Minimizing the Duration of Flows
Stanford HPNG Technical Report TR04-HPNG-061604

Nandita Dukkipati\(^1\), Masayoshi Kobayashi\(^2\), Rui Zhang-Shen\(^3\), Nick McKeown\(^4\)
\(^1\)Electrical Engineering Department, Stanford University
\{nanditad, rzhang, nickm\}@stanford.edu
\(^2\) System Platforms Research Laboratories, NEC Corporation
m-kobayashi@eo.jp.nec.com

Abstract—Congestion control algorithms are usually designed to maximize network utilization, while ensuring fairness among flows. While desirable, these goals are network-centric, and don’t necessarily best serve the user. Our goal is to design a user-centric congestion control algorithm that minimizes the expected duration of each flow (e.g. minimize the time to download a web page). We first observe that by focusing on flow duration, flows can often be one, or even two, orders of magnitude shorter than with conventional congestion control algorithms, such as TCP Reno. Second, we demonstrate that in a network with heavy-tailed flows (such as the Internet), each router can pick a single rate for all flows passing through it, and come close to minimizing the expected flow duration. And so finally, we propose a congestion control algorithm, RCP (Rate Control Protocol) that tries to minimize the expected duration of each flow. We demonstrate that under varying link capacities, RTTs, network loads and topologies the flow durations in RCP are close to the minimum achievable. Stability analysis indicates that RCP is stable independent of link capacities, RTTs and network load for a broad range of protocol parameters.

Index Terms—Congestion Control, Minimizing Delay, Control Theory, Simulations

I. INTRODUCTION

There have been many, many papers written about congestion control for networks. The topic has been studied extensively in theory ([11], [2], [3], [4], [5]), for ATM networks ([6], [7], [8], [9], [10]) and for TCP/IP and the Internet ([11], [12], [13]). The most common metrics used for a congestion control algorithm are utilization of the bottleneck link, and fairness among the flows. Maximizing bottleneck link utilization seems like a worthwhile goal as it maximizes the forward progress of data over the congested link; particularly if this can be done while maintaining fairness. Many interesting and successful schemes have been proposed to achieve these goals; from the simple TCP Reno [14], to the enhanced Explicit Control Protocol (XCP) [15] to the complex schemes in ATM [8]. The problem is that these protocols favor network throughput (which is good for the network operator) over flow duration (which the user would like to minimize).

Our goal is to design a congestion control algorithm that: (1) Minimizes the expected duration of flows, and (2) Is provably stable under a very broad class of conditions. We want to minimize the expected duration of a flow because this is how users experience network performance. When a user downloads a web page or file, s/he wants the download to complete in the shortest time.

To demonstrate that existing schemes don’t minimize flow duration, Figure 1 shows a simple example. The figure shows the time evolution at the receiver of a flow using TCP and XCP. In theory, the flow could complete in about one and a half round trip times (as we will see later), but takes eight times longer with TCP, and 40 times longer with XCP. There are several reasons for the long duration of flows with TCP. First, is “slow-start”, which tries to determine the flow’s fair-share rate over several round-trip times. Second, once a flow has reached the “congestion-
avoidance” mode, TCP’s additive increase adapts slowly. While this was a deliberate choice to stabilize TCP, it has the effect of increasing flow duration. We’ll see later that there are other stable congestion control algorithms that don’t require additive increase. A third reason TCP flows are so long is because of buffer occupancy. TCP deliberately fills the buffer at the bottleneck, so as to obtain feedback when packets are dropped. Extra buffers mean extra delay, which add to the duration of a flow.

Our approach is very different from TCP. In our work we want to know if there is a rate that the source can pick – with very limited information from the routers – that will minimize the duration of each flow.

II. IS THERE A RATE THAT WILL MINIMIZE FLOW DURATION?

Before approaching the design of a new congestion control algorithm, we’ll consider a hypothetical but interesting question:

Is there a rate that a source should pick, so as to minimize flow-duration?

The answer is well known for the case of a path with a single router: a Shortest Remaining Processing Time (SRPT) scheduler at the router would yield the shortest expected flow duration [17]. If every source knew the number of bytes in all the flows that pass through the same buffer, they could assign rates to flows so as to emulate SRPT. This is impractical for several reasons; most notably the amount of information each source needs to have about other flows from other sources elsewhere in the network, and the complexity of exchanging information and calculating rates. Furthermore, even if it was possible to emulate SRPT, it is not necessarily the best scheduling policy for a network of links [16]. In fact, in general, the scheduling policy to minimize flow duration in a network is not known.

Let’s consider a slight variation on the problem. Imagine that each router along a path could pick a rate based on current network conditions, and communicate it back to the source. The source picks the minimum rate picked by the routers along the path of the flow, and sends at that rate for the duration of the flow. Taking this one step further, we’re going to assume that a router must pick the same rate for all flows passing through it. It can change the rate over time; but at any one instant, the router gives the same rate to all newly starting flows.

If each router is to pick a single rate for all flows, what rate should it choose? And is there a rate that will minimize flow duration? It turns out that there isn’t in general; but, when flow lengths are heavy-tailed (which they are in the Internet), there is an optimal rate.

A. Intuition

Let’s see why there is an optimal rate. Define the flow duration, $\tau$, to be the time from when the first packet of a flow enters the network to when the last packet of the flow leaves the network, i.e.,

$$\tau = \frac{L}{R} + d_L(R)$$

where $L$ is the flow size, $R$ is the flow rate, and $d_L(R)$ is the propagation plus queueing delay of the flow’s last packet (as a function of $R$). We want to minimize flow-duration, and so we want to find $R$ that minimizes $E[\tau]$.

Notice that $\frac{L}{R}$ (the flow transmission time) decreases as $R$ increases, whereas $d_L(R)$ (the fixed propagation delay plus the variable queueing delay) increases with $R$. It all depends on whether $d_L(R)$ increases faster, or slower, than $\frac{L}{R}$ decreases with $R$. Clearly, $d_L(R)$ depends on the flow size distribution. The more deterministic the flow size, the lower the queueing delay, and so the slower $d_L(R)$ will grow with $R$. For example, if all flows are the same size, and that new flows arrive as a Poisson process to a router (which can be modeled as a single server FCFS queue with capacity $C$), then as $R \to \infty$ the flows become single entities and the router becomes an M/D/1 queue. Since all the flows are the same size, FCFS is equivalent to the optimal SRPT discipline. In other words, if flows are fixed size, the routers should not limit their rate; it should set
Complementary distribution function exponentially distributed: they have a heavy-tailed distribution, and are modeled well by the Pareto distribution with the length of simulation. For example, when the flow sizes are Pareto with the shape parameter \(1 < \alpha \leq 2\), the mean flow size is finite while the second moment is infinite, and the expected flow duration is infinite.\(^3\)

The key point here is that the plot is always ‘U’ shaped, and so always has a minimum, corresponding to the optimal rate. We can represent the ‘U’ curve analytically; see Appendix I for the details. Our approach is to define a simple adaptive rate control algorithm that will always converge on the minimum value.

We can summarize our findings so far with the following conjecture:

**Conjecture 1** When flow sizes have a bounded Pareto distribution, there exists a unique rate that minimizes the expected flow duration, and hence minimizes the user-response time.

We also conjecture that there is an optimal rate for all heavy-tailed distributions, because queue lengths will also be heavy-tailed [21], and so it is true for the Internet in general. Ideally, this is the rate a router should give to flows passing through it.

In what follows, we will describe an algorithm called RCP for routers to pick the rate of a flow.

### III. RCP: An Algorithm to Minimize Flow Durations

#### A. Picking the Flow Rate

The routers could find the optimal rate by measuring the distribution of flow-sizes and the offered load, and then numerically determine the values in Figure 3, and then solve to find the optimal sending rate. But this probably doesn’t make sense. First, it involves a lot of complicated processing by the routers, which are already burdened enough by the tasks of forwarding packets and many other.

