Modifying TCP's Congestion Control for High Speeds

Sally Floyd, Sylvia Ratnasamy, Scott Shenker (in some order)
VERY ROUGH PRELIMINARY DRAFT!!!

May 5, 2002

1 Introduction

We are working on a modification to TCP's window increase/decrease algorithm that would allow TCP to run with high congestion windows with realistic packet drop rates. TCP's sending rate is roughly $1.2/\sqrt{p}$ packets per round-trip time, for p the packet loss rate on the path. This is a direct consequence of TCP's halving its congestion window in response to loss, and increasing the congestion window by one packet per round-trip time otherwise. TCP's response function, with its average congestion window of $W = \sqrt{1.5}/\sqrt{p}$ packets, places an upper bound on achievable congestion windows, given some underlying packet drop rate p_1 from corruption and/or congestion. For example, if the packet corruption rate p_1 is 10^{-2k}, this limits the average congestion window to $\sqrt{1.5} * 10^k$. (Note that, for 1500-byte packets, a *packet* corruption rate of 10^{-2k} results from a *bit* corruption rate of $1.5 * 10^{-2k-3}$)

For example, for a TCP connection with 1500-byte packets and a 100 ms round-trip time, filling a 10 Gbps pipe would require a congestion window of $W = 83,333$ packets, and a packet drop rate of at most one drop every $N = 5,000,000,000$ packets (because $N = W^2/1.5$). This is at most one drop per $S = 6000$ seconds (because $S = N/(10W) = W/15$). This is past the limits of achieveable fiber error rates. In contrast, to fill a 100Mb pipe, the loss rate must not exceed 1 in 500,000 packets, or equivalently, one per $S = 60$ seconds (because $S = N/(10W) = W/15$).

It is easy to design congestion control schemes that achieve higher sending rates at a given loss rate. However, the challenge is to do so while retaining the TCP-compatibility (or TCP-friendliness) properties of the congestion control algorithm; any new congestion control algorithm will have to coexist with existing TCP implementations, and the challenge is to enable the modified congestion control algorithms to achieve high speed while, at the same time, not unfairly stealing bandwidth from unmodified TCPs. Some might argue that fairness does not matter, and that future networks will use QoS mechanisms and per-flow scheduling (or other forms of router-enforced fairness). In contrast, we expect that, while QoS mechanisms and a range of scheduling mechanisms will likely have an important place in future networks, best-effort traffic and FIFO scheduling will continue to have their place also, due to their fundamental simplicity and "good enough" performance.

To allow TCP to realistically operate at very high congestion windows, we are exploring a modification to TCP so that, once a moderately high congestion window was achieved, TCP's additive-increase and multiplicative-decrease parameters would be a function of the current window size. This would allow TCP to achieve congestion windows of tens of thousands of packets with realistic packet drop rates (e.g., one in 100,0000 packets dropped), with minimal danger to the rest of the network. This would increase the dynamic range of a single TCP connection, while maintaining strict fairness with current TCP implementations in the heavy and moderate packet drop ranges more typical of the current Internet. Currently, users who require very high bandwidth open up multiple TCP connections, or use MulTCP [GRK99], which behaves roughly like an aggregate of N virtual TCP connections. We believe that our approach offers more flexibility, competes more fairly, and will more effectively scale to a wide range of available bandwidths.

We note that our research is not about the specifics of TCP itself. That is, we are not exploring TCP-specific issues such as the number of bits needed in the header for receiver window advertisements, but are exploring general congestion control issues applicable to other transport protocols as well.

Moving to extremely high speeds poses special challenges to the Active Queue Management (AQM) algorithms in routers. We don't propose requiring changes to routers in order for our above proposals to work, but we do plan to investigate the interactions between highspeed TCP and the AQM mechanisms at the router.

The above modification would enable flows to eventually reach higher steady-state speeds than are attainable today, but this modification does not address the problem of reaching these steady-state speeds quickly. More aggressive sending, allowing best-effort flows to start with higher initial windows, for example, or to increase their sending rate more rapidly, would require more feedback from the routers along the path. We are restricting out attention to what is attainable given the current feedback from routers along the path.

2 What's the Problem?

2.1 Basics

TCP uses the following algorithm to adjust its congestion window w:

$$\mathrm{ACK} : w \leftarrow w + \frac{a}{w},$$

$$\mathrm{Drop} : w \leftarrow w - bw,$$

and

$$\mathrm{Slow-StartACK} : w \leftarrow w + c,$$

where w, a, and c are all defined in units of the Maximum Segment Size (MSS). TCP uses increase parameter $a = 1$, decrease parameter $b = 0.5$, and slow-start parameter $c = 1$. [Jac88].

