Buffer Sizing results for RCP Congestion Control under Connection Arrivals and Departures

Ashvin Lakshmikantha
ECE and CSL
UIUC
ashvin213@gmail.com

R. Srikant
ECE and CSL
UIUC
rsrikant@illinois.edu

Nandita Dukkipati
Computer Systems Lab
Stanford
nanditad@stanfordalumni.org

Nick McKeown
Computer Systems Lab
Stanford
nickm@stanford.edu

Carolyn Beck
IESE and CSL
UIUC
beck3@illinois.edu

ABSTRACT

Buffer sizing has received a lot of attention recently since it is becoming increasingly difficult to use large buffers in high-speed routers. Much of the prior work has concentrated on analyzing the amount of buffering required in core routers assuming that TCP carries all the data traffic. In this paper, we evaluate the amount of buffering required for RCP on a single congested link, while explicitly modeling flow arrivals and departures. Our theoretical analysis and simulations indicate that buffer sizes of about 10% of the bandwidth-delay product are sufficient for RCP to deliver good performance to end-users.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network protocols - congestion control

General Terms

Modeling of communication Networks, Theory

Keywords

Internet, Rate control

1. INTRODUCTION

We would like to study the performance of RCP (Rate Control Protocol) congestion control when there are flow arrivals and departures. RCP has been extensively studied through simulations, modeling, and experiments; in particular scalability and stability of RCP have been established and simulations suggest it is very promising under a broad range of conditions. In this work we are interested in a practical aspect of RCP: how much buffering does RCP require at routers and switches? Specifically, our goal is to study the impact of buffer size on the average flow completion time (AFCT) when RCP is used.

The RCP protocol was introduced in [10], which also describes the algorithm and presents simulations showing short flow-completion times. The argument for flow completion time being the appropriate metric for congestion control is presented in [11]. In [4], control theory is used to show that RCP is stable independent of the link-capacities, number of flows and network round-trip times. It is well known that RCP achieves max-min fairness. Variants of RCP which achieve other forms of fairness have been proposed and studied in [19]. In [9, 24], an RCP testbed using real implementations in software (Linux-based) and hardware (using Stanford's NetFPGA) is described.

We briefly review the RCP algorithm here to motivate why it is interesting to study its buffer-size requirements. In the basic RCP algorithm a router maintains a single rate, $Z(t)$, for every link. The router "stamps" $Z(t)$ on every passing packet (unless it already carries a smaller value). The receiver sends this value back to the sender so that it knows the smallest (or bottleneck) rate along the path. In this way, the sender quickly learns what rate it should be using (without the need for slow-start). The router updates $Z(t)$ approximately once per roundtrip time (RTT), and strives to emulate processor sharing among flows. Intuitively, to emulate processor sharing, the router should offer the same rate to every flow, try to fill the outgoing link with traffic, and keep the queue occupancy close to zero. The RCP rate update equation (1) is based on this intuition (see Section 2 for details). We note that the RCP algorithm shares some similarities to those in [15, 18]. Since the goal of RCP is to quickly learn the available capacity in the bottleneck link, it is interesting to understand whether this feature is also helpful in decreasing the buffer-size requirements at routers in the Internet. We answer this question in the affirmative in this paper.

The rest of the paper is organized as follows: we first briefly explain RCP's mechanisms (Sec. 2), then analytically characterize the amount of buffering RCP requires (Sec. 3), go on to validate our results in simulations (Sec. 4), and finally conclude the paper (Sec. 5). In the interest of space, we avoid presenting proofs of all the theorems and refer readers

to [20] for details.

2. MAIN PROPERTIES OF RCP

Under RCP every router maintains a rate $Z(t)$ called the fair-share rate at time t. This fair-share rate is updated by the router depending on the congestion seen at the router. The rate information is sent to the end-hosts using the explicit feedback mechanism described in Section 1. End hosts transmit at the rate dictated by the router.

Given a single congested link of capacity C accessed by N long-lived flows, the goal of RCP is to ensure that every flow gets a fair-share of the bandwidth. If RCP is stable, then it follows that the equilibrium fair-share rate Z_{eq} under RCP should be equal to

$$Z_{eq} = \frac{C}{N}.$$

If the router knows the number of ongoing flows, N, then the algorithm converges to Z_{eq} instantaneously. However, explicitly measuring N requires per-flow information and is not practical. Therefore RCP estimates N implicitly.

To give an example, consider the same single congested link being accessed by N long-lived flows. For the moment, let us assume that all flows have the same delay (RTT) and the flows are completely synchronized. At time $t = 0$, let $Z(0) = z_0$. Each packet that arrives at the link is stamped with the fair share rate z_0. One RTT later, all flows would have received the network fair-share rate and therefore would adjust their transmission rates to be equal to z_0. Thus the total arrival rate at the router at time $t = RTT$ would be $y(RTT) = Nz_0$. By measuring the total arrival rate y at time $t = RTT$, the router can estimate the number of flows using

$$\widehat{N} = \frac{y}{z_0},$$

where $\widehat{N}$ denotes the router's estimate of the number of flows in the system. Using $\widehat{N}$, the router updates its fair-share rate:

$$Z(RTT) = \frac{C}{\widehat{N}}.$$

In the trivial case with N completely synchronized long-lived flows, the fair-share rate $Z(RTT) = \frac{C}{N}$ independent of z_0. Even though the router has no prior knowledge of N, it can implicitly estimate N accurately without requiring per-flow information and thus converge to the equilibrium fair-share rate.

The naive scheme that we just described is not robust. Measurement errors due to unsynchronized users, different feedback delays, etc. result in large-scale oscillations of the fair-share rate $Z(t)$, which are not desired. RCP improves upon this basic algorithm by using the following rate adaptive scheme:

$$Z(t) = Z(t-d)\left(1 + \frac{\frac{d}{\overline{RTT}}(\alpha(C - y(t)) - \beta\frac{q(t)}{\overline{RTT}})}{C}\right), \quad (1)$$

where $\overline{RTT}$ is a moving average of the round trip times measured across all packets, d is the update interval (the time interval between two successive updates), C is the link-capacity, $y(t)$ is the measured aggregate input traffic rate during the last update interval and $q(t)$ is the instantaneous queue size. The advantage of this scheme is that the parameters α and β can be tuned to ensure stability and achieve desired performance. The parameter α represents the trade-off between stability and response times. A larger value of α results in a faster response time at the expense of reduced stability margins and vice versa. In an earlier study [10], it has been shown that RCP is stable if $\alpha < 1$. The parameter β represents the trade-off between acceptable queueing delay and the fair-share rate during the transient periods. A large value of β results in small queueing delays at the expense of reduced fair-share rate during the transient period and vice versa. The impact of the parameters α, β on the stability and performance of the system has been studied in detail in [4]. We refer the reader to [10, 4] for additional details of the algorithm.

