D Getting up to Speed Fast: Detailed Analyses

D.1 TCP Average Rate Model

This is a model of the average rate of a single TCP flow in dedicated bottleneck capacity, defined using the following independent variables:

Link capacity, X [b/s]

Round trip time, R [s]

Initial window, i [pkt]

Packet size, S [B] or $s = 8S$ [b]

Flow size, F [B].

Flow size, $f = F/S$ [pkt]

Bandwidth-delay product, $W = XR/s$ [pkt].

The general approach is to find the number n of round trips in which slow-start only partially fills the bottleneck, *before* the round in which either slow-start ends or the flow ends, if sooner. Then any remaining packets will arrive at the bottleneck rate, whether they are all sent in the next round while still in slow-start phase or there are enough to progress into the fast retransmit and congestion avoidance phases. In the latter case, the number of retransmissions is added to the number of packets to be forwarded before completion. One round must be added between the receiver's request and the start of the response.[36] Then **completion time** T between the client's request and it receiving the last data consists of:

$$T = (n+1)R + (m+M)s/X,$$

The analysis below finds the values of n, m & M,

Round index, n [integer]

Remainder, m is the remaining packets of the flow that all leave the bottleneck at rate X

Retransmissions, M due to losses during the overshoot at the end of slow-start.

We define the number of packets in slow-start as f_s. Index u [real number] and the final window w [pkt] are only used to derive formulae during slow-start, outside which they are invalid:

$$\begin{aligned}
&\text{If } f \geq i \\
&\qquad w = i2^u \\
&\qquad \sum_{j=0}^{n} i2^j \leq f_s && < \sum_{j=0}^{n+1} i2^j \\
&\qquad i(2^{(n+1)} - 1) \leq f_s && < i(2^{(n+2)} - 1) \\
&\qquad n \leq \lg(f_s/i + 1) - 1 && < n + 1 \\
&\qquad n = \lfloor \lg(f_s/i + 1) \rfloor - 1; \\
&\text{If } f < i \\
&\qquad n = 0.
\end{aligned}$$

[36] Any handshaking rounds are excluded by the assumption that either the flow is re-starting after idle or using TCP fast open after an earlier connection between the same hosts.

The critical flow size, f_c is defined as the number of packets that can be sent until buffer overflow ends slow-start, and the critical round as the real number u_c. The buffer is assumed perfectly sized for the flow, at 1BDP = W. Therefore at the critical flow size, the critical window,

$$\begin{aligned} w_c &= 2W \\ &= i2^{u_c} && \text{if } 2W \geq i \\ \text{when } f_c &= i(2^{(u_c+1)} - 1) \\ &= 2i2^{u_c} - i \\ &= 4W - i && \text{if } 2W \geq i; \\ f_c &= 2W && \text{if } 2W < i; \\ f_s &= \min(f, \quad \max\big(2W,\ (4W - i)\big); \end{aligned}$$

As an implementation trick, we can use $n = \lfloor \lg(f_s/i + 1) \rfloor - 1$ to cause the condition $f < i$ to be flagged by the special value $n = -1$, then, the number of packets, e, in partially empty rounds of slow-start is the number of packets in n rounds,

$$\begin{aligned} e &= i(2^{(n+1)} - 1) \\ m &= f - e \\ M &= \min\big(\max(i, 2W),\ \max(0, (m - 2W))\big). \end{aligned}$$

Explanation of the last formula for M: The buffer always has time to empty the packets sent in the the round before the round in which overflow occurs (if it occurs). The number of packets dropped is the number of packets not constrained by slow-start m less the 2 BDP of packets that the link can forward or buffer, but limited by the number that will ever be sent in one round during SS or CA, which is also $2W$ (or the initial window i if it is larger, so it can overflow the buffer on its own).