2.2 Steady-State

At steady-state, TCP's sending rate T, measured in max-sized packets per round-trip time (ppr), is given by [FHP00]:

$$T = \frac{\sqrt{\frac{a(2-b)}{2b}}}{\sqrt{p}} \text{ ppr}.$$

Using the canonical parameters, this gives the following TCP response function [FF99]:

$$T = \frac{\sqrt{1.5}}{\sqrt{p}} \text{ ppr},$$

where p is the per-packet drop rate, for a TCP sending an acknowledgement packet for every data packet.

Assuming 1500-byte packets and a round-trip time $R =$ 100 msec, then a sending rate of T ppr is equivalent to $T \times 1.2 \times 10^5$ bps. For a TCP connection to maintain a sending rate of 10 Gbps in this case, for example, TCP would require

$$\frac{\sqrt{1.5}}{\sqrt{p}} \times 1.2 \times 10^5 = 10^{10},$$

or $p \approx 2 \times 10^{-10}$. Note that p represents the fraction of *packets* dropped or corrupted, and not simply the *bit* corruption rate. This is probably past the limits of achievable fiber error rates. For example, a packet drop/corruption rate p of 2×10^{-10} corresponds to one in every 0.5×10^{10} packets dropped, or a packet dropped or corrupted every $1\frac{2}{3}$ hours, for our example with a 10 Gbps link and 1500-byte packets. That is not a very dynamic congestion control algorithm if we can only send a congestion signal every hour or two!

2.3 Ramping Up with Slow-Start

During the slow-start phase, the instantaneous throughput $T(t)$ in ppr is given by $T(t) = 2^{\frac{t}{R}}$ ppr. To reach a speed T in ppr, it takes time $t(T) = R \log_2(T)$ seconds. Using the parameters above, a TCP connection requires a window of 83,333 packets to achieve a speed of 10 Gbps, and it requires roughly 17 round-trip times (or less than two seconds, using the parameters above) to achieve this speed during slow-start. This seems like an acceptably-fast ramp-up time for many purposes.

2.4 Recovery from Consecutive Time-Outs

If the congestion window is reduced to $w = 1$ MSS, and the TCP sender is forced to linearly increase the congestion window after that, then it takes $W - 1$ round-trip times to achieve a congestion window of W. In this case, it would take 83,332 round-trip times to achieve a speed of 10 Gbps, given our parameters above. Given those parameters, this is 8,333 seconds, more than two hours.

After a single timeout occurs at congestion window W, the TCP sender slow-starts back up to a window of $W/2$, recovering reasonably rapidly to that state. However, if a second timeout occurs when the congestion window is one, for example, because the retransmitted packet is itself dropped or corrupted, then there is essentially no slow-start, and the sender has to linearly increase its congestion window after that.

3 A Proposal for a Higher-Speed TCP

This section describes a proposal for a *higher-speed TCP* that would be able to achieve speeds of 10 Gbps with realistic packet drop rates.

3.1 Requirements

Performance Goals:

- Sustain high speeds without requiring unrealistically low loss rates.
- Reach those high speeds reasonably quickly when in slow-start.
- Recover from multiple time-outs or other periods with small congestion windows without overly burdensome delays.

Compatibility Goals:

- When coexisting with unmodified TCPs, the ideal goal would be not to decrease the speed of the unmodified TCPs in any setting (meaning that the unmodified TCP gets as much bandwidth competing against the higher-speed TCP as it would competing against with another unmodified TCP). This is probably a unrealistically stringent goal, so that more realistic goal is that an unmodified TCP competing with a higher-speed TCP would get as much bandwidth as it would with some low-level ambient (*i.e.*, non-congestion) packet loss rate p_a. That is, we say that the higher-speed TCP is no worse, for the unmodified TCP, than a low-level ambient packet loss rate p_a. If p_a is in the range of, say, 0.1%, this seems like an acceptable compatibility goal.

- The higher-speed TCP should be incrementally deployable, with no router involvement.

It is not a performance goal of our work that a transfer should be completed in a small number of round-trip times. Accomplishing this would seem to require explicit feedback from all of the routers along the path, and we are not attempting this in this work. In particular, we are not considering any start-up behaviors faster than TCP's current slow-start behavior of doubling its congestion window each round-trip time.