\(^3\)Of course in simulations which run for a finite amount of time, the second moment of file size is not infinite. The value of \(E[T]\) increases with the length of simulation. Substituting \(E[T]\) for \(EL^2\) in Equation (2) gives the asymptotic value for large \(R\), shown in Figure 3.
features. Second, by the time the router measures the flow-size distribution, traffic intensity and other parameters needed to calculate the optimal rate, they will most likely have changed. So, we propose that the routers have an adaptive algorithm that updates the rate assigned to the flows such that the mean flow duration is minimized, and tracks it as link conditions change. RCP is a particular heuristic that is designed to minimize the flow duration. It has two main characteristics that makes it simple and practical:

1) The flow rate is picked by the routers based on very little information such as the queue occupancy and the aggregate input traffic rate.
2) Each router assigns a single rate for all flows passing through it.

A key observation from the 'U' curve in Section II helped us design such an adaptive algorithm:

- Our simulations show that at the point when flow duration is minimized the average queue occupancy is small and the link utilization is high.

Appendix II has example simulation plots illustrating the above observation. RCP specifically tries to keep buffer occupancy low and link utilization high, and gives all flows the same rate. This is also what an ideal processor sharing scheduler would do. It turns out that the design choice in RCP to assign rates so that flows are close to processor sharing. If so, the mean flow delay will be close to optimal, as observed in 'U' curve. For example, if we had perfect information on the number of ongoing flows at time t, and there was no feedback delay then the rate assignment algorithm would simply be:

\[ R(t) = \frac{C}{N(t)} \]

where \( R(t) \) is the rate assigned by the router at time t and C is the link capacity. But, the router does not know N(t) and it is complicated to keep track of it. Even if it could, there is a feedback delay and so by the time \( R(t) \) reaches the source, N(t) would have changed. So instead, RCP uses an adaptive algorithm to approximate Processor Sharing (PS) in the presence of feedback delays, without any knowledge of the number of flows.

**B. The Algorithm**

The basic RCP algorithm operates as follows.

1) Every router maintains a single value, \( R(t) \) that it will give to all flows. It updates the value approximately once per RTT.

2) Every packet header carries a rate field, \( R_p \) and an RTT field, \( RTT_p \). When transmitted by the source, \( R_p = \infty \) and \( RTT_p \) is the source’s current estimate of the RTT for the flow. When a router receives a packet, it uses \( RTT_p \) to update its moving average of the RTT of flows passing through it, \( d_0 \). If \( R(t) \) at a router is smaller than \( R_p \), then \( R_p \leftarrow R(t) \); otherwise it is unchanged. The destination copies \( R_p \) into the acknowledgment packets, so as to notify the source.

3) Each router periodically updates its local \( R(t) \) value according to Equation (3) below.

The rate update equation is based on the following idea:

\[ R(t) = R(t - d_0) + \frac{[\alpha(C - y(t)) - \beta q(t)]}{N(t)} \]  

(3)

where \( d_0 \) is a moving average of the RTT measured across all flows, \( R(t - d_0) \) is the last updated rate, \( C \) is the link capacity, \( y(t) \) is the measured input traffic rate during the last update interval \( d_0 \) in this case, \( q(t) \) is the instantaneous queue size, \( N(t) \) is the router’s estimate of the number of ongoing flows (i.e. number of flows actively sending traffic) at time \( t \) and \( \alpha, \beta \) are parameters chosen for stability and performance.

The central idea of the update equation is: If there is spare capacity available, i.e., \( (C - y(t)) > 0 \), then the flow rate is increased by equally dividing this spare capacity among all flows. Similarly, if \( (C - y(t)) < 0 \), then the link is oversubscribed and the flow rate is decreased. Further, the flow rate is also decreased if any queue has built up. The bandwidth needed to drain the queue within an RTT is \( \frac{q(t)}{d_0} \). The expression \( \alpha(C - y(t)) - \beta \frac{q(t)}{d_0} \) is the desired aggregate change in traffic in the next control interval, and dividing this expression by \( N(t) \) gives the change in traffic rate needed per flow.

RCP doesn’t exactly use the equations above, for two reasons. First, the router can’t directly measure the number of ongoing flows, \( N(t) \), and so estimates it as \( \hat{N}(t) = \frac{N(t - d_0)}{R(t - d_0)} \). Second, we would like to choose how often \( R(t) \) is updated – perhaps choose it to be faster than once per RTT like once every fixed control period, \( T \). The desired aggregate change in traffic over one average RTT is \( \alpha(C - y(t)) - \beta \frac{q(t)}{d_0} \), and to update the rate more often than once per RTT, we scale this aggregate change by \( T/d_0 \). And, \( \hat{N}(t) = C/R(t - T) \). Then the equation becomes:

\[ R(t) = R(t - T)[1 + \frac{T}{d_0} \frac{\alpha(C - y(t)) - \beta \frac{q(t)}{d_0}}{C}] \]

(4)

\(^{3}\)The update interval is actually \( \min(T, d_0) \) since we want it to be at least equal to RTT.
1) How good is the estimate $C/R$?: When the router updates the rate, it knows precisely the spare capacity and the queue size it needs to drain. So the accuracy of the algorithm depends on how well $C/R$ estimates $N(t)$. We will see in this section that the estimate is quite accurate.

In the simplest scenario with only long-lived flows, $C/R$ converges to the correct number of flows, $N$. An example is shown in Figure 19 where 20 flows start at time $t = 0$ and 20 more flows start at time 40, and 20 flows complete at time 100. In each case, $C/R$ converges to $N$. The values of $\alpha$ and $\beta$ only affect the rate of convergence; we’ll examine the stability region for $\alpha$ and $\beta$ in a later section.

If flows arrive as a Poisson process, and the flow sizes are Pareto, then we have found that $C/R$ is still a good estimate of the number of active flows. It is a slightly noisy estimate since flows arrive and depart quickly and therefore $N(t)$ itself is changing rapidly. An example of this case is shown in Figure 4.

But, what if some of the flows are bottlenecked elsewhere and cannot send in traffic at rate $R$. For example if $N_1$ flows are bottlenecked at a certain node and $N_2$ flows are bottlenecked elsewhere and are arriving at rate, $R_2$, less than their fair share i.e. $R_2 < C/(N_1 + N_2)$. In this case, the rate $R$ in RCP will be such that $R = C/N_1 + R_2 N_2$. In other words, RCP achieves max-min fairness. An example is shown in Figure 5. From time, $t = 40$ to 100, $N_1 = 8$ flows (Group A) and $N_2 = 4$ flows (Group B) share the bottleneck of 400 Mbps (Link C). Group B flows are bottlenecked at Link B of 80 Mbps. As seen in the bottom figure, the flows achieve their max-min fair rates. In this case, $C/R$ is an estimate of $N_1 + \theta N_2$ where $\theta = R_2/R$. This is exactly what is desired if we want max-min fairness.

It is perhaps surprising that a simple rate equation as Equation 4 which uses minimal information at the router leads to max-min fairness among flows.

C. RCP for the Internet

This is an outline of how RCP can be implemented in the Internet. We assume that – as with TCP – flows continue to have the connection set up phase to establish state at both ends of the connection, which allows the starting rate information to be piggy-backed on the SYN and SYN-ACK messages. This is very important for short-lived flows, which last (or could last) close to one RTT. Current feedback-based algorithms don’t work well for short-lived flows, yet most flows in the Internet are of this type [18]. An example of the RCP startup mechanism is illustrated in Figure 6. The SYN message sent by the source indicates the rate at which it wants to send the flow (which could be infinite). As detailed in the last section, each router maintains a single rate, $R(t)$, that it assigns to all flows. As the message passes through the network, if the current rate $R(t)$ at a router is lower than the value in the SYN packet, the router overwrites it. When the SYN packet reaches its destination, it has the lowest value corresponding to the most congested link along the path. This value is sent back to the source in the SYN-ACK message to set the starting rate. When the flows last longer than an RTT then they are periodically and explicitly told a new rate by the network. This rate is piggy-backed on the data and the ACK messages.