3. BUFFER REQUIREMENTS UNDER RCP

Sizing core-router buffers for TCP has received quite a lot of attention recently [17, 23, 26, 14, 3, 8]. These works suggest that core router buffers can be reduced significantly without compromising on link utilization. These guidelines make use of the fact that a large number of flows access the core router and therefore statistical multiplexing gains substantially reduce the required buffered size.

In the same spirit, it is possible to characterize the buffer size required by RCP enabled routers. The statistical multiplexing benefits that can be reaped in a network with large number of users apply for RCP flows as well. In particular, simulations and intuitive reasoning indicate that fixed buffer sizes (independent of the delay-bandwidth product) can be employed in the core routers of a RCP enabled network. Interested readers can refer to the longer version of this paper [20] for more details on this model.

In this work, we consider scenarios where statistical multiplexing is not a key factor in determining buffer sizing requirements. Instead, we focus on the possible reduction in buffer size due to the fact that RCP provides explicit rate information to the sources. We consider a single congested link used by multiple RCP flows competing for bandwidth. We impose no access speed limitations on the flows. In other words, we allow for the possibility of a single flow being able to saturate the link. We also explicitly model flow arrivals and departures. This model is applicable to the edge routers in the Internet and in data center networks where there are very few flows for statistical multiplexing to be effective. The scenario is similar in spirit to [21] but the models we use here are quite different and specific to RCP.

We derive buffer sizing results for two extreme cases: (a) the mean flow size is large compared to the bandwidth delay product and (b) the mean flow size is small compared to the bandwidth delay product. If the mean flow-size is $\frac{1}{\mu}$, the capacity of the router is C and the feedback delay is RTT, the two scenarios mentioned above correspond to $\mu CRTT \ll 1$ and $\mu CRTT \gg 1$ respectively. For analytical convenience, we assume that all flows have the same RTT. However, our simulations later on are carried out for scenarios where different flows have different RTTs.

The first case is a natural extension of the static scenario

in which a fixed number of flows N stay in the system for an infinitely long-time. In this setting, the number of flows in the network changes very slowly when compared to the convergence times of RCP. Therefore, RCP can track the number of flows in the network quite accurately, thereby making efficient use of link capacity. From the point of view of a congestion control protocol designer, this is the regime of interest.

The second case represents a network that is dominated only by short-flows. The motivation for studying short-flows is the presence of a large number of short flows in the Internet today. Since short-flows cannot be controlled, one could argue that large buffers must be employed to eliminate large-scale packet loss. In this work, we show that even when *no* congestion control is imposed on the short-flows, small buffers do not degrade the performance of RCP significantly.

The Internet consists of a large number of short-flows that contribute to a small fraction of the traffic volume, and a small number of long-flows that contribute to a large fraction of the traffic volume. Our analysis considers the two extreme cases (short-flows and long-flows) while the mixture is studied in simulations.

The rest of this section is organized as follows. In Section 3.1, we derive an expression for AFCT of flows as a function of packet loss probability. We then consider the impact of buffer size on packet loss probability by studying two extreme cases in Sections 3.2 and 3.3, respectively: one where the mean file size is large compared to the bandwidth delay product and the other where the mean file size is small compared to the bandwidth delay products. The effect on buffer size on AFCT can be evaluated using the results derived in these sections.

3.1 The effect of loss probability on AFCT

For protocols like TCP, packet losses directly affect throughput since the window size is decreased by half whenever losses are seen. Therefore, it is rather obvious that the packet-loss probability should be kept as small as possible. With RCP, packet-losses and the rate are decoupled. So, losses do not affect the throughput directly. However, packet losses can affect the AFCT.

This can be best explained using an example. Consider a flow which has seven packets to transmit. In a network *with no loss*, RCP would transmit all the seven packets one after another while adjusting its transmission rate to match the rate dictated by the latest *ack* packet. Now consider a network in which packets can be dropped. At the end of the first transmission round, not all packets are received successfully. Packets numbered 4 and 6 are dropped by the network. However the RCP source cannot know about the dropped packets instantaneously. The RCP source comes to know of the lost packets in the following fashion. After transmitting the seventh packet, the RCP source waits for some time in order to receive *ack* packets from the receiver (in this example, by the time the RCP source transmits the seventh packet, the only packets that are unacknowledged are packets numbered 4, 6 and 7). Since packet numbered 4 and 6 were dropped from the network, the source will not receive *ack* packets for these data packets. Eventually (i.e., $2 \times RTT$ later), the RCP source assumes that these packets are lost and decides to retransmit them. Let us assume that packet numbered 6 is again dropped by the network. This time the RCP source will receive an *ack* packet for packet numbered 4, but not for packet numbered 6. The RCP source will again wait for $2 \times RTT$ and then decide to retransmit packet numbered 6. The flow is considered complete once the RCP source receives an *ack* packet for packet numbered 6. In this example, the RCP source had to transmit packets in three "rounds" to ensure that all the packets are received by the receiver.

Generalizing the above examples, an RCP source retransmits data using the following retransmission mechanism. In any "round", the RCP source transmits all of its unacknowledged packets, while adjusting its rate according to the network conditions (as indicated by the latest *ack* packet). After transmitting the last unacknowledged packet, the RCP source waits for $2 \times RTT$ in order to receive the *ack* packets. If some packets are lost, these packets would remain unacknowledged. If the RCP source does not receive an *ack* packet for a particular data packet within the waiting period, it assumes that the data packet was dropped by the network. The RCP source will retransmit all the lost packets in the subsequent "round." This process is continued until all the packets have been acknowledged. Thus, increased packet losses increase AFCT for the following two reasons: (a) Lost packets must be transmitted again which means that the same packet may be transmitted multiple times. (b) Packet loss also inserts a waiting period of $2 \times RTT$ between successive transmission rounds.