3.2 Basic Approach

Our approach is to start with the TCP response function

$$T = \frac{\sqrt{1.5}}{\sqrt{p}}\text{ppr},$$

to decide how we would like to modify the response function at very low packet drop rates to achieve a 10 Gbps TCP with realistic packet loss rates, and finally to decide how to translate this response function to modified window increase and decrease parameters a, b, and c. Because we do not want to more than double the congestion window in one round-trip time, in the absence of explicit feedback from the routers, we leave the slow-start parameter c set to one packet, as in the current TCP.

For compatibility with existing TCPs, higher-speed TCP uses TCP's standard increase and decrease parameters when the current packet drop rate p is greater than some value P (or, roughly equivalently, when the current congestion window is at most W packets). As the default in NS, we choose $P = 0.0015$, corresponding to $W = 31$.

Next, we decide on the target packet drop rate for a congestion window of W_1 packets, say for $W_1 = 83,333$, corresponding to a sending rate of 10 Gbps for our parameters for the round-trip time and packet size. For example, consider a packet drop rate of $P_1 = 10^{-k}$ for a congestion window of W_1 packets. As the default in NS, we choose $W_1 = 83000$, and $P_1 = 10^{-7}$. This compared to the loss rate of at most 10^{-10} required by standard TCP for maintaining that congestion window.

Given the two points (P, W) and (P_1, W_1) for the response function for highspeed TCP, for $p < P$, and assuming that we choose for the new response function to still be linear on a log-log scale, we end up the the following response function for highspeed TCP:

$$w = 10^{S(log p - log P) + log W}$$

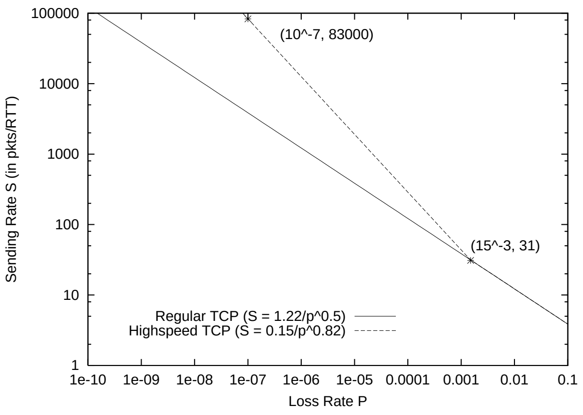

Figure 1: New and old response functions.

for

$$S = (log W_1 - log W)/(log P_1 - log P).$$

This gives:

$$w = p^S (1/P)^S W.$$

For example, for (P, W) set to (0.0015, 31), and (P_1, W_1) set to (10^{-7}, 83000), as in Figure 1, this gives $S = -0.82$, for the following response function:

$$w = \frac{0.15}{p^{0.82}}.$$

Figure 2 shows the relative fairness between highspeed TCP and regular TCP for these values.

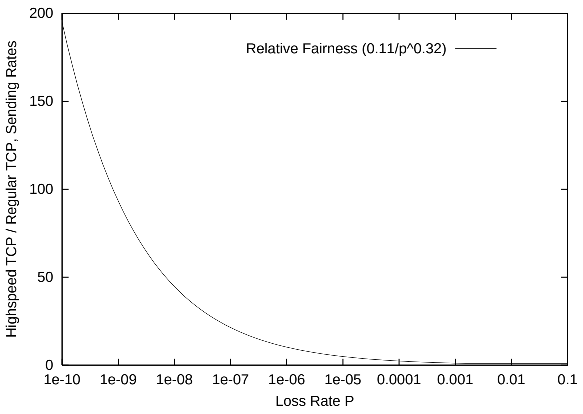

Figure 2: Relative fairness.

We let the increase and decrease parameters $a(w)$ and $b(w)$ be functions of the current window size w when the congestion window is larger than the default value W packets. That would mean that for a congestion window of W_1 packets, we would want to choose the increase and decrease parameters $a(w)$ and $b(w)$ to satisfy the following equation:

$$W_1 = \frac{\sqrt{\frac{a(w)(2-b(w))}{2b(w)}}}{\sqrt{P_1}}\text{ppr}.$$

Thus, we have an equation for $a(w)$ as a function of $b(w)$, for $w = W_1$.

In particular, we require the following:

$$a(w) = W_1^2 * P_1 * 2b(w)/(2 - b(w)).$$

This would be satisfied by $a(w) = 72$ and $b(w) = 0.1$, for example, for (P_1, W_1) set to (10^{-7}, 83000).

This would translate to a decrease of 10% after a drop, and an increase of just under 0.1% per round-trip time in the absence of a drop or mark.