IV. RCP Performance

A. Simulation Setup

In this section we study RCP’s performance through simulations. Simulations are done with the network simulator ns-2 [26] (Version 2.26) augmented with RCP end-host and queue modules.

The main performance metric that we are interested in is the average flow completion time (AFCT). Flow com-
Completion time (FCT) is defined as the time interval between the sender sending the SYN packet and the receiver receiving the last packet of the flow, i.e., \( \text{FCT} = 1 \text{ RTT} \) for the connection set-up + duration of the data transfer. We will call the base RTT of a flow i.e., RTT without queuing delay, as round-trip propagation delay abbreviated as RTPD. AFCT is the average of FCT over all flows (or a subset of flows based on flow sizes as will be mentioned where necessary) during the simulation run. Note that the minimum possible value of AFCT is \( 1.5 \text{ RTPD} + \frac{E[L]}{C} \).

This is because, ignoring the queueing delay the minimum FCT for any flow of size \( L \) is: \( 1 \text{ RTT} \) for SYN/SYN-ACK and \( (1/2 \text{ RTPD} + L/C) \) for the data transfer. Since this is easy to compute in all the simulation setups, we will use this as the reference for the best achievable AFCT; henceforth we will refer to this value as OPT (for optimal). We will also use the terms AFCT and delay interchangeably. As a secondary measure, we are also interested in the average queue size at the router.

In all simulations, we assume that the queue size at a router is set to the product of the outgoing link capacity and the maximum RTPD among the flows passing through the link [20]. We also assume that packets are dropped from the tail of the queue. In all cases, simulations are run for at least 120,000 flows and 600 RTPDs so as to reach equilibrium. We note here that in all the simulations we did so far, we did not observe any packet drops in RCP or XCP. There are packet drops in TCP.

Unless mentioned otherwise, in all simulations in this section flows arrive as a Poisson process with rate \( \lambda \) and their sizes are Pareto distributed with mean, \( E[L] = 2500 \) bytes (25 packets), and shape parameter of 1.2 [19]. The load on a link, \( \rho \), is defined as \( \rho = \lambda E[L] / C \), where \( C \) is the link capacity. \( C, \rho \) and RTPD will vary with the simulation setup.

Equation. (4) is the rate update equation used in the RCP router. The RCP parameters are: Control period,  

\[
T = \min(10\text{ms}, \text{RTT}) \quad \text{and} \quad \alpha = 0.1, \beta = 1.0.
\]

For TCP, we used TCP Reno module in ns-2 with an initial window size of two packets. The ns-2 implementation of XCP (Version 1.1) was obtained from the author's website, and the parameters are set as in the paper [15].

**B. Comparison with the 'U' curve**

In all our simulations, the RCP rate update Equation. (4) achieves an average flow duration at least as small as the optimal delay in the 'U' curve. An example is shown in Figure 7, which is representative of the many simulations we did. The horizontal dotted line shows the average flow duration of RCP. RCP can, in some cases, have a lower AFCT than the 'U' curve because the rate of a long-lived flow is changed periodically when \( R(t) \) is updated. The 'U' curve is from the case when the source sends at a constant rate for the duration of the flow.

**C. Single bottleneck topology**

In this section we study the performance of RCP in the case when there is a single bottleneck link in the network.

1) Average Flow Completion Time vs. Flow Size: In this section we will compare AFCT against flow size for the following two setups. These setups are chosen to represent high bandwidth-delay product \( (C \times \text{RTPD}) \) networks, since this is the scenario that often differentiates the performance of protocols.

- Setup 1: \( C = 2.4 \text{ Gbps}, \text{RTPD} = 100 \text{ ms}, \rho = 0.9 \)

AFCT is plotted against flow size in the top two figures of Figure 8. In this case AFCT is the average flow completion time corresponding to the particular flow sizes on the x-axis. In these graphs, except for a few points, the delay in RCP is always lower than for TCP or XCP. For flows up to 2000 pkts, RCP delay is close to the OPT line, TCP delay is 4 times higher than in RCP, and XCP delay is as much as 30 times higher for flows around 2000 pkts. Note the logscale of y-axis.
Fig. 8. AFCT for different flow sizes when $C = 2.4$ Gb/s, RTPD=0.1s, and $\rho = 0.9$. The top plot shows the AFCT for flow sizes 0 to 2000 pkts; the middle plot shows the AFCT for flow sizes 2000 to $10^4$ pkts; the bottom plot shows the maximum flow completion time among all flows of the particular size.

With longer flows (> 2000 pkts), the ratio of XCP and RCP delay still remains around 30, while TCP and RCP are similar. TCP did not complete several of the long flows by the simulation end time, and the delay achieved by some of the flows in TCP is partly at the expense of the unfinished flows. This is also true with XCP. So, within a fixed simulation time, not only is RCP better for the flows that completed, but it also finished more work than TCP and XCP.

The third graph in Figure 8 shows the maximum delay for a given flow size. For RCP and XCP, the maximum is close the average, suggesting low variance. TCP delays have high variance, often ten times the mean. With all flow sizes, the maximum delay for RCP is better for the flows that completed, but it also finished more work than TCP and XCP.

- Setup 2: $C = 150$ Mbps, RTPD = 1.6 s, $\rho = 0.9$

AFCT is plotted against flow size in the top two graphs of Figure 9. Once again, the AFCT for RCP is always lower than in TCP and XCP. In general, RCP’s delay is close to the OPT line for flow sizes up to about 2000 pkts. The longer flows take multiple RTPDs. TCP’s delay is about 5 times higher than RCP, while that in XCP is 20 times higher.

The third figure shows the maximum delay for all three algorithms, and again TCP has the highest delay variance. RCP’s mean and maximum are close.

The results above are representative of the large number of simulations we performed; Appendix III has more examples.

Let’s see what it is about these protocols that give such different delays.

**RCP vs. TCP:** Notice that in both figures, 8 and 9, the TCP delay for most flows follows the Slow-start curve. The delay in TCP slow-start for a flow of size, $L$, is
Fig. 10. Time evolution of the sequence numbers of two flows under TCP, XCP and RCP, chosen from the simulation set up of Figure 8. The flow size in the top plot is 230 pkts, and in the bottom plot is 3600 pkts.

\[ \log_2(L + 1) + 1/2 \times RTPD + L/C \] (excluding the queueing delay). With RCP the same flows get a jump-start because the router’s set a higher initial rate. Hence their delay is close to the best achievable OPT line. This is very clear from the time evolution of a typical flow, as shown in Figure 10 (top plot).

Next, consider the TCP flows which deviate from the Slow-start curve. These flows experienced at least one packet drop in their lifetime and entered the additive increase, multiplicative decrease (AIMD) phase. Once a flow is in the AIMD phase, it is slow in catching up with any spare capacity and therefore lasts longer than it needs to. RCP on the hand is quick to catch up with any spare capacity available and flows finish sooner. An example of the time evolution of a flow is shown in Figure 10 (bottom plot).

**RCP vs. XCP:** The time evolution of XCP for two sample flows is shown in Figure 10. XCP is slow in giving bandwidth to the flows, giving a small rate to newly starting flows. It gradually reduces the window sizes of existing flows and increases the window sizes of the new flows, making sure there is no bandwidth over subscription. It takes multiple RTTs for most flows to reach their fair share rate (which is changing as new flows arrive). Many flows complete before they reach their fair share rate. In general, XCP stretches the flows over multiple RTPDs, to avoid over subscribing the link, and so keep buffer occupancy low. This is at odds with our goal of short AFCT.

\[ \text{AFCT} = \frac{\text{Normalized Avg. Queue Size}}{\text{Link Capacity}} \]

On the other hand, RCP tries to give the equilibrium rate to every flow based on the information it has so far, at the expense of temporary bandwidth over subscription.

2) When link capacity increases: As we increase the link capacity, the bandwidth-delay product increases, which means flows can potentially complete in fewer RTTs. In particular, more flows could finish within the optimal, OPT.