Note that the above retransmission mechanism is more naive than retransmission protocols implemented in existing versions of TCP. For example, the sender does not have to wait for a full round before retransmitting a lost packet. Thus, more sophisticated retransmission protocols will improve the file completion time by reducing the $2 \times RTT$ waiting time between rounds. In other words, from the point of view of estimating the AFCT, our retransmission assumption is worst-case, i.e., the performance of RCP can only be better than our analysis suggests if a better retransmission scheme is used. Our main result of this section, captures the trade-off between the packet-loss probability and the AFCT.

THEOREM 1. *Consider a single bottleneck link of capacity C being accessed by many competing flows. Suppose that packet losses occur with probability p, and the packet-loss events are assumed to occur independently across packets. Then, for a flow which has K packets to transmit, the average flow completion time is upper-bounded by,*

$$\bar{T}_K^p = \bar{T}^0_{\frac{K}{(1-p)}} + 2RTT \sum_{k=1}^{K} (-1)^k \left(\begin{array}{c} K \\ k \end{array} \right) \frac{p^k}{1-p^k}, \quad (2)$$

where $\bar{T}_K^p$ denotes the AFCT of a flow with K packets when the packet-loss probability at the router is p and $\bar{T}_K^0$ denotes the AFCT of a flow with K packets when the system has zero losses (i.e., RCP with large buffers).

Proof. We only provide a sketch of the proof here. For the complete proof, the interested reader is referred to [20]. The increase in the AFCT is due to two reasons:

(i) In a lossy network, the same packet could be transmitted multiple times, so that the receiver can receive it successfully. If the packet-loss probability at the link is p, then to ensure that K packets are received successfully, the source has to transmit, on average, $\frac{K}{(1-p)}$ packets. The time taken for an end user to transmit $\frac{K}{(1-p)}$ packets is $\bar{T}^0_{\frac{K}{1-p}}$. This gives the first part of (2).

(ii) As mentioned earlier, RCP transmits packets in "rounds." At the end of each round, RCP waits for $2 \times RTT$ before retransmitting the unacknowledged packets. To characterize the increase in AFCT due to these waiting periods, we have to compute the average number of "rounds" taken by RCP to transmit K packets. Let R_i be the random variable that represents the round in which packet i was successfully received by the receiver. The flow is complete if all the packets are successfully received at the receiver. Therefore, the average time spent waiting for the feedback to arrive is $2 \times RTT E[\max_{i\in\{1,K\}}\{R_i\}]$. It can be shown that this is equal to the second term in (2). ■

REMARK 1. *We consider the following example of a single link of capacity $C = 100$ Mbps with flows having $RTT = 100$ ms. Let us suppose that each packet is 1 KB long and therefore $C = 12500$ packets/sec. Flows arrive according to a Poisson process of rate λ and the file-sizes are exponentially distributed with mean $\frac{1}{\mu}$. Further let us suppose that λ and μ are chosen such that the load on the system $\rho = \frac{\lambda}{\mu} = 80$ Mbps.*

To calculate $\bar{T}^p_K$, we need to know $\bar{T}^0_K$, i.e., the AFCT of flows under RCP when the network has zero losses. According to earlier studies [10, 12], the AFCT of flows under RCP can be well approximated by the AFCT of flows under processor-sharing (PS) discipline with an extra overhead of $2 \times RTT$. This overhead is the time required to setup and tear down the connection[1]. In other words, the AFCT of a K packet flow under RCP in a zero-loss network is approximately

$$\bar{T}^0_K \approx 2 \times RTT + \frac{K}{C(1 - \frac{\rho}{C})}.$$

Using this expression for $\bar{T}^0_K$, we plot the AFCT of flows as a function of the packet-loss probability p for various values of the flow size K. To compare across different flow sizes K, we normalize $\bar{T}^p_K$ with the AFCT of the same flow under zero loss (i.e., we plot $\frac{\bar{T}^p_K}{\bar{T}^0_K}$). The results are provided in Fig. 1.

Suppose we operate the network at a packet-loss probability of $p = 0.1$. At this loss probability, the short-flows ($K = 10$) and long-flows ($K = 10,000$) do not suffer much degradation (less than 20%) in their AFCT compared to a network with zero packet loss. On the other hand, the AFCT increases more significantly for intermediate-size flows. For example, for $K = 100$, the AFCT increases by 65%. Under a Pareto distribution which is commonly proposed for Internet file sizes, conditioned on the fact that a file is not short, the mean file size is typically very large. Thus, the above increase in AFCT for medium file sizes should not significantly affect the overall AFCT. This observation is borne out by simulations presented in a later section. In summary, we expect that changes to the buffer size, which vary the loss probability in the range between 0 and 10%, will not affect the AFCT significantly. This intuition is verified later through simulations.

[1]RCP uses the standard three-way handshake used by TCP to initiate a session. Similarly, at the end of a session, the source transmits a FIN message to the receiver to tear down the connection. The time required to transmit these messages is $2 \times RTT$.

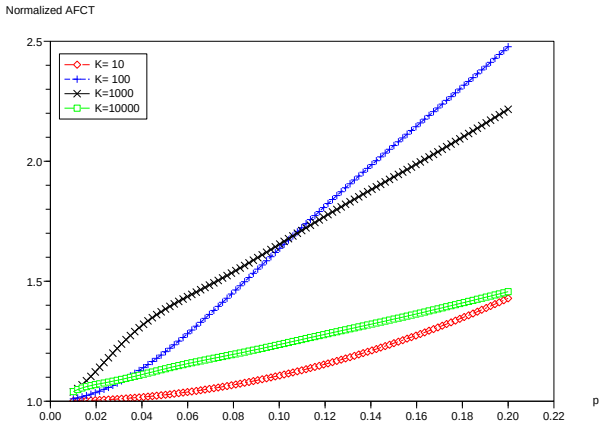

Figure 1: Normalized AFCT as a function of the packet loss probability p. Setup : $C = 100$ Mbps, RTT $= 100$ ms, $\rho = 80$ Mbps

3.2 Overflow probability in large flow-size networks

In the previous subsection, we have derived an expression for AFCT as a function of loss probability. In this subsection and the next, we relate the loss probability to the buffer size. Computing the loss probability in a queueing system is quite hard. Instead, we use overflow probability as an approximation to the loss probability, i.e., we assume that the buffer size is *infinite* and estimate the overflow probability $Prob\{Q > B\}$, where B is the buffer size. Approximating the loss probability using overflow probability is quite standard in the queueing theory literature and has been used effectively to size buffers using large-deviations techniques [16].