Thus, for $w \leq W$ packets, we use TCP's current increase/decrease parameters;

$$a(w) = 1 \quad \text{for } w \leq W$$

$$b(w) = 0.5 \quad \text{for } w \leq W$$

For an increase parameter $a(w)$ and decrease parameter $b(w)$, we have the following response function:

$$w = \frac{\sqrt{\frac{a(w)(2-b(w))}{2b(w)}}}{\sqrt{p}} \text{ ppr.}$$

Our goal is to choose $a(w)$ and $b(w)$ so that the response function gives W ppr for a packet drop rate of $p(W) = P$, and W_1 ppr for a packet drop rate of $p(W_1) = P_1$. We have the following for the desired packet drop rate $p(w)$, for $w \geq W$:

$$\log(p(w)) = (\log(P_1) - log(P))\frac{\log(w) - \log(W)}{\log(W_1) - \log(W)} + log(P).$$

For the decrease parameter $b(w)$, we have $b(W) = 0.5$, and $b(W_1) = B$. For $w \geq W$, we let $b(w)$ vary linearly as the log of w, as follows:

$$b(w) = (B - 0.5)\frac{\log(w) - \log(W)}{\log(W_1) - \log(W)} + 0.5.$$

The increase parameter $a(w)$ can then be computed using the standard TCP equation,

$$a(w) = \frac{w^2 * 2.0 * b(w) * p(w)}{2.0 - b(w)},$$

to give the desired packet drop rate $p(w)$ for that window w.

However, the design space to be explored includes a range of tradeoffs between more severe or more moderate increases and decreases. At the most severe extreme, the constraints between $a(w)$ and $b(w)$ would be satisfied by $b(w) = 0.5$, for a decrease of 50% for any value of the congestion window w. And at the most moderate extreme, we could consider $a(w) = 1$ for any value of the congestion window w, for an increase of 1 packet per round-trip time. However, neither of these two extremes are attractive alternatives. As a plausible tradeoffs between overly-large and overly-small increases per round-trip time, we let the decrease function $b(W_1)$ be 0.1, for the large window W_1.

Other possibilities that we have not yet explored would be to have the increase and decrease parameters $a(w)$ and $b(w)$ be functions of the longer-term packet drop rate, or of the number of packets acknowledged since the last drop, as well as depending on the current congestion window w.

4 Evaluation Strategy

We have implemented our higher-speed TCP in NS, and have just begun the process of evaluating it for fairness, dangers of congestion collapse, and other properties. [1] We have also validated the fairness of HSTCP flows competing with other highspeed flows and with regular unmodified TCP flows.

We have performed preliminary simulations examining the transient fairness of HSTCP in the face of rapidly fluctuating available bandwidth.

One of the fairness scenarios would be to consider a transient fairness when a new high-speed TCP starts up, in slow-start, in an environment with existing high-speed or unmodified TCPs. A second fairness scenario would be to consider the transient fairness when a high-speed TCP has had a low congestion window, because of congestion downstream, and then has to increase its congestion window in the absence of slow-start when the downstream congestion is removed.

The initial simulations make it clear that additional work is also needed to investigate the potential for unwelcome oscillations and synchronization when multiple multiple high-speed TCPs are sharing a link.

Because our modified TCP still uses ACK-clocking, in that the TCP data sender still requires incoming ACK packets to clock new out-going data packets, we do not expect significant dangers of congestion collapse, even with the more aggressive window increase parameters at higher congestion windows.

We hope to also have this proposal tested with real-world experiments in the TABS project.

5 Future work

We are assuming a model where the modified TCP still is sensitive only to the presence or absence of a mark or drop in a window of data, and is not sensitive to the *number* of marks or drops in a window of data. Future work could explore more thoroughly the limitations imposed by this assumption.

It is also possible that this very-high-bandwidth and very-high-congestion-window regime will raise new issues regarding active queue management for this part of the design space.

[1] In order to avoid global synchronization from many TCP connections having packets dropped in the same round-trip time, we changed the minimum value for max_p in Adaptive RED from 0.01 to something smaller.

References

[FF99] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking*, Aug. 1999. URL http://www-nrg.ee.lbl.gov/floyd/end2end-paper.html.

[FHP00] S. Floyd, M. Handley, and J. Padhye. A comparison of equation-based and aimd congestion control, February 2000. URL http://www.aciri.org/tfrc/.

[GRK99] P. Gevros, F. Risso, and P. Kirstein. Analysis of a method for differential tcp service. Dec.. 1999.

[Jac88] V. Jacobson. Congestion avoidance and control. pages 314–329, 1988. An updated version is available via ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z.