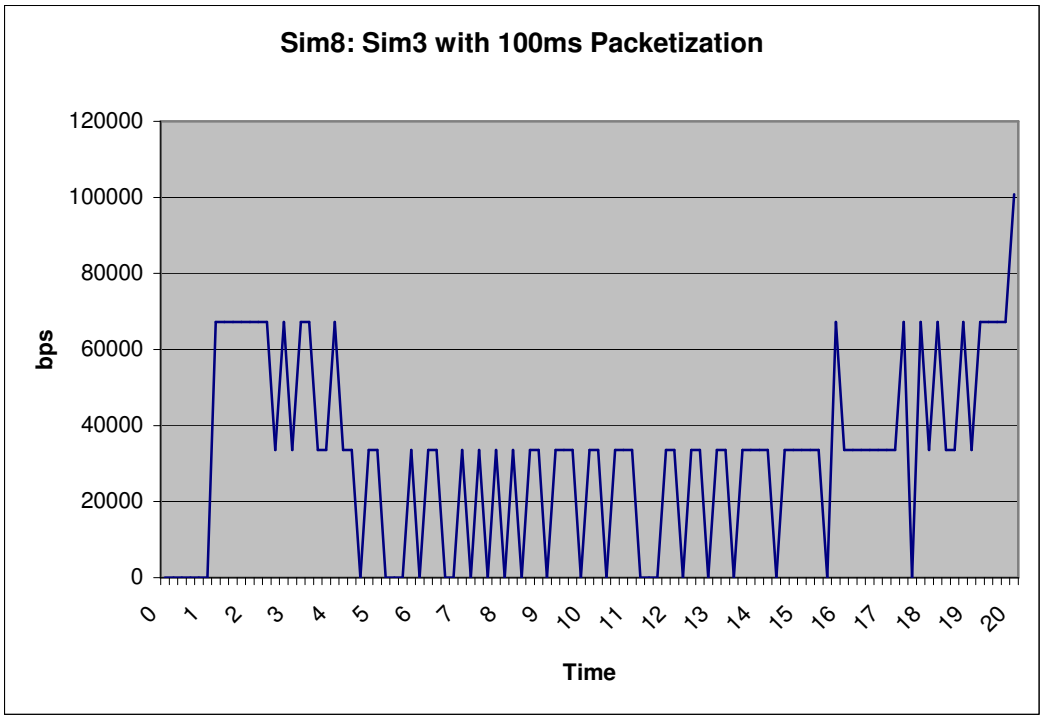

Figure 17: Sim8, Sim3 redone with 100ms packetization (TFRC stream only).

But regardless of how this big packet test worked out, big packets are just not possible for two-way telephony applications. The resultant additions to end-to-end delay make conversation too difficult for general use. While it is good to better understand the nature of TFRC, that doesn't change the conclusions of this study.

I'd like to thank Dado Colussi for his thoughtful analysis of what's going on within TFRC during the simulations, and for pointing out the packet rate versus byte rate aspects of TFRC (http://www1.ietf.org/mail-archive/web/dccp/current/msg01147.html). I'd also like to thank Philippe Gentric for his common-sense definition of fairness from a media app's point of view (http://www1.ietf.org/mail-archive/web/dccp/current/msg01149.html).