When the mean flow-sizes are very large ($\mu CRTT \ll 1$), the network conditions do not change rapidly. Congestion control protocols operate on time-scales much faster than the time-scales of arrival and departures. To give a clear picture of the possible situations where these results are applicable, consider a link in a metro-Ethernet network, providing services to data storage traffic. Let the capacity of the link be 100 Mbps. Since the service is typically local, the RTT will be quite small (say 10 ms). The data storage traffic consists of large file transfers, and in this case suppose that, on average, the files are about 10 MB long. For this example,

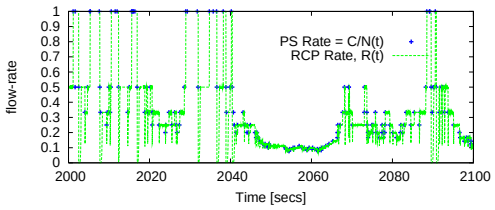

Figure 2: Time variation of the RCP rate under flow arrivals and departures. Simulation parameters: $C = 100$ **Mbps,** $\frac{1}{\mu} = 10$ **MB,** $RTT = 10$ **ms,** $\rho = 0.8$

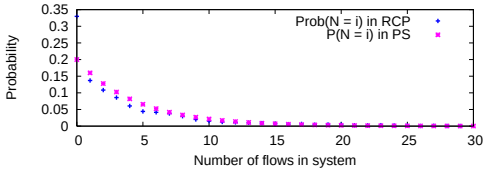

Figure 3: Distribution of the number of flows under RCP and under PS. Simulation parameters: $C = 100$ **Mbps,** $\frac{1}{\mu} = 10$ **MB,** $RTT = 10$ **ms,** $\rho = 0.8$

$\mu CRTT = 0.0125 \ll 1$.

Now, to support a load of 80%, the arrival rate $\lambda = 1 flow/sec$. Therefore, on average, it takes about 50 RTTs for the system to change its state[2] (either an arrival or a depature). In an earlier work, the convergence times of RCP has been studied for the case of N long-lived flows [4]. These results indicate that typically RCP takes about 5 to 10 RTTs to reach the equilibrium fair share (depending on the parameters α and β in (1)). In other words, when the system is varying very slowly compared to the time scale of the congestion control protocol, for most of the duration, the fair share rate under RCP is equal to the rate under processor sharing (see Fig. 2). Therefore, the statistics of the number of flows under RCP would be close to the statistics of the number of flows under processor sharing (see Fig. 3).

ASSUMPTION 1. *If $\mu CRTT \ll 1$, then the statistics of the number of flows under RCP is the same as the statistics of number of flows under processor sharing (PS). In other words,*

$$P_{RCP}\{N = i\} = P_{PS}\{N = i\} = \left(\frac{\rho}{C}\right)^i \left(1 - \frac{\rho}{C}\right). \qquad (3)$$

We now state our main result of this section.

THEOREM 2. *Consider a single-link accessed by many flows. The flows arrive into the network according to a Poisson process with rate λ. The file-sizes are assumed to be i.i.d with mean flow size $\frac{1}{\mu}$. Let N denote the number of active flows in the system. Furthermore, suppose that the file-sizes are large enough such that $\nu = \mu CRTT \ll 1$. Let $\kappa = \frac{B}{C \times RTT} < 1$. Then, the probability that the queue Q exceeds a threshold B is given by,*

$$Prob\{Q > B\} = \nu \left(\frac{\rho}{C}\right)^2 E_{RCP}\left[\left(\frac{1}{\alpha}\frac{1}{N} - \kappa\right)^+ \middle| N > 0\right] + o(\nu),$$

where $\rho = \frac{\lambda}{\mu}$ and $E_{RCP}[\cdot]$ denotes the expectation under the stationary distribution of N under RCP. Here we have used the convention that

$$(X)^+ = \begin{cases} X, & \text{if } X > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Now suppose that Assumption 1 holds. Then the above expression can be simplified to obtain,

$$P\{Q > B\} = \nu \left(\frac{\rho}{C}\right)^2 \left(\frac{1}{\alpha}\frac{C-\rho}{\rho} \log\left(1 - \frac{\rho}{C}\right)\right) - \nu \left(\frac{\rho}{C}\right)^2 \left(\kappa\left(1 - \left(\frac{\rho}{C}\right)^{N^*}\right)\right),$$

where $N^* = \dfrac{1}{\kappa\alpha}$.

Proof. We provide a sketch of the proof here. The reader is referred to [20] for more details. Since $\nu \ll 1$, flows arrive and depart on time-scales that are much slower than the time required for congestion control protocols to converge. Furthermore, buffers are required only to store packets when there are new arrivals into the system (a departure leaves spare capacity in the system and no packets are dropped when flows depart from the system). Almost zero buffering is required when the system is in steady state.

Since arrivals occur rarely, the most likely cause of buffer overflow is the arrival of a *single* new flow into the system. The packet losses that occur due to the arrival of many flows within a short duration are so small that they can be neglected in our calculation.

To calculate the packet loss probability, we need to calculate the number of extra packets that are generated during a transition event. Suppose that there are $N = n$ flows in the system at time $t = 0$, and at time $t = \delta$ a new flow arrives into the system. Assuming that the n flows were in steady-state, the packet arrival rate at the core router at time $t = \delta$ would be $\frac{(n+1)}{n}C$. Since the link is partially oversubscribed, the RCP algorithm would adjust the fair-share rate to reduce congestion. During this transition the fair-share rate maintained at the router changes from $Z = \frac{C}{n}$ to $Z = \frac{C}{n+1}$. The evolution of the fair-share rate $Z(t)$ under RCP can be approximated by the following difference equation.

$$Z((k+1)RTT) = Z(kRTT) + \alpha(C - (n+1)Z(kRTT)).$$

This difference equation can be solved using standard z-transform methods. Assuming $Z(0) = \frac{C}{n}$, the evolution of $Z(t)$ can be expressed as,

$$Z((k+1)RTT) = \frac{C}{n+1} + (1-\alpha)^k \frac{C}{n(n+1)}.$$

[2]An arrival would occur on average once in 100 RTTs. Similarly a depature is seen once in 100 RTTs as the system is stable. Therefore, the system changes its state on average, once in 50 RTTs.