The top two plots in Figure 11 show the AFCT for flows with size < 500 pkts, the second plot shows AFCT of flows with size ≥ 500 pkts. The last plot shows the normalized average queue size.

Fig. 11. The comparison of RCP, TCP and XCP under different capacities; RTPD = 0.1s and \( p = 0.9 \). The first plot shows AFCT of flows with size < 500 pkts, the second plot shows AFCT of flows with size ≥ 500 pkts. The last plot shows the normalized average queue size.

4RTPD and load are fixed at 100 ms and 0.9 respectively for all simulations.
flows with more than 500 pkts. The AFCT of TCP is twice that of RCP for flows with fewer than 500 pkts and about 6 times more for flows with more than 500 pkts.

Figure 11 also shows the normalized queue sizes i.e. average queue size divided by the product $C \times \text{RTPD}$. Detailed plots of the absolute value of queue sizes are in [23]. The normalized queue size of RCP is 2% of the bandwidth-delay product for all the link capacities and is smaller than that of TCP. XCP always maintains a very small queue size.

3) When Round Trip Propagation Delay increases: Similarly, as we increase the path length, the bandwidth-delay product increases, which means flows can potentially complete in fewer RTTs.

Figure 12 shows the performance for different RTPDs ranging from 10 ms to 1.6 s, with a fixed $C = 150 \text{ Mb/s}$ and $\rho = 0.9$.

The top two figures show the AFCT for flows with fewer and more than 500 packets respectively. Again, RCP flow duration is lower than for TCP and XCP, and follows the OPT line closely for all RTPDs. For flows with less than 500 pkts the AFCT in TCP and XCP is about 2.5 times higher than for RCP. And for flows with more than 500 pkts, the AFCT in TCP is 4 times higher than for RCP, while for XCP it is 13 times higher.

The results reinforce that for high $C \times \text{RTPD}$ networks, RCP succeeds in fitting the majority of the flows into the one-way pipe from sender to receiver without increasing the queuing delay and achieves close to the minimum possible AFCT.

4) As load increases: The results so far were for the case when average load is at or below 90%. Figure 13 shows the AFCT of RCP. We have found that as the load increases, the disparity in the AFCT for RCP and TCP, XCP increases. Figure 13 illustrates this fact. Figure 13 also shows the average normalized queue sizes under RCP, XCP and TCP.

5) Flows with different round-trip times: All three congestion control schemes depend on feedback to adjust the window size and/or sending rate. If different flows have shorter round trip times, we can expect them to benefit at the expense of others.

To explore this effect, we simulated flows that share a common bottleneck link, but with different RTPDs. The round-trip delay of the common bottleneck link is 0.01 s and its capacity is 640Mb/s. Arriving flows are classified
Fig. 14. Comparison of RCP, TCP and XCP when flows with different RTPDs coexist on a single bottleneck of $C = 0.64$ Gb/s. RTPD of flows varies from 0.02s to 0.2s. The upper figure is the AFCT of flows with RTPD = 0.02s and the lower figure shows the AFCT for flows with RTPD = 0.2s.

Fig. 15. Comparison of RCP, TCP and XCP when flows with different RTPDs coexist on a single bottleneck of $C = 0.64$ Gb/s. RTPD of flows vary from 0.02s to 0.2s. The top figure is the AFCT of flows with flow size $\leq 2000$ pkts and the bottom figure shows the AFCT for flows with size $< 2000$ pkts.

Fig. 16. Multiple bottleneck topology. Flow group 0 traverses all the bottleneck links and the other group of flows (group 1,..,n-1) use only one bottleneck link.

Fig. 17. RCP always achieves a delay close to the OPT irrespective of the number of hops.
TCP’s delay is 3 times higher than RCP for flows with fewer than 500 pkts and is about 6 times higher for flows with more than 500 pkts. The performance of TCP degrades as the number of hops increase, in both cases. XCP’s performance is consistent with the increase in the number of hops. Its delay is about 3 times higher than RCP for flows with fewer than 500 pkts and about 13 times higher for larger flows.

The detailed plots of AFCT vs. flow-size as the number of bottlenecks increase are shown in Figure 18. Notice how the performance of TCP degrades as the number of bottlenecks increase; the AFCT of TCP gets farther away from the Slow-Start curve as the flow passes through more bottlenecks. This is because the probability of a loss increases as a flow traverses through multiple bottlenecks, and hence having a greater chance of entering into AIMD phase.

E. Long-lived flows

Although RCP was primarily designed to minimize flow durations, it also works well in scenarios where sustained throughput is more important. An example of this would be long-lived and bulky data transfers.

As an example consider a single link bottleneck with $C = 150$ Mb/s and the flows have RTPDs uniformly distributed from 20-200ms. 20 flows start at $t = 0$, then 20 more flows arrive at $t = 40s$, and 20 flows finish at $t = 100s$. Figure 19 show the time evolution of the rate factor $\gamma(t) = R(t)/C$ and the measured utilization at the bottleneck link. In equilibrium, the value of $C/R$ exactly equals $N$ that is, RCP shares the bottleneck link fairly in a very precise way and the measured utilization at the bottleneck is almost 100%.

V. Stability Analysis

A desirable characteristic of any congestion control scheme is that it is stable, in the sense that — under equilibrium load conditions — it converges to some desirable operating behavior. Significant research has lead to an understanding of the stability region for TCP and other algorithms [1], [2], [5].

In this section we derive the stability region of RCP for long-lived flows. We find that we can choose RCP’s parameters, so that the network is stable independent of the link capacity, number of flows and round trip delay.
A. RCP System Equations

The RCP system is described by the following coupled nonlinear differential equations:

\[
\dot{R}(t) = R(t - T) \left( \frac{\alpha(C - y(t)) - \beta y(t)}{Cd(t)} \right) \tag{5}
\]

\[
y(t) = NR(t - d_0) \tag{6}
\]

\[
d(t) = d_0 + \frac{q(t)}{C} \tag{7}
\]

\[
\dot{q}(t) = \begin{cases} y(t) - C & \text{if } q(t) > 0 \\ \max[y(t) - C, 0] & \text{if } q(t) = 0 \end{cases} \tag{8}
\]

where \(d(t)\) is the RTT of the flows at \(t\). The rest of the symbols are as we saw in Section III-B.

B. Equilibrium and Linearization

Taking \((R, q)\) as the state, the equilibrium point \((R_e, q_e)\) is defined by \(\dot{R} = 0\) and \(\dot{q} = 0\). So we get \((R_e, q_e) = \left( \frac{C}{N}, 0 \right)\). Because the rate update equation (5) is non-linear, we’ll first linearize the RCP rate update equation around the equilibrium point. Note that Equation (8) for the queue evolution is nonlinear as well. This equation has a discontinuity at the equilibrium point, making it hard to deal with the nonlinearity. For now we will ignore this non-linearity and approximate the queue evolution as:

\[
\dot{q}(t) = y(t) - C
\]

We will come back to this nonlinearity later. The details of the linearization steps are in Appendix IV, and lead to the following linearized equations:

\[
\delta q(t) = \frac{\alpha}{d_0} \delta R(t - d_0) \tag{9}
\]

\[
\delta \dot{R}(t) = -\frac{\alpha}{d_0} \delta R(t - d_0) - \frac{\beta}{Nd_0^2} \delta q(t)
\]

where \(\delta R \doteq R - R_e\) and \(\delta q \doteq q - q_e\) are perturbations around the equilibrium point. Taking Laplace transform of the linearized equations gives:

\[
s \delta R(s) = -\frac{\alpha}{d_0} e^{-sd_0} \delta R(s) - \frac{\beta}{Nd_0^2} \delta q(s)
\]

\[
s \delta q(s) = Ne^{-sd_0} \delta R(s)
\]

The block diagram of this feedback control system is shown in Figure 20.