Finally, we must not forget to modify the completion time formula to strip out the special flag $n = -1$, which means $n = 0$ and $f < i$:

$$T = (\max(0, n) + 1)R + (m + M)s/X.$$

Then the **average rate** $\bar{x}$ is simply,

$$\bar{x} = 8F/T.$$

D.2 QV at multiple bottlenecks

Formally, for constant rate flows and two constant queues, q_1 & q_2 in series, the resulting spacing between 0 markings is:

$$\begin{aligned} q_1; && \text{if } q_1 \geq q_2 \\ \lceil q_2/q_1 \rceil q_1; && \text{if } q_2 \geq q_1 \end{aligned}$$

The rounding up in the second case is on the safe side and anyway, in more realistic scenarios with varying flow rates, the spacing approaches the desired length of the longer queue, q_2.

D.3 Parallel Slow-Starts

If a source opens m parallel flows all using the same slow-start with initial window i, it only gives equivalent performance to one flow in slow-start using a larger initial window mi (but without all the overhead of the extra flows). The total window across all flows still only doubles in the rounds after the first.

The increase in bit-rate from using m-times larger packets is also equivalent to using an m-times larger initial window or m parallel flows (but without so much packet overhead). However, in this case, less packets means less per-packet information can be gleaned, either from inter-packet delays or from QV.

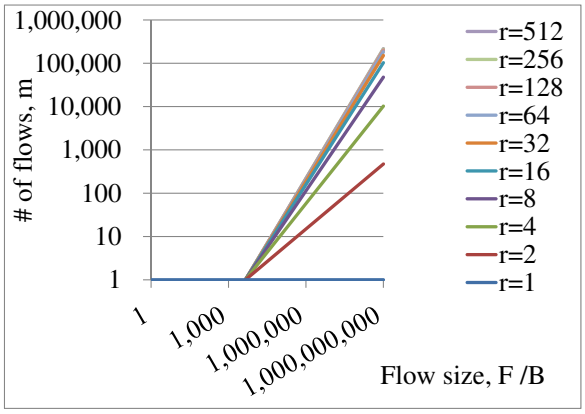

Figure D.1: The number of parallel flows m required to achieve speed-up ratio r for various flow sizes F

Opening parallel flows (or equivalently increasing IW or packet size) only offers a modest increase in average rate. Put the other way round, to achieve a decent speed-up ratio r requires a stupidly large number of parallel flows m.

The following analysis proves this by approximating from the model of slow-start in subsection D.1 on condition that $f \gg i$, where f is the flow-size in packets. The notation $\overline{x}_m$ or T_m means the average rate or completion time for m parallel flows (or equivalently IW=mi or m times larger packets).

$$
\begin{aligned}
T_m &\approx \lg(f/mi); \\
r &= \frac{\overline{x}_m}{\overline{x}_1} \\
&= \frac{T_1}{T_m} \\
&\approx \frac{\lg(f/i)}{\lg(f/mi)} \\
&\approx \frac{\lg(f/i)}{\lg(f/i) - \lg m} \\
\lg m &\approx (1 - 1/r)\lg(f/i) \\
m &\approx 2^{\left((1-1/r)\lg(f/i)\right)}.
\end{aligned}
$$

Figure D.1 illustrates this formula for the number of parallel flows m required to achieve various speed-up ratios r for a range of flow sizes $F - fs$, where s is the packet size.

For a 100kB transfer, it can be seen that a modest speed-up ratio of $r = 4$ can be achieved with 18 parallel flows. Although 18 flows implies a lot of overhead, it is still easier than waiting for QV to be deployed.

However, to speed up even slightly larger transfers quickly becomes stupidly infeasible. For instance, to speed-up a 1MB transfer 4 times requires 159 flows and to speed up a 10MB flow 4 times would need 1,373 flows.

Clearly it would be similarly infeasible to increase the initial window 1,373 times (to over 4000 1500B packets), because the initial window has to be set conservatively to fit easily into any size buffer that may be encountered anywhere on the Internet.