This note is a reply to some of the points about buffer sizing raised by Dhamdhere and Dovrolis [1]. I’ll use a simple mathematical model to make stark some of the tradeoffs that need to be weighed when deciding on buffer size.

**Efficiency.** The most important finding in [1] is that in all their simulations, with both small and large buffers, “the utilization of the target link has remained at almost 100%”. The debate is therefore about which flows should be the winners and which should be the losers when capacity is shared out.

**Fairness.** Here is a simple model. Consider a single link shared by several long-lived flows. The round trip time of a flow is composed of propagation delay and queueing delay. Let $PT_i$ be the propagation delay of flow $i$, and let the common queueing delay be $QT$. According to the classic TCP throughput equation, the throughput of flow $i$ is roughly

$$\theta_i = 0.87 \frac{C}{(PT_i + QT)\sqrt{p}}$$

where $p$ is the per-packet drop probability. As the link is fully utilized, $\sum \theta_i = C$ where $C$ is the link capacity. Thus

$$\theta_i = \frac{C}{QT + PT_i} / \sum_j \frac{1}{QT + PT_j}.$$

To illustrate this, consider a single link shared by $N$ long-lived flows, with $PT$ drawn from a truncated exponential distribution with minimum 30ms, maximum 540ms, and harmonic mean 60ms, the parameters in [1]. The throughput of a randomly chosen flow is then

$$\frac{C}{QT + PT} / \mathbb{E}\left(\frac{1}{QT + PT}\right)$$

and, with $C/N = 80$Bps, the CDF is

![CDF](attachment:image.png)

Note that the system becomes completely fair as $QT \rightarrow \infty$.

**QoS.** The TCP throughput equation also tells us about drop probability. As pointed out in [1], the solution for $p$ is

$$p = \left[ \frac{0.87 C^2}{(QT + PT)^2} \right]^{1/2}.$$

With the same parameters as before,

![Graph](attachment:image.png)

By reducing $QT$ we make the link ‘more attractive’ to TCP, so the flows increase their rates, so drop probability goes up. (This is only a concern when the link is overloaded i.e. the capacity-per-flow $C/N$ is too low.)

**Conclusion.** Jacobson [2] believed that “While algorithms at the transport endpoints can insure the network capacity isn’t exceeded, they cannot insure fair sharing of that capacity.” It’s possible to force TCP to be fair, by making queueing delay large as suggested by [1]. But there are other ways to achieve fairness, for example with clever AQM schemes, or by modifying TCP so as to remove the dependence on round trip time (a change to the window increase rule [3]). To rely on large queueing delay is to use router buffers as sticking plaster for an historical accident in the design of TCP.

Large buffers will reduce drop probability, but at the cost of increasing round trip time. For long-lived TCP flows the tradeoff is immaterial, since all that matters is throughput. For other flows, and especially for real-time flows, I believe that it is the job of the network to provide a low-delay service. One can always mitigate the effects of loss by using forward error correction, or even by buffering at the end-system and relying on retransmission; but if the network slaps on an extra 100ms of queueing delay then it is impossible to win back that lost time. I believe that small buffers will lead to a more flexible and evolvable Internet.

**References.**