C. Bode Plot Analysis

We will now obtain the stability region of our system via the Bode plot analysis. From Figure 20, the open loop transfer function of the RCP system is:

\[
G(s) = e^{-sd_0} \frac{\alpha s d_0 + \beta}{s^2 d_0^2} \tag{10}
\]

We can already see that the variables \(N\) and \(C\) do not appear above in the transfer function, which means that the system stability is independent of the number of flows and link capacity. Let us then see if there is a region for \(\alpha\) and \(\beta\) such that the system is stable for any \(d_0\).

From Equation (10) the magnitude and phase of \(G(s)\) are given by:

\[
|G(j\omega)| = \sqrt{\frac{\beta^2 + (\omega d_0 \alpha)^2}{(\omega d_0)^2}} \tag{11}
\]

\[
\angle G(j\omega) = -\omega d_0 + \arctan\left(\frac{\omega \alpha d_0}{\beta}\right) - \pi
\]

The stability criterion for the closed loop system is:

\[
|G(j\omega)| < 1 \text{ at } \angle G(j\omega) = -\pi \tag{12}
\]
Fig. 21. Region enclosed by the curve is the stable region

This stability criterion is true for systems where $|G(j\omega)|$ crosses the magnitude $= 1$ line only once. This is true for our system. If $\omega_s$ is the frequency at which $|G(j\omega)| = 1$ and $\omega_c$ is the frequency at which $\angle G(j\omega) = -\pi$, then the stability criterion in Equation (12) is equivalent to $\omega_s < \omega_c$ i.e.

$$\omega_s = \frac{1}{d_0} \sqrt{\frac{\alpha^2 + \sqrt{\alpha^4 + 4\beta^2}}{2}} < \omega_c$$  \hspace{1cm} (13)

where $\omega_c$ is the solution of the equation:

$$\frac{\omega d_0 \alpha}{\beta} = \tan(\omega d_0)$$ \hspace{1cm} (14)

There is no closed form solution to the above equation. We need the following condition for the existence of a non-zero solution to Equation (14). See Appendix V for details on this.

$$\frac{\alpha}{\beta} > 1$$  \hspace{1cm} (15)

All that remains now is to solve for $\alpha$ and $\beta$ satisfying Equations (13), (14) and (15). We did this in Matlab by first choosing $\alpha_i$ and $\beta_i$ satisfying (15), then solving (14) numerically for $\omega_c$ and finally determining if inequality (13) is satisfied for the chosen $\alpha_i$ and $\beta_i$. If it is, then for these parameter values ($\alpha_i$, $\beta_i$) the closed loop system is stable. Observe from (13) and (14) that the term $d_0$ on both sides of the inequality gets cancelled and hence the the system is stable for this ($\alpha_i$, $\beta_i$), for every $d_0$. Continuing this way, we obtained the region ($\alpha$, $\beta$) for which the RCP system is stable for all $N$, $C$ and $d_0$. This is shown in Figure 21.

D. The Nyquist Stability Analysis

In the last Section we obtained the stability region from the Bode plot analysis. There are some conditions to be satisfied for the Bode analysis to hold. Specifically, the stability criterion, $|G(j\omega)| < 1$ at $\angle G(j\omega) = -\pi$, holds for systems where $|G(j\omega)|$ crosses the magnitude $= 1$ line once, the most common situation. However, there are systems when the $|G(j\omega)|$ crosses magnitude $= 1$ more than once. A rigorous way to resolve these ambiguities is to use the Nyquist stability criterion. So, in this Section we will use the Nyquist criterion and confirm the stability region obtained before.

Recall that the open loop transfer function of our system is given by $G(s) = e^{-sd_0} (\alpha sd_0 + \beta)/(s d_0)^2$. The closed loop transfer function is $G(s)/(1 + G(s))$. Therefore, the closed loop roots are the solutions of $1 + G(s) = 0$ i.e.

$$d_0^2 s^2 + \alpha d_0 e^{-sd_0} s + \beta e^{-sd_0} = 0$$ \hspace{1cm} (16)

We will write $d_0^2 s^2 + \alpha d_0 e^{-sd_0} s + \beta e^{-sd_0} = 0$ in the form $1 + \frac{b(s)}{a(s)} = 0$. This is given by:

$$1 + \frac{d_0 e^{-sd_0} s + \beta e^{-sd_0}}{d_0^2 s^2 + \beta e^{-sd_0}} = 0$$

i.e. $b(s) = d_0 e^{-sd_0} s$ and $a(s) = d_0^2 s^2 + \beta e^{-sd_0}$. Let $G_1(s)$ denote $\frac{d_0 e^{-sd_0} s}{d_0^2 s^2 + \beta e^{-sd_0}}$. Then, the procedure for obtaining the stability region can be summarized as follows:

1. Fix a value of $\beta$. Plot the nyquist plot of $G_1(s)$
2. Determine the number of unstable (i.e. Right Hand Plane) poles of $G_1(s)$ and call that number $P$.
3. Determine the region on the real axis where $-\frac{1}{\alpha}$ should lie such that if $N$ denotes the number of encirclements of $-\frac{1}{\alpha}$, then $N + P$ should be equal to zero. $Z = N + P$ are the number of unstable closed loop roots, and therefore we want $Z$ to be equal to 0. Note that, $N$ is negative if the encirclement of $-\frac{1}{\alpha}$ is in anticlockwise direction and positive if in clockwise direction.

To draw the nyquist plot of $G_1(s)$, we will use the Pade approximation for $e^{-sd_0}$. Substituting this in $G_1(s)$ we get:

$$G_1(s) = \frac{d_0^2 s^3 - \frac{d_0^2}{2} s^2 + d_0 s}{d_0^4 s^4 + d_0^2 s^3 + (d_0^2 + \beta) s + \beta}$$

Let’s go through the three steps above with an example. Let us take $\beta = 0.3$. Now, we want to find the range of $\alpha$ for which the system is stable. The nyquist plot of $G_1(s)$ is shown in Figure 22 for $\beta = 0.3$. The value of $d_0$ taken does not matter, the plot does not change with $d_0$. $G_1(s)$ has two unstable poles and so $P = 2$. Therefore, in order for $Z$ to be equal to 0 we need $N = -2$, which means $-\frac{1}{\alpha}$ must be encircled twice in anticlockwise direction in the

$$e^{-sd_0} \approx (1 - \frac{d_0}{2} + \frac{(4d_0)^2}{3!})/(1 + \frac{d_0}{2} + \frac{(4d_0)^2}{3!})$$
Fig. 22. Nyquist plot of $G_1(s)$ when $\beta = 0.3$

Fig. 23. Nyquist plot of $G_1(s)$ when $\beta = 0.6$

The only region in the plot where there are two anti-clockwise encirclements is between points $A$ and $B$ shown in the figure. So the stable region is $-2.91 < -\frac{1}{\alpha} < -0.696$ i.e. $0.3436 < \alpha < 1.436$ when $\beta = 0.3$. And thus continuing, we can obtain the stable range for $\alpha$ for every value of $\beta$. When $\beta$ is larger than about 0.55 there does not exist any value of $\alpha$ for which the system is stable. An example of such a nyquist plot is shown in Figure 23. As can be seen, there is no region on the plot where there are two anticlockwise encirclements, and hence we cannot get $Z$ to be equal to 0, and therefore the closed loop system will have atleast one unstable pole for this value of $\beta$.

The stable region obtained from the procedure above is shown in Figure 24. The region obtained from the nyquist analysis is shown in + signs, while that obtained from the Bode plot analysis is shown in solid line. As can be seen, they match well. Thus the nyquist criterion analysis confirms the region that we obtained.

E. Stable Region

Let us now step back and understand why the stability region looks like the way it does in Figure 24. Recall the RCP rate update equation:

$$R(t) = R(t - T)(1 + \frac{T}{\alpha} [\alpha (C - y(t)) - \frac{q(t)}{d(t)}])$$

If there is spare capacity available then the role of $\alpha$ in the above equation is to decide how much of this available capacity should be allotted to the flows in one round trip time. It is intuitively clear that if $\alpha$ is large, say 10 for example, and if there is spare capacity available then we are increasing the aggregate flow rate by 10 times more than what is available. Obviously, this will result in an overshoot from the equilibrium rate, everytime there is even a small $\delta$ amount of spare capacity. So, it is not surprising that $\alpha$ cannot be indefinitely large. As seen in Figure 24, there does not exist a stable region for $\alpha > 1.6$.