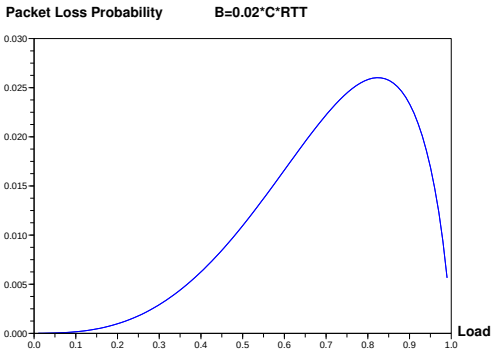

Figure 4: Loss probability as a function of the offered load: Buffer Size $= 0.02 \times CRTT$

Assuming that $Q(0) = 0$, we calculate the number of dropped packets D:

$$\begin{aligned} D &= \sum_{k=0}^{\infty} \left((n+1)Z(kRTT) - C\right) RTT - \kappa CRTT \\ &= CRTT\left(\frac{1}{n\alpha} - \kappa\right)^+ \end{aligned}$$

Therefore, the average number of dropped packets is

$$E_{RCP}[D] = CRTT\frac{\rho}{C}E_{RCP}\left[\left(\frac{1}{N\alpha} - \kappa\right)^+ \middle| N > 0\right]$$

The average inter-arrival time is $\frac{1}{\lambda}$. The average number of packets transmitted during this time would be

$$T = C\frac{1}{\lambda}$$

Therefore the fraction of lost packets is

$$\begin{aligned} Prob\{Q > B\} &= \frac{E_{RCP}[D]}{T} \\ &= \nu\left(\frac{\rho}{C}\right)^2 E_{RCP}\left[\left(\frac{1}{\alpha N} - \kappa\right)^+ \middle| N > 0\right] \end{aligned}$$

The second result of Theorem 2 is obtained by a straightforward algebraic simplification after substituting for P_{RCP} from (3). ■

We now consider the same example considered earlier in this section ($C = 100$ Mbps, $\frac{1}{\mu} = 10$ MB, RTT $= 80$ ms and $\alpha = 0.2$). The loss probability is plotted as a function of the load in Fig. 4. As demonstrated by the figure, even when the buffer size is only about $0.02 \times CRTT$, the loss probability does not suffer significantly. The maximum packet loss seen is about 2.5%.

REMARK 2. *It is interesting to note that as $\rho \to C$, the loss probability tends to zero. This may seem counter-intuitive at first (with TCP, losses dictate the throughput and since throughput decreases with the increase in the load, the loss probability should increase as the load increases), but we have found that simulations exhibit the same behavior. The reason is that buffering is used only to store packets from newly arrived flows. For existing flows, no buffering is required since their total arrival rate is equal to capacity. As the load increases, the average per-flow rate of the newly arrived flow decreases and therefore, chances of overflowing the buffer decreases. This is captured in Fig. 4.*

3.3 Overflow probability in small flow-size networks

Whenever congestion control works well (i.e., flows with large mean size), we have seen that RCP requires very small amounts of buffering at the core router. However, as mentioned earlier, Internet traffic consists of mostly short-flows which do not last long enough to react to congestion. In this section, we complement the results in the previous subsection by studying another extreme case where the mean flow-size is so small that most flows do not react to congestion information. The purpose of this exercise is to show that, even in this case, small buffers are sufficient. As in Section 3.2, we assume that the buffer sizes are *infinite* and our analysis provides an estimate of the probability that the queue size exceeds a threshold B.

Before we start studying the buffer size requirements, it is worthwhile to consider a simple example to understand the assumptions that we make to derive the overflow probability. A key observation that facilitates the analysis is that, if the mean flow size is small, then the network conditions change significantly within a single update interval. Consider a single congested router of capacity 10 Gbps and RTT 80 ms. Further suppose that the users primarily access the network for web page downloads and therefore, the mean flow size is fairly small. For the purposes of this example, let us assume that the mean flow size on this link is $\frac{1}{\mu} = 50$ KB. Further let us suppose that the update interval $d = 10$ ms. Then, if the offered load is 80%, the flow arrival rate should be about $20,000$ flows/sec. This means that on average about $1,600$ flows would arrive and depart within a single round trip time. Thus, even if the network were to estimate the fair-share rate accurately, by the time this feedback information reaches the end-host, the network conditions would have changed dramatically, making the information stale. In other words, the rate information conveyed to the source one round trip time earlier would carry little information about the present level of network congestion. Therefore, in carrying out our analysis, we make the following assumption.

ASSUMPTION 2. *The rate z at which a flow is transmitting data, is independent of the number of flows in the network N.*

Recall from Section 2 that RCP updates the rate information once every d seconds. Thus, every packet arriving to the router within the same update period will carry the same rate information. If all the flows have the same RTT, then this information would reach the end hosts at the same time. Thus, all the end-hosts will transmit at the same rate. This leads to our second assumption.

ASSUMPTION 3. *All the end users are assumed to transmit at the same rate z within a single update period.*

Suppose that flow arrivals occur according to a Poisson process with rate λ and the file-sizes are exponentially distributed. Then, the flow departure process can be described by a Poisson process with rate $\mu N(t)z$, where $N(t)$ denotes the number of active flows at time t. Thus $N(t)$ is a Markov chain and the statistics of $N(t)$ within a single update period are the same as that of a $M/M/\infty$ queue with mean arrival rate λ and mean service time $1/\mu z$.

Our main objective here is to find out how many packets arrive within an update period or in other words, to determine in the distribution of the packet arrival process within a single update period. If A_k is a random variable that denotes the total number of packet arrivals within the update period k, then

$$A_k = \int_0^d N(t)zdt.$$

Since z is a constant, the distribution of A_k is determined by the distribution of $\int_0^d N(t)dt$. We will approximate $\int_0^d N(t)dt$ by a Gaussian distribution and assume that A_k are independent. Admittedly, such assumptions are difficult to justify. The Gaussian approximation can be partly justified by heavy-traffic limit theorems for infinite server queues [25]. On the other hand, whether a time-interval of size d is sufficient for the time scaling used in heavy-traffic limit theorems to apply is difficult to judge. However, simulations indicate that our model provides the right qualitative and quantitative insight into the relationship between buffer size, loss probability and AFCT.

