Chapter 2

Buffer Sizing Rules: Overview and Analysis

As seen in Chapter 1, based on the rule-of-thumb, buffers need to be at least as large as the delay-bandwidth product of the network, i.e., $RTT \times C$, to achieve full utilization.

In this chapter, we study a CIOQ router model with contention buffers at the input ports and congestion buffers at the output ports (Figure 1.1). We first consider the congestion buffers, explain the origins of the rule-of-thumb for determining the size of these buffers, and briefly review the case with many TCP flows on the bottleneck link. We then give an overview of the *tiny buffers* analysis [25], which shows the feasibility of making the congestion buffer size only a few dozen packets. Finally, we consider the contention buffers in the CIOQ model, and show that the tiny buffers rule could also be applied to the contention buffers at the input side of the router.

2.1 How big should the congestion buffers be?

To understand how large to make the congestion buffers, we first study output-queued routers, which only have congestion buffers, and packets are immediately transferred to the output ports as soon as they arrive. Each output port has one FIFO queue, which is shared by the flows going through that port. The size of the buffer depends on the arrival traffic: If traffic is light or non-bursty, buffers can be very small; if big bursts are likely to arrive, buffers need to be much larger.

In what follows, we explore how large to make the congestion buffers under three scenarios:

1. When a link carries just one TCP flow. This turns out to be the worst-case, and leads to the rule-of-thumb $B = RTT \times C$.

2. When a link carries many TCP flows, allowing us to reduce the buffer size to $B = \frac{RTT \times C}{\sqrt{N}}$.

3. Finally, when traffic comes from slow access networks, or when the source paces the packets it sends. In this case, we can reduce the buffer size to a few tens of packets.

2.1.1 When a link carries just one TCP flow

To understand why we need $RTT \times C$ buffers with just one TCP flow, we need to understand the dynamics of TCP. The dynamics of a TCP flow are governed by the window-size (the number of outstanding unacknowledged packets). A long-lived flow spends most of its time in the additive-increase and multiplicative-decrease (AIMD) congestion avoidance mode, during which the window size increases additively upon receiving an ACK packet, and is halved when a packet or ACK is lost.

To maximize the throughput of the network, the buffer in a router's output port needs to be big enough to keep the outgoing link busy during times of congestion. If the buffer ever becomes empty, the link goes idle and we waste the link capacity.

On the other hand, TCP's sawtooth congestion control algorithm is designed to fill any buffer, and deliberately causes occasional loss to provide feedback to the sender. No matter how big we make the buffers at a bottleneck link, TCP will occasionally overflow the buffer. Consider the simple topology in Figure 2.1, where a single TCP source sends data packets to a receiver through a router. The sender's access link is faster than the receiver's bottleneck link of capacity C, causing packets to be queued at the router. The sender transmits a packet each time it receives

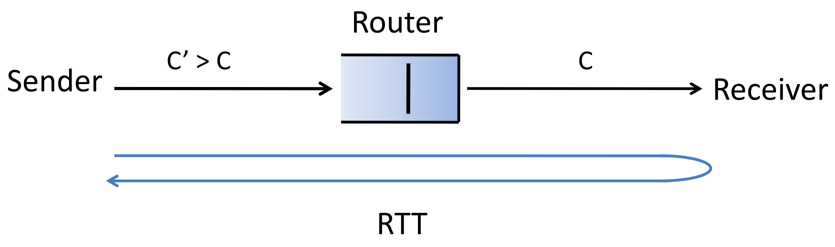

Figure 2.1: Single-bottleneck topology. The sender's access link is faster than the receiver's bottleneck link, causing packet accumulation in the router.

an ACK, and gradually increases the number of outstanding packets (the windows size), which causes the buffer to gradually fill. Eventually a packet is dropped, and the sender does not receive an ACK. It halves the window size and pauses until the number of outstanding packets has fallen to $\frac{W_{\max}}{2}$ (where $W_{\max}$ is the peak window size). Figure 2.2 shows the window size dynamics of a single flow going through a bottleneck link. The key to sizing the buffer is to make sure that while the sender pauses, the router buffer does not go empty and force the bottleneck link to go idle.

The source pauses until it receives $\frac{W_{\max}}{2}$ ACK packets, which arrive in the next $\frac{W_{\max}}{2C}$ seconds (remember that C is the bottleneck rate). During the pause, $\frac{W_{\max}}{2}$ packets leave the buffer; for the bottleneck link to stay busy, the buffer needs to hold at least $\frac{W_{\max}}{2}$ packets when the pause starts. Now we just need to determine $W_{\max}$.

At the instant the pause is over, the source can send $\frac{W_{\max}}{2}$ consecutive packets as $\frac{W_{\max}}{2}$ ACKs arrive. It then pauses until it receives an ACK one RTT later (the first ACK arrives after exactly RTT because the buffer is empty). In other words, the source sends $\frac{W_{\max}}{2}$ packets in RTT seconds, which must be just enough to keep the bottleneck link busy; i.e., $\frac{W_{\max}/2}{RTT} = C$, which means $B = RTT \times C$, the rule-of-thumb for one TCP flow.

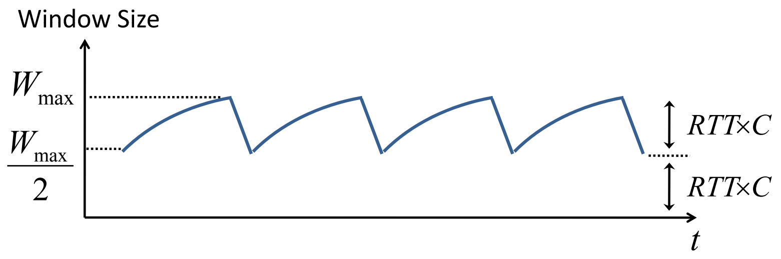

Figure 2.2: Window size dynamics of a TCP flow going through a bottleneck link. To achieve 100% utilization, the buffer size needs to be large enough to store $RTT \times C$ packets.

2.1.2 When many TCP flows share a link

If a small number of flows share a link, the aggregate window size (the sum of their individual window sizes) tends to follow the same TCP sawtooth, and B is the same as for one flow.

If many flows share a link, small variations in round-trip time and processing time desynchronize the flows [40, 29, 26]. Therefore, the aggregate window size gets smoother. This is studied in detail in [9], where it is shown that with N long-lived TCP flows, the variation in the aggregate window size scales down by a factor $\sqrt{N}$. As with one flow, the variation in the aggregate window size dictates the buffer size needed to maintain full utilization of the bottleneck link. Hence $B = \frac{RTT \times C}{\sqrt{N}}$. With 10,000 flows on a link, this suggests the buffer size could be reduced by 99% without any change in performance (i.e., a 1Gb buffer becomes 10Mb). These results have been found to hold broadly in real networks [9, 11].

2.1.3 When traffic comes from slow access networks

In backbone networks, in addition to the above multiplexing effect, each individual flow gets smoother too. This is because a backbone network interconnects many slower networks. When packets from slower networks are multiplexed together onto a fast backbone, the bursts are spread out and smoothed. Hence, in backbone networks not only is the aggregated TCP's AIMD sawtooth smoothed, but also the underlying traffic arrivals are smoothed. We will see that the smoothing substantially reduces the required buffer size.

To get a feel for how smoothing could help reduce the buffer size, imagine for a moment that the traffic is so smooth that it becomes a Poisson process. The drop rate has an upper bound of ρ^B, where ρ is the load, and B is the buffer size. At 80% load and with only 20-packet buffers, the drop rate will be about 1%, independent of RTT and C. At the other extreme, compare this with the buffer size needed for 100% utilization with a single TCP flow, when RTT is 200ms and C is 10Gb/s; $B = 2$Gb, or about a million average sized packets.

Traffic in backbone networks cannot be modeled as a collection of independent Poisson flows, since a TCP flow can send a whole window of packets at the beginning of a round-trip time, creating significant bursts. But there are two ways the bursts can be broken. We can explicitly break them by using Paced TCP [7], in which packets are spread uniformly over the round-trip time. The rate and behavior of each flow are almost indistinguishable from regular TCP, but as we will see shortly, the amount of required buffering drops dramatically.

Even if we do not modify the TCP source, the burst is naturally broken if the core links are much faster than the access links, as they typically are. As the packets from one flow enter the core, they are spread out, with gaps, or packets from other flows being multiplexed between them.

To see how breaking the bursts reduces the required buffer size, we start by analyzing TCP pacing. Sources follow the AIMD dynamics, but rather than sending out packets in bursts, they spread traffic over a round-trip time.

Assume that N long-lived TCP flows share a bottleneck link. Flow i has a time-varying window size $W_i(t)$ and follows TCP's AIMD dynamics. If the source receives an ACK at time t, it will increase the window size by $1/W_i(t)$, and if the flow detects a packet loss, it will decrease the congestion window by a factor of two. In any time interval $(t, t']$ when the congestion window size is fixed, the source will send packets as a Poisson process at rate $W_i(t)/RTT$. Under this assumption buffering $O(\log W_{\max})$ packets is sufficient to obtain close to peak throughput. More precisely, to achieve effective utilization of θ, buffer size

$$B \geq \log_{1/\rho} \frac{W_{\max}^2}{2(1-\theta)} \tag{2.1}$$

suffices [25], if the network is over-provisioned by a factor of $1/\rho$. Here, ρ is assumed to be less than or equal to one.

This result assumes that the network is over-provisioned. In other words, it assumes that the maximum traffic rate—with all TCP sources simultaneously transmitting at their maximum rate—is $1/\rho$ times smaller than the bottleneck link bandwidth. Here, θ is the desired effective utilization of the shared link. It represents the fraction we aim to achieve out of the maximum possible utilization ρ (i.e., a fraction $\rho\theta$ of the full link rate). Although this result has not been extended to the under-provisioned case, the simulation results of Chapter 3 indicate that over-provisioning is not a requirement.

The above result suggests that TCP traffic with $W_{\max} = 83$ packets and $\rho = 75\%$ needs a buffer size of 37 packets to achieve link utilization $\theta\rho$ of 70%.

According to Equation 2.1, the buffer size needs to increase only logarithmically as the maximum window size grows larger. In a TCP connection, $W_{\max}$ is the maximum amount of data the transmitter can send over one RTT. This amount is limited by the source transmission rate, even if the operating system does not explicitly limit $W_{\max}$: at a source rate of C_T , at most $C_T \times RTT$ units of data can be sent over a round-trip time. If this amount increases from 10KB to 100MB, then the buffer size only needs to be doubled.

What happens if instead of TCP Pacing we simply rely on the multiplexing of flows from slow access links onto fast backbone links? It is shown in [25] that if access links run at least $\log W_{\max}$ times slower than the bottleneck link, then approximately the same buffer size as in Equation 2.1 is required. In our example above, $\log W_{\max}$ was less than seven, whereas in practice access links are often two orders of magnitude slower than backbone links (for example, a 10Mb/s DSL link multiplexed eventually onto a 10Gb/s backbone link). Under these conditions, the packet loss probability is

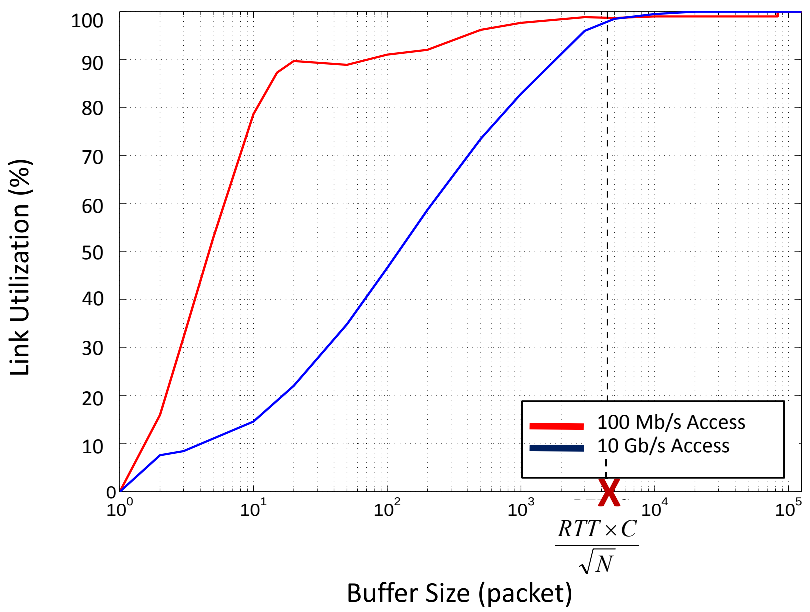

Figure 2.3: Link utilization vs. buffer size. With 800 flows on the link, close to 100% utilization will be achieved if buffer size is $\frac{RTT \times C}{\sqrt{N}}$. If flows come from slower access links, a tiny buffer size of 10 packets suffices for 80% utilization.

comparable to Poisson traffic with the same buffer size.

To compare the results of Sections 2.1.1-2.1.3, we illustrate them through the simulation of a 10Gb/s bottleneck link with 800 long-lived TCP flows sharing the link (Figure 2.3). The average RTT is 100ms. We measure link utilization as we vary the link's buffer size from only one packet to $RTT \times C = 125,000$ packets. As the graph shows, utilization remains almost unchanged (above 99%) with buffer sizes larger than $\frac{RTT \times C}{\sqrt{N}} \approx 4419$ packets. With access links running at 100Mb/s, i.e., 100 times slower than the bottleneck link, we can set the buffer size to only 10 packets and achieve close to 80% utilization.

2.2 How big should the contention buffers be?

Now we turn our attention to the size of the contention buffers.

The size of contention buffers in a CIOQ switch depends on the internal speedup of the switch (i.e., how fast the switch fabric runs compared to the link rate). Larger speedups reduce the average number of packets waiting at the input side, since packets are removed faster from input buffers. We show that when speedup is greater than or equal to two, the occupancy of contention buffers on any port is less than twice the size of congestion buffers. In other words, buffers of size $O(\log W_{\max})$ at input ports lead to the same performance as in an output-queued switch.

Definition: Consider two routers R and S, and assume that the same input traffic is fed to both routers. Router R is said to exactly emulate router S, if it has exactly the same drop sequence and the same departure sequence as router S.

With unlimited buffer size, a CIOQ router with speedup two can exactly emulate an OQ router [21]. In other words, despite the contention at the ingress side, there exists an algorithm that guarantees not to keep packets longer than what an OQ router does. Now assume that S is an OQ router, and R is a CIOQ router, both with output buffers of size B. Consider the scenario where router R drops an arriving packet exactly when there is a drop at router S (i.e., when the total number of packets destined for a given output port exceeds B). With such emulation, we show that the occupancy of the input buffers in router R is limited according to the following theorem.

Theorem 2.1. If router R exactly emulates router S, then at any time t, $Q(i,t) \leq 2B$, where B is the output buffer size in both routers, and $Q(i,t)$ is the buffer occupancy of Router R at input port i.

Proof. Assume the contrary. There must be a time t_0, and an input port i_0 such that $Q(i_0,t_0) > 2B$. With speedup of two, at most two packets are removed from port i_0 at any time slot. Therefore, there is a packet in router R that cannot be sent out in B time slots. This contradicts the exact emulation assumption, since any packet

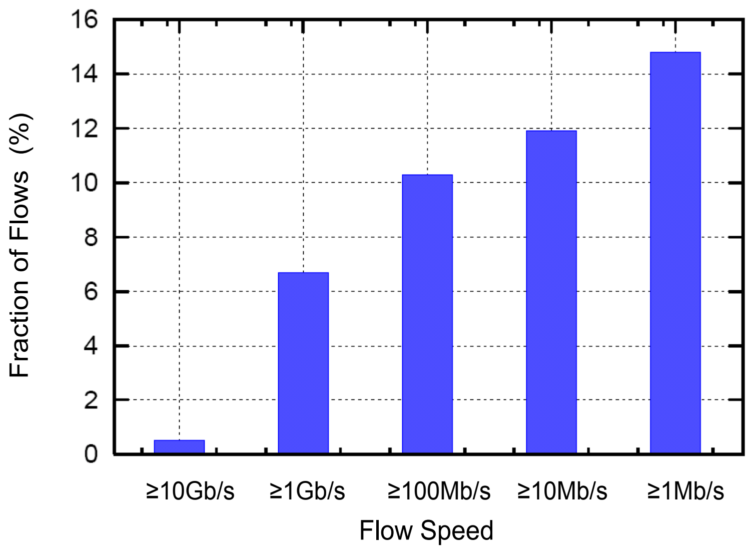

Figure 2.4: Cumulative distribution of access bandwidth in a commercial backbone network. The backbone link runs at 10Gb/s.

in the OQ router should be sent out in at most B time slots. $\square$

Our analysis assumes that a stable marriage algorithm [21] controls the switch configuration. However, the simulation results shown in Chapter 3 suggest that even under more practical algorithms, using very small ingress buffers results in high utilization. For example, in an 8×8 switch, setting the input buffer size to only five packets per virtual output queue (VOQ) gives 80% link utilization.

2.3 Core-to-access bandwidth ratio

Figure 2.4 shows the distribution of access link bandwidth in a commercial backbone network. This data is based on traces collected by CAIDA [4] in May 2008 from an OC192 backbone link running at 10Gb/s between Seattle and Chicago.

For each value on the x axis, the bar shows the percentage of flows that appear on the backbone link faster than that value. For example, about 10% of flows are faster than 100Mb/s. We measure the speed of a flow by finding the minimum time interval between any two consecutive packets of the flow. Note that this measurement shows the speeds at which the flows appear on the backbone link, and may not reflect their exact access link bandwidths. This is because the time intervals between packets can be changed by upstream buffering.

With the distribution shown in Figure 2.4, our assumption of an average ratio of 100 (made in our simulations and experiments that will be described in the next two chapters) seems conservative. We can see that about half a percent of flows are as fast as the backbone link (10Gb/s). However, the majority of flows (more than 85%) are either slower than 1Mb/s, i.e., 10,000 times slower than the backbone link, or they are shorter than three packets (for which we cannot have an accurate measurement).