The parameter $\beta$ on the other hand determines the amount that the aggregate incoming traffic should be reduced so that any queue at the router can be drained. Given any stable value of $\alpha$, $0 < \alpha < 1.6$, a large value of $\beta$ only means that we are being very aggressive in draining the queue. It is not intuitive as to why this would lead to an unstable system and why $\beta$ is limited to values $\leq 0.55$. The answer to this is that we did not take into...
account the non-linearity in the queue size in Equation 8. Without this non-linearity, it would mean that the queue length can become negative as well. This explains why $\beta$ is capped.

We hypothesize that the actual stability region (i.e. taking the queue non-negativity into account) is larger and includes the one obtained through the linearized model. The initial evidence of this is from simulations. The set up is as follows: 5 long flows, all with the same round trip time, start at time 0. Their equilibrium normalized rate is $\frac{R}{\tau} = 0.2$. 5 more flows then start at time 40. Now, the equilibrium rate of the 10 flows is 0.1. The original five flows then depart at time 100. This experiment is done for every value of $0.1 \leq \beta \leq 1, 0.1 \leq \alpha \leq 2$ in steps of 0.1. For each value of $\beta$ we note the $\alpha$ that is the onset of instability. For example in Figure 25 when $\beta = 0.1$, the system does not stabilize for $\alpha \geq 1.6$. In Figure 26, when $\beta = 0.4$, the system is unstable from $\alpha = 1.3$ onwards. And in Figure 27, when $\beta = 0.6$, the system is unstable from $\alpha = 1.2$ onwards. Note that in our linearized stability region, there is no stable point for $\beta = 0.6$. The plots shown here are just a sample of the simulations we did. The stable region obtained as above is shown in Figure 28. It supports our hypothesis that the stable region of the non-linear system is larger than the linearized system.

**F. Phase Plane Analysis**

In this Section we will obtain the stability region of the RCP parameters, taking into account the non-linearity in
the queue length equation. We will use a method called *phase-plane analysis*, which is used to solve and understand nonlinear control problems. We will first give a brief introduction to this method. The RCP system described by Equations (5) and (8), can be written as:

\[
\begin{align*}
\dot{R}(t) &= f(R(t - T), R(t - d_0), q(t), N, C) \\
\dot{q}(t) &= g(R(t - d_0), N, C)
\end{align*}
\]

where \( f(\cdot) \) and \( g(\cdot) \) are non-linear functions. If we take \( R \) and \( q \) as coordinates of a plane, then to each state of the system there corresponds a point in this plane. As \( t \) varies, this point describes a curve in the \( R - q \) plane, indicating the history of the system dynamics. Such a curve is called a trajectory. The geometrical representation of the system behaviour in terms of trajectories is called a phase-plane representation of the system dynamics. The initial condition determines the initial location of a representative point on the trajectory. As time increases, the representative point moves along the trajectory. A family of such trajectories is called a phase-plane portrait.

We will now plot the phase-plane portraits of the RCP system, and determine the stability region from these plots. The phase portraits are generated by a C program which given an initial state \((R, q)\) does a step-by-step evolution of the RCP system state from Equations (5), (7), (6) and (8). Following is the set up for obtaining the phase portraits:

\( C = 0.15 \) Gbps, \( d_0 = 0.2s \), \( T = 0.01s \) and \( N = 10 \) flows

The initial conditions are chosen to be: \( \gamma_{init} = \{0.5, 0.1, 0.01\} \) where \( \gamma \) is the normalized flow rate i.e. \( \gamma = \frac{R}{N} \), and \( q_{init} = \{2 \cdot C \cdot d_0, 0\} \). The trajectories are drawn for all combinations of \((\gamma_{init}, q_{init})\). \( \alpha \) and \( \beta \) vary from \((0, 1)\) in steps of \(0, 1\)

For each value of \((\alpha, \beta)\) we get a family of trajectories corresponding to each of the initial conditions chosen. Note that the equilibrium state of the system is \((\gamma, q) = (0, 1, 0)\). So, for a given \((\alpha, \beta)\), if for any of the initial conditions the trajectory does not finally end at the equilibrium state we can conclude that this point does not lie in the stable region. Figures 29, 30 and 31 show some sample phase portraits. Figure 29 shows the phase portrait for \((\alpha, \beta) = (0, 6, 0.2)\). This point is well within the linearized stability region and as seen in this Figure, all the trajectories converge to the equilibrium point. Figure 30 shows the phase portrait when \((\alpha, \beta) = (0, 6, 0.8)\), which is a point outside the linearized stability region but within our hypothesized stable region. Here too all the trajectories converge to the equilibrium point. The top plot in Figure 31 shows phase portrait for \((\alpha, \beta) = (1.4, 0.8)\), a point well outside the stable region. The bottom plot in Figure 31 shows that for the initial condition \((\gamma_{init}, q_{init}) = (0.01, 2 \cdot C \cdot d_0)\), the trajectory never settles down at the equilibrium. It continuously keeps oscillating around the equilibrium point. The magnitude of these oscillations only grow larger as we get farther away from the stable region. Finally, Figure 32 shows the stability region obtained from the phase-plane analysis. The stable regions obtained from the linearized analysis and simulations are also shown for the purposes of comparison. As can be seen, the region obtained from the phase-plane analysis matched well with that obtained via the simulations.

The stable region in Figure 32 holds true for varying \( C, d_0 \) and \( N \). The range that we tried are:
C = \{5.6 \text{Kbps}, 0.15 \text{Gbps}, 2.5 \text{Gbps}, 10 \text{Gbps}, 1000 \text{Gbps}\}
\quad d_0 = \{0.01, 0.5, 1, 2\}
\quad N = \{10, 100, 1000, 5000\}

Under all these varying conditions the stable region that we obtained remained the same, as shown in Figure 32.

A part of the stability region of the non-linear system has also been analytically derived using tools from non-linear control theory and is the subject of a separate paper [27].

VI. RELATED WORK

- eXplicit Control Protocol (XCP) [15]: Both XCP and RCP try to approximately emulate processor sharing among flows, which is why their control equations are very similar. The manner in which they converge to PS is quite different; the main difference between XCP and RCP is in the kind of feedback that flows receive. XCP gives a window increment or decrement over the current window size of the flow (which is small for all newly starting flows). At any time XCP flows could have different window sizes and RTTs and therefore different rates. XCP continuously tries to converge to the point where all flows have their fair share rate, by slowly reducing the window sizes of the flows with rates greater than fair share and increasing windows of the flows with rates lesser than fair share, making sure every time that there is no bandwidth over subscription. New flows start with a small window, and the convergence could take several RTTs especially if there is little or no spare capacity. If the flows arrive as a Poisson process with heavy-tailed flow sizes, then most of the flows finish by the time they reach their fair share. In RCP, all flows (new and old) receive the same rate feedback which is their equilibrium rate. This helps flows finish quickly.

XCP is computationally more complex than RCP since it gives different feedback values to different flows, and involves multiplications and additions for each packet. RCP maintains a single rate for all flows and involves no per packet computation.

There are other differences such as: the frequency of updates, the way they estimate the number of flows etc, both of which we found make a difference in performance.

- ATM ABR: Explicit rate based feedback is definitely not novel to this paper, and has been proposed in the ATM ABR literature, for example [7], [8]. However, the algorithms (and their goals) are very different from RCP. In particular, we are not aware of any ATM ABR algorithm whose explicit goal is to minimize flow completion times or has a rate algorithm like that in RCP.
• QuickStart: Jain et al. proposed QuickStart, which is an enhancement to TCP to allow an initial rate to be set, or an allowed initial congestion window, in the TCP SYN packet [24]. They propose that a TCP host indicate its desired sending rate in packets per second in the TCP SYN or SYN/ACK packet and each router in turn could either approve the specified rate or decrease it. Their proposal outlines the framework (which is essentially the same as the startup phase of RCP), but does not suggest how the routers should pick this initial rate.