From the covariance function of the $M/M/\infty$ queue (see, for example, [13]), it is straightforward to compute the mean and variance of A_k as

$$E(A_k) = \rho d, \quad \text{and} \quad Var(A_k) = \frac{2\rho d}{\mu}\left(1 - \frac{1 - e^{-\mu z d}}{\mu z d}\right).$$

Since we are considering the regime where the file size is small compared to the delay-bandwidth product for each user, i.e., $1/\mu << zd$, we make the following assumption regarding A_k.

ASSUMPTION 4. *The number of packet arrivals in a measurement interval is assumed to be a Gaussian random variable with mean ρd and variance $\frac{2\rho d}{\mu}$. We also assume that the random variables representing the number of arrivals in each measurement interval are independent.*

Now we are ready to state the main result of this section.

THEOREM 3. *Consider a single-link of capacity C. The flows arrive into the network according to a Poisson process of rate λ. The file-sizes are exponentially distributed with mean $\frac{1}{\mu}$. We denote the update period be denoted by d. Suppose that Assumptions 3-4 hold. Then, the buffer size B required to ensure that the overflow probability is less than some desired threshold p_{des} is given by*

$$B > \frac{\rho}{\mu(C-\rho)} \log\left(\frac{1}{p_{des}}\right).$$

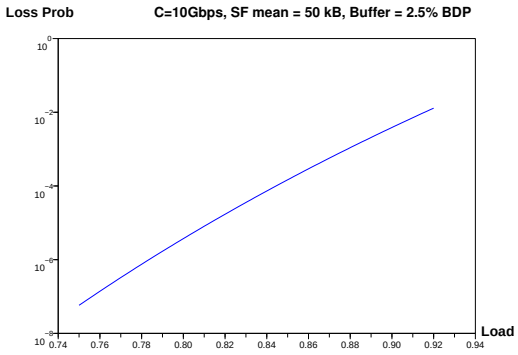

Figure 5: Packet loss probability as a function of the offered load: $C = 10$ Gbps, $RTT = 80$ ms, $\frac{1}{\mu} = 50$ KB, $B = 2500$ KB ($0.025 \times C \times RTT$)

Proof. From large deviations theory [7], one can achieve an overflow probability p_{des} with a buffer size B if

$$\frac{\Lambda(\gamma)}{\gamma} < Cd,$$

where

$$\gamma = -\frac{1}{B}\log(p_{des}), \quad \text{and} \quad \Lambda(\gamma) = \log E[e^{\gamma A_1}].$$

Since A_i is known to be a Gaussian random variable with mean ρd and variance $\frac{2\rho d}{\mu}$, the above equation can be simplified to obtain the desired result. ∎

Consider the same example considered earlier in this section ($C = 10$ Gbps, $\frac{1}{\mu} = 50$ KB and RTT $= 80$ ms). Let the buffer size be equal to 10 Gbps $\times 2$ ms. For this buffer size, the loss probability is plotted as a function of the offered load in Fig. 5. As the figure demonstrates, even with only 2 ms of buffering (it is equivalent to $0.025 \times C \times RTT$), the packet loss probability is extremely small. Even when the offered load is 90% of the capacity, the figure indicates that the loss probability is around 0.01.

4. SIMULATION RESULTS

Our simulations use $ns - 2.29$ [1] and we have augmented the RCP end-host algorithm described in [10] (code available at RCP web page [2]) with a simple retransmission scheme, which works as follows: the receiver informs the sender of the cumulative number of packets it has received so far, through the acknowledgment packets. It does not send the source any information regarding the sequence numbers of the lost packets. Thus, the source assumes that the flow is complete when the number of packets received by the receiver is equal to the flow size. This abstraction simplifies the design of the retransmission mechanism significantly and is consistent with the analytical model used earlier. At the same time, the approach is detailed enough to study the effects of small buffers on flow completion times.

We consider a single bottleneck link accessed by many flows. The mean packet size is set to be 1000 bytes. Flows arrive into the network according to a Poisson process of rate λ and

transfers a random amount of data, with mean $\frac{1}{\mu}$. The load on the system $\rho = \frac{\lambda}{\mu}$. We change the load on the system, by changing λ. The RCP parameters are chosen as $(\alpha, \beta) = (0.4, 0.5)$ in all our simulations.

We perform three sets of simulations to understand the impact of buffer size on the AFCT of RCP-controlled flows: the first set of simulations considers the case of long flows, the second set of simulations considers short flows and the third set of simulations considers a mixture of short and long flows. To simulate a mixture of file sizes with a large variance as in the Internet, we use a hyper-exponential file-size distribution, i.e., with a certain probability q the file size is exponential with a large mean $1/\mu_l$, and with probability $1-q$ the file size is exponential with a small mean $1/\mu_s$. To study the accuracy of the analytical results, we use an exponential file size of $1/\mu_l$ to simulate the case of long flows only, and an exponential file size of $1/\mu_s$ to simulate the case of short flows only.

Due to space limitations, we only present a representative simulation example for each of the three cases. More extensive simulations can be found in a technical report [20]. We note that the purpose of the simulations presented here is to show that the performance of RCP does not degrade significantly when the buffer size is reduced from the full bandwidth-delay product (BDP) to $1/20^{\text{th}}$ of the BDP. In addition, we note that RCP performs significantly better than TCP for this range of buffer sizes. This comparison between RCP and TCP is not shown here but can be observed from the simulations in [20].

In our first set of simulations, we plot the AFCT as a function of the load for three different buffer sizes: BDP, 10% of the BDP and 5% of the BDP. As the Figure 6 shows, the AFCT is not significantly affected by the buffer size. In

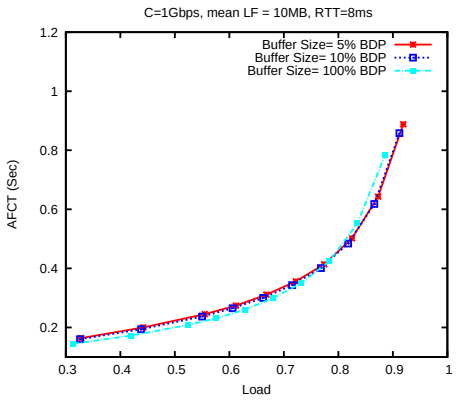

Figure 6: AFCT as a function of load for long flows

Figures 7-8, we compare the AFCT obtained from simulations with the theoretical expressions derived in the earlier sections. As can be seen from the figures, there is strong agreement between simulations and the theory.

