PIQI-RCP: Design and Analysis of Rate-Based Explicit Congestion Control

Saurabh Jain and Dmitri Loguinov*
Texas A&M University, College Station, TX 77843
Email: saujain@tamu.edu, dmitri@cs.tamu.edu

***Abstract*—Recent research has created a significant interest [18], [19] in the design and development of a new global-scale communication network that can overcome the limitations of the current Internet. Among the many directions for improving networking technology, recent pursuit to design better flow control has led to the emergence of *explicit* congestion control methods that solicit active feedback from intermediate routers [5], [11], [22], [25], [26]. As a step towards understanding these methods, we analyze stability and transient performance of Rate Control Protocol (RCP) [5], which is a recent method that aims to emulate processor sharing and achieve quick flow completion. However, we find that RCP can become unstable in certain topologies and may exhibit very high buffering requirements at routers. To address these limitations, we propose a new controller called *Proportional Integral Queue Independent RCP* (PIQI-RCP), prove its stability under heterogeneous delay, and use `ns2` simulations, as well as Linux experiments, to show that the new method has significantly better transient dynamics (i.e., much smaller link-capacity overshoot) and better stability properties in single- and multi-bottleneck scenarios.**

I. Introduction

The enormous growth and success of the Internet has recently raised questions about the ability of existing network algorithms to maintain the same level of performance as the Internet continues to change. One specific area of concern is congestion control in high-speed networks, where theoretical, simulation, and experimental results [9], [12] suggest that the current version of TCP may experience serious scalability problems as the bandwidth-delay product of network links continues to increase. One approach to solving this issue is to develop more aggressive end-to-end algorithms [3], [9], [10], [12], [13], [20], [24], while another direction [18], [19] is to re-design the Internet to utilize explicit network feedback from Internet routers [5], [11], [22], [23], [25], [26] to achieve faster convergence, smaller packet loss, and better fairness between end-users. Analysis and improvement of protocols in the latter category is the focus of this paper.

Explicit congestion control algorithms require network devices in each path to send feedback to end-hosts indicating the actual degree of congestion rather than just a binary signal (e.g., packet loss or ECN bits) stating whether congestion is present or not. To generate feedback $p_l(t)$, each router l performs certain processing of incoming data and executes a controller whose input is the aggregate rate of all flows observed within a certain time-interval. This feedback is then inserted into headers of all passing packets and later communicated to sources in acknowledgments. Explicit feedback provides more accurate congestion information to end-hosts and allows them to adjust their congestion window size or sending rate with much better results, often leading to high link utilization, max-min fairness, and zero packet loss.

Among