• Variations of TCP: There are several proposals that change TCP at the end-hosts without any changes at the routers [11], [12], [13]. All of these address the problems with the Congestion Avoidance mode in TCP in high bandwidth-delay networks, when all the flows are long lived. The goals of these algorithms are the same as those of the original TCP which are to maximize link utilization without losing too much on fairness of the flows.

• TCP with scheduling algorithms: There have been proposals to differentiate short and long TCP flows at the router to improve the response times of short flows, by using schedulers at the routers that give preferential treatment to the short flows [25]. While, we think this is a good approach, they do not however change slow-start or AIMD, which are the primary causes of long flow durations.

VII. Conclusion

It is a common experience when downloading file over a fast link, that the transfer takes longer than expected. This is, in part, due to the unnecessary number of round trip times taken by the TCP slow-start and AIMD algorithm to find the fair-share rate. Often, the flow has finished before the fair-share rate has been found. Unfortunately, making the network faster doesn’t help, because the flow duration is dominated by the propagation delay.

It is a premise of this paper that it is better to design a congestion control algorithm to minimize flow duration, than to maximize network throughput. This is, of course, debatable and depends on whether one weighs the network or the user viewpoint most. But we observe that congestion control algorithms almost always favor throughput over flow duration, and our goal is to go some way towards redressing the balance. With network utilization so low nowadays, and with it common to download files from remote servers, it seems unnecessary to fine-tune algorithms so as to squeeze the most throughput from each link. It seems that optimizing for user-perceived performance is more important instead.

RCP is designed to be a practical way to reduce flow duration; and seems to come close to the optimal value in most cases. That it should reduce the flow duration by so much, came as quite a surprise to us, which makes us believe it is a promising congestion control mechanism as bandwidth-delay products increase.

REFERENCES

APPENDIX I

A MODEL FOR THE 'U' CURVE

In this Section we propose an approximate model that predicts the expected flow duration when the flow sizes have bounded Pareto distribution and the traffic in each flow is arriving at a rate, \( R \). The cumulative distribution function of a bounded Pareto distribution is:

\[
F(x) = \frac{1}{1 - (m/M)^\alpha}, \quad m \leq x \leq M, \quad 0 < \alpha \leq 2
\]

Following is the notation used in this section:

- \( C \): Link capacity
- \( R = \frac{C}{K} \): Flow rate (assume that \( K \) is an integer)
- \( E[L] \) and \( E[L^2] \): First and second moments of flow size
- \( E[Q_i] \): Expected queue length as seen on the arrival of the last packet of a flow
- \( \lambda \): Poisson flow arrival rate (flows/s)
- \( \rho = \frac{E[L]}{C} \): Offered load
- \( E[\tau] \): Expected flow duration

We have seen in Section II-A that the expected flow duration is given by:

\[
E[\tau] = \frac{E[L]}{R} + \frac{E[Q_i]}{C}, \quad (18)
\]

So for a given rate, in order to find the expected flow duration, it suffices to find the expected queue length seen by the last packet of a flow, which we will approximate as the expected queue length. To find the expected queue length we will consider two cases:

**Case 1: \( R < C \)**

Consider the three systems shown in Figure 33. Figure 33(a) represents our system of interest where flows arrive according to a Poisson process and each flow transmits data at rate \( R \) where \( R = C/K, K > 1 \). Let us assume for now that \( K \) is an integer. The buffer is served by an FCFS server of rate \( C \). Figure 33(b) represents what we call a Flow-wise FCFS system. This system has \( K \) servers, each of capacity \( C/K \) and the flow arrival process is the same as that of system (a). Flows are queued according to when the first packet of a flow arrives. Whenever a server becomes free, it picks the first flow in queue that is not being served (if there are any) and serves the entire flow to completion. Note that a server will not serve a flow if all flows in the queue have started being served by other servers. Figure 33(c) represents the M/Pareto/K system, which is the same as (b) except each flow arrives as a single entity.

Systems (a) and (b) have the same arrival process for all time instants, and systems (b) and (c) have the same departure process for all time instants. Let \( A_j(t) \) denote the amount of traffic that arrived into system \( j \) by time \( t \) and \( D_j(t) \) denotes the amount of traffic served by system \( j \) up to time \( t \). Assume that all systems are empty at time 0, then the expected queue size in system \( j \) is given by

\[
E[Q_j] = \lim_{t \to \infty} \int_0^t (A_j(t) - D_j(t)) dt.
\]

(19)

Systems (a) and (b) have the same arrival process, but (a) is work-conserving while (b) is not, because an idle server in (b) cannot serve anything if all the flows in queue
have started being served. Thus, we have $D_b(t) \leq D_a(t)$, $\forall t$. So, it follows from (19) that:

$$E[Q_a] \leq E[Q_b].$$  

(20)

Systems (b) and (c) have the same departure process, while $A_b(t) \leq A_c(t)$, $\forall t$ by setup. Thus:

$$E[Q_b] \leq E[Q_c].$$  

(21)

Since the systems (b) and (c) differ only in their arrival processes, we can find the difference between the expected queue lengths as the difference between the arrival processes:

$$E[Q_c] - E[Q_b] = \lim_{t \to \infty} \int_0^t (A_c(t) - A_b(t)) dt.$$  

Suppose $N(t)$ flows have arrived to the system till time $t$, then the contribution of flow $i$ to the above difference, as shown in Figure 34, is $\frac{L_i^2}{2R}$. Summing over all flows gives:

$$E[Q_c] - E[Q_b] = \lim_{t \to \infty} \frac{\sum_{i=1}^{N(t)} \frac{L_i^2}{2R}}{t}.$$  

(22)

From Equations (20), (21) and (22), we have:

$$E[Q_a] \leq E[Q_b] = E[Q_c] - \lambda \frac{E[L]^2}{2R}.$$  

(23)

We will now obtain an approximation for $E[Q_c]$, the expected queue length for an M/Pareto/K system, using the results from [28]. In [28], the authors propose a simple model that accurately predicts the expected response time of a flow in a M/Pareto/K system, where the flows sizes have a bounded Pareto distribution. The expected flow response time in this system is approximated as:

$$E[D] \approx \frac{K E[L]}{C} + \frac{\rho}{1 - \rho} \frac{E[L]^2}{2E[L]} P(\text{blocking}),$$  

(24)

where

$$P(\text{blocking}) = 1 - F_{P(\rho K)}(K(1 - \rho_s) - 1),$$  

(25)

$F_{P(\rho K)}(\cdot)$ denotes the value of the cumulative density function of a Poisson distribution with parameter $\rho_i K$,

$$\rho_s = \frac{\alpha A}{E[L]^2} \rho,$$

$$\rho_i = \frac{(1 - \alpha) B}{E[L]^2} \rho,$$

$$A = E[L] - \sqrt{(E[L]^2 - (E[L])^2 \frac{1 - \alpha}{\alpha}),}$$

$$B = E[L] + \sqrt{(E[L]^2 - (E[L])^2 \frac{1 - \alpha}{\alpha}),}$$

and $\alpha$ is the percentage of flows with sizes between $m$ and $M/20$.

Now, in order to obtain $E[Q_c]$, suppose $E[N]$ is the expected number of flows in the M/Pareto/K system. The probability the system is busy is approximately $\rho$.

$$E[Q_c] \approx \rho \left[ \frac{E[N]}{\rho} - K \right] E[L] + K E[L_r],$$  

(26)

where $E[L_r]$ is the expected residual file size and is equal to $\frac{E[L^2]}{2E[L]}$. Using Little’s Law, $E[N]$ in the above expression is equal to $\lambda E[D]$ where an approximation to $E[D]$ is given in Equation (24). Then, substituting for $E[D]$ in (26) and using (23) we have:

$$E[Q_a] \approx \lambda \frac{\rho}{2(1 - \rho)} \frac{E[L]^2}{C} P(\text{blocking}),$$  

(27)

where $P(\text{blocking})$ is given in (25).