Next, we present similar simulation results for the case of short flows in Figures 9-11. Again, the AFCT is not significantly affected by the buffer size, and the theoretical results are in agreement with the simulation results.

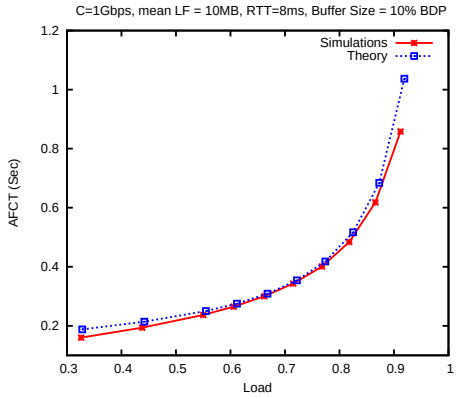

Figure 7: Comparison of the theoretical and simulation results for the AFCT of long flows when the buffer size is equal to 10% of the BDP

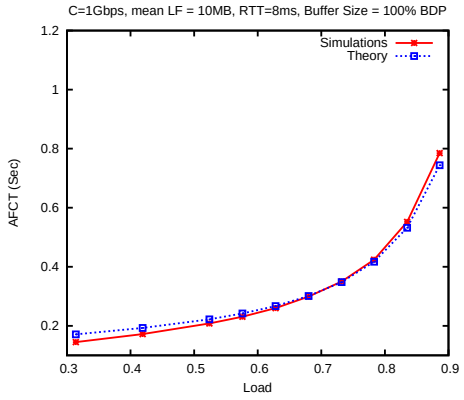

Figure 8: Comparison of the theoretical and simulation results for the AFCT of long flows when the buffer size is equal to the BDP

Finally, we present simulation results for a mixture of long and short flows. Here, since there are no theoretical results in this case, we only compare the AFCT obtained through simulations for three different cases of buffer sizes. The results are presented in Figures 12-14. The figures indicates that the difference between the AFCT with a buffer size equal to the full bandwidth-delay product and a buffer size equal to 10% of the bandwidth-delay product is less than 15%. When the buffer size is reduced to 5% of the bandwidth-delay product, the AFCT increases by approximately 25%. Based on these and other extensive simulation results, we recommend that the buffer size can be reduced to 10% of the bandwidth-delay product without significantly affecting performance.

5. CONCLUSIONS

In this paper, we have developed a simple model for RCP and studied the amount of buffering needed on RCP-enabled

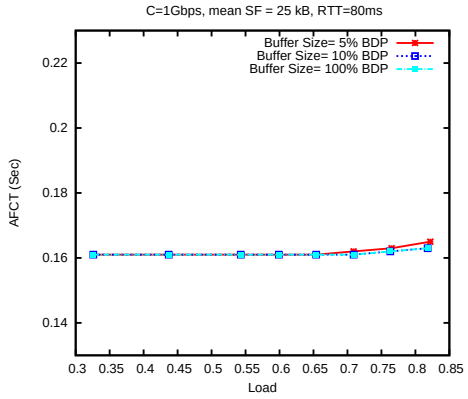

Figure 9: AFCT as a function of load for short flows

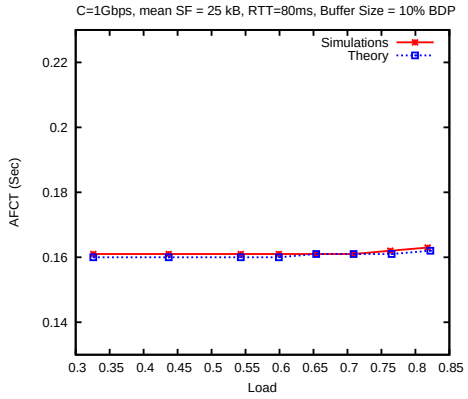

Figure 10: Comparison of the theoretical and simulation results for the AFCT of short flows when the buffer size is equal to 10% of the BDP

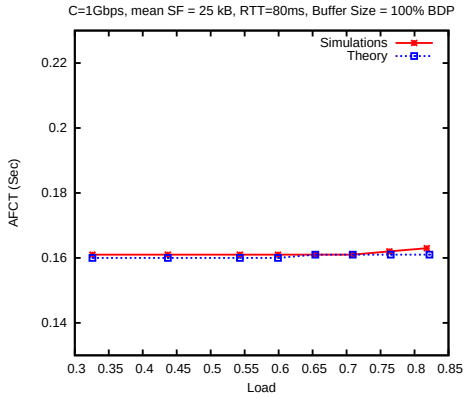

Figure 11: Comparison of the theoretical and simulation results for the AFCT of short flows when the buffer size is equal to the BDP

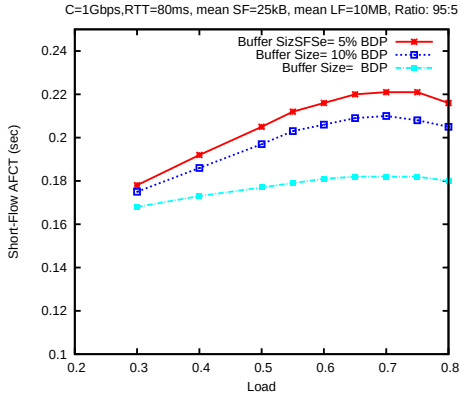

Figure 12: Comparison of the AFCT for short flows under various buffer sizes, when the network traffic consists of a hyper-exponential mixture of short and long flows

routers. Specifically, we consider scenarios where the number of flows is not sufficiently large to take advantage of statistical multiplexing to reduce buffer sizes. Rather, we focus on the benefits of explicit rate feedback in reducing buffer size requirements. We first model the impact of loss probability on AFCT. Then, we approximate loss probability by overflow probability and model the impact of buffer size on overflow probability in two separate cases: one where flow sizes are large and one where flow sizes are small. Numerical studies using our models allow us to conclude that RCP delivers good performance even with buffer sizes as small as 10% of the bandwidth delay product. We also validate our results using simulations where, in addition to the separate large and small file-size cases, we also consider networks with a mixture of file-sizes.

Prior work suggests that there are four main features of RCP that make it an appealing and practical congestion control algorithm: (i) RCP is inherently max-min fair. (ii) RCP's flow-completion times are often one to two orders of magnitude better than in TCP SACK and XCP [6], and close to what flows would have achieved if they were ideally processor shared. (iii) There is no per-flow state or per-flow queuing. (iv) The per-packet computations at RCP router are simple.