Now we have the expected queue occupancy of our system. If we approximate the expected queue occupancy seen by the last packet of a flow, $E[Q_a]$, to be the expected queue occupancy, $E[Q_a]$, then substituting (27) in (18), we get:

$$E[\tau] \approx E[L] \frac{K}{C} + \lambda \frac{\rho}{2(1 - \rho)} rac{E[L]^2}{C^2} P(\text{blocking}).$$  

(28)

**Case 2: $R \geq C$**

Now let us consider the case when $R \geq C$. Consider the three systems as in Figure 33, but with $K = 1$ and $R \geq C$. Then, systems (a) and (b) have the same arrival and departure processes, therefore $E[Q_a] = E[Q_b]$. For 6The probability that system (a) is busy is $\rho$. Since system (c) is not work conserving the probability that the system is busy is higher than $\rho$. For simplicity, we shall assume that it is approximately $\rho$.
systems (b) and (c), we again have $D_b(t) = D_c(t)$ and $A_b(t) \leq A_c(t)$. Thus, $E[Q_b] \leq E[Q_c]$. In the same way as in Case 1, the difference $E[Q_c] - E[Q_b]$ is found to be $\lambda \frac{E[L^2]}{2R}$. Thus,

$$E[Q_a] = E[Q_c] - \lambda \frac{E[L^2]}{2R},$$

where $E[Q_c]$ is the expected queue occupancy in an M/G/1 system and is equal to $\frac{\lambda E[L^2]}{2(1-\rho)C}$. So,

$$E[Q_a] = \frac{\lambda E[L^2]}{2} \left( \frac{1}{(1-\rho)C} - \frac{1}{R} \right),$$

Substituting $E[Q_a]$ in the delay equation (18) gives:

$$E[\tau] = \frac{E[L]}{R} + \frac{\lambda E[L^2]}{2C} \left( \frac{1}{(1-\rho)C} - \frac{1}{R} \right).$$ (29)

Equations (28) and (29) model the expected flow duration, when $R < C$ and $R \geq C$ respectively. Plotting these delay equations verses $R$ show that the expected flow duration has a unique minimum. Figure 35 shows an example comparing the average flow duration given by the model and the simulations.

**APPENDIX II**

**SOME OBSERVATIONS AT THE OPTIMAL DELAY POINT**

Our observation in several simulations is that at the point when the delay is minimized, the average queue occupancy is small as shown in the Figure 36, and further the link utilization is equal to the offered load, $\rho$. At rates below the optimal, although the queue occupancy is still small, the link utilization is less then the offered load (meaning that the work given to the system is unnecessarily stretched), while at rates above the optimal, although the link utilization is high, the mean queue occupancy is high as well.
Fig. 37. AFCT of RCP, TCP and XCP under a single bottleneck with C = 2.4 Gbps, RTPD = 0.1 s, ρ = 0.4, pareto distributed flows with shape = 1.2 and mean flow size = 500 pkts.

Fig. 38. AFCT of RCP, TCP and XCP under a single bottleneck with C = 2.4 Gbps, RTPD = 0.1 s, ρ = 0.8, pareto distributed flows with shape = 1.2 and mean flow size = 500 pkts.

\[ R(t) = R(t-T) \left( \frac{\alpha(C-NR(t-d(t))}{Cd(t)} - \beta \frac{q(t)}{d(t)} \right) \]

We define:

\[ f(R_T, R_d, q) = R_T \left( 1 + \frac{\alpha(C-NR_d)}{C} \right) \]
\[ g(R_d) = NR_d - C \]

where \( R_T = R(t-T), R_d = R(t - d(t)), d = d(t) \) and \( q = q(t) \). Recall that the equilibrium point is given by:

\[ \dot{q}(t) = 0 \Rightarrow NR_e = C \Rightarrow R_e = \frac{C}{N} \]
\[ \dot{R}(t) = 0 \Rightarrow R_e \left( \frac{\alpha(C-NR_e)}{Cd} \right) = 0 \Rightarrow q_e = 0 \]

From above, the equilibrium value of \( d(t) \) is \( d_e = d_0 \). Evaluating partials of \( f \) and \( g \) at the equilibrium point \((R_e, q_e, d_e) = \left( \frac{C}{N}, 0, d_0 \right) \) gives:

\[ \frac{\partial f}{\partial R_d} = -\frac{NR_T \alpha}{Cd} \bigg|_{R_T = \frac{C}{N}, d = d_0} \]
\[
\frac{df}{dR_T} = \frac{\alpha}{d} - \frac{N \alpha R_d}{Cd} - \frac{\beta q}{d^2C} \quad R_d = \frac{c}{k}, \quad d = d_0, \quad q = 0
\]

\[
\frac{df}{dq} = \frac{\beta R_T (\frac{q}{c} - d)}{C(\frac{q}{c} + d)^3} \quad R_T = \frac{c}{k}, \quad q = 0, \quad d = d_0
\]

\[
\frac{dg}{dR_d} = N
\]

The linearized equations are:

\[
\delta \dot{R}(t) = \frac{\partial f}{\partial R_d} \delta R(t - d_0) + \frac{\partial f}{\partial R_T} \delta R(t - T) + \frac{\partial f}{\partial q} \delta q(t)
\]

\[
\delta \dot{q}(t) = \frac{\partial g}{\partial R_d} \delta R(t - d_0) = N \delta R(t - d_0)
\]

where

\[
\delta R \triangleq R - R_e \quad (33)
\]

\[
\delta q \triangleq q - q_e
\]

**APPENDIX V**

**Bode Plot Analysis**

In this Appendix we will see why we need the condition \( \frac{q}{\beta} > 1 \), in order for Equation (14) to have a non-zero solution. Recall that if Equation (14) has a solution, \( \omega_c \), then this is the frequency at which the phase plot of \( G(s) \) crosses the \(-\pi\) line. In other words, at \( \omega_c \) we have:

\[
\angle G(j \omega_c) = -\omega_c d_0 + \arctan\left(\frac{\omega_c \alpha d_0}{\beta}\right) = -\pi
\]

Notice that \( \angle G(j \omega) = -\pi \) at \( \omega = 0 \). And for large \( \omega \), \( \angle G(j \omega) \) is much smaller than \(-\pi\). So, unless \( \angle G(j \omega) \) first increases and then decreases, as \( \omega \) increases from 0, it will not cross the \(-\pi\) line. Thus, the condition that there should exist a maxima for \( \angle G(j \omega) \). Differentiating \( \angle G(j \omega) \) and setting it to 0 gives:

\[
\frac{d}{d\omega} \angle G(j \omega_m) = -d_0 + \frac{\alpha d_0}{\beta} \frac{1}{1 + (\frac{\omega_m \alpha d_0}{\beta})^2} = 0
\]

\[
\Rightarrow \omega_m = \frac{\beta}{\alpha d_0} \sqrt{\frac{\alpha}{\beta} - 1
\]

Fig. 39. \((\alpha, \beta) = (0.2, 0.4): \omega_c \) does not exist

Fig. 40. \((\alpha, \beta) = (0.4, 0.2): \omega_c \) exists

Obviously, the above maxima exists only if \( \frac{q}{\beta} > 1 \)

Thus, if the condition \( \frac{q}{\beta} > 1 \) is satisfied then the the phase plot crosses the \(-\pi\) line. Examples: the plot in Figure 39 shows a Bode Plot for \( (\alpha, \beta) = (0.2, 0.4) \). Notice that the phase always decreases starting from \(-\pi\), and never crosses the \(-\pi\) line for any non-zero \( \omega \). Hence \( \omega_c \) does not exist. Figure 40 shows a Bode Plot for the case \( (\alpha, \beta) = (0.4, 0.2) \) and in this case, since \( \frac{q}{\beta} > 1 \), \( \omega_c \) exists.