In addition, we have now established that small buffer-sizes are sufficient which makes RCP an attractive protocol for deployment in high-speed networks. As noted in Section 1, RCP can also be modified to achieves other forms of fairness such as proportional fairness which may have other desirable properties as well (see [5, 22] and references within).

Acknowledgment

Research supported in part by NSF grant CNS 07-21286 and the NSF 100x100 Clean Slate Program.

6. REFERENCES

[1] The network simulator: ns-2. Available at http://www.isi.edu/nsnam/ns.
[2] Rcp web page, 2006. http://yuba.stanford.edu/rcp.
[3] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *ACM/SIGCOMM*, 2004.

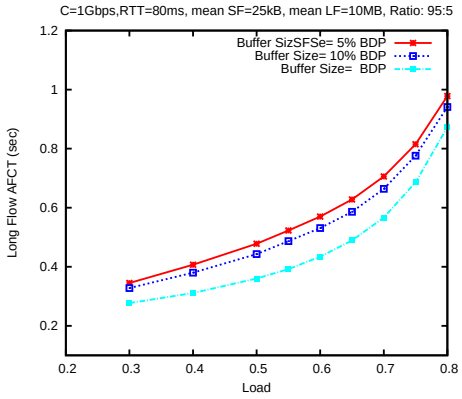

Figure 13: Comparison of AFCT for long flows under various buffer sizes when the network traffic consists of a hyper-exponential mixture of short and long flows

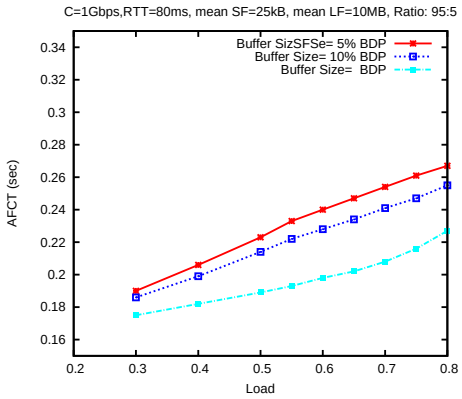

Figure 14: Comparison of overall AFCT under buffer sizes when the network traffic consists of a hyper-exponential mixture of short and long flows

[4] H. Balakrishnan, N. Dukkipati, N. McKeown, and C. Tomlin. Stability analysis of explicit congestion control protocols. *IEEE Communication Letters*, 11(10), October 2007.

[5] T. Bonald, L. Massoulie, A. Proutiere, and J. Virtamo. A queueing analysis of max-min fairness, proportional fairness and balanced fairness. *Queueing Systems*, 2006.

[6] M. Handley D. Katabi and C. Rohrs. Internet congestion control for future high bandwidth-delay product environments. In *Proceedings of ACM SIGCOMM*, 2002.

[7] G. de Veciana, G. Kesidis, and J. Walrand. Resource management in wide-area ATM networks using effective bandwidths. *IEEE Journal on Selected Areas in Communications*, 13:1081–1090, 1995.

[8] A. Dhamdhere and C. Dovrolis. Open issues in router buffer sizing. *ACM/SIGCOMM Computer Communication Review*, pages 87–92, January 2006.

[9] N. Dukkipati, G. Gibb, N. McKeown, and J. Zhu. Building a RCP (rate control protocol) test network. In *Proceedings of Hot Interconnects*, August 2007.

[10] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown. Processor sharing flows in the Internet. In *Thirteenth International Workshop on Quality of Service (IWQoS)*, 2005.

[11] N. Dukkipati and N. McKeown. Why flow-completion time is the right metric for congestion control. In *ACM SIGCOMM Computer Communication Review*, January 2006.

[12] N. Dukkipati, N. McKeown, and A. Fraser. RCP-AC: Congestion control to make flows complete quickly in any environment. In *Proceedings of the IEEE INFOCOM*, April 2006.

[13] S. G. Eick, W. A. Massey, and W. Whitt. The physics of the $M_t/G/\infty$ queue. *Operations Research*, 41:731–742, 1993.

[14] M. Enachescu, Y. Ganjali, A. Goel, T. Roughgarden, and N. McKeown. Part III: Routers with very small buffers. *ACM/SIGCOMM Computer Communication Review*, 35(3):7, July 2005.

[15] C. Fulton, S. Q. Li, and C. S. Lim. An ABR feedback control scheme with tracking. In *Proceedings of IEEE INFOCOM*, pages 806–815, 1997.

[16] A. J. Ganesh, D. Wischik, and N. O'Connell. *Big Queues*, volume 1838 of *Lecture notes in Mathematics*. Springer-Berlin, 2004.

[17] M. Handley and D. Wischik. Congestion, synchronization and buffer size in backbone routers, 2006. preprint.

[18] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, and B. Vandalore. The ERICA switch algorithm for ABR traffic management in ATM networks, 1997. http://www.cis.ohio-state.edu/ jain/papers.

[19] F. Kelly, G. Raina, and T. Voice. Stability and fairness of explicit congestion control with small buffers. *SIGCOMM Computer Communication Review*, pages 51–62, 2008.

[20] A. Lakshmikantha, N. Dukkipati, R. Srikant, N. McKeown, and C.L. Beck. Performance analysis of RCP, 2006. Technical Report, available at http://www.ifp.uiuc.edu/~lkshmknt/rcp.pdf.

[21] A. Lakshmikantha, R. Srikant, and C. Beck. Impact of file arrivals and departures on buffer sizing in core routers. In *Proceedings of IEEE Infocom*, 2008.

[22] L. Massoulie. Structural properties of proportional fairness: stability and insensitivity. *The Annals of Applied Probability*, 2007.

[23] G. Raina, D. Towsley, and D. Wischik. Part II: Control theory for buffer sizing. *ACM/SIGCOMM Computer Communication Review*, pages 79–82, July 2005.

[24] C. H. Tai, J. Zhu, and N. Dukkipati. Making large scale deployment of RCP practical for real networks. In *IEEE INFOCOM Mini-conference*, 2008.

[25] W. Whitt. On the heavy-traffic limit theorems for $GI/G/\infty$ queues. *Advances in Applied Probability*, 14:171–190, 1982.

[26] D. Wischik and N. McKeown. Part I: Buffer sizes for core routers. *ACM/SIGCOMM Computer Communication Review*, pages 75–78, July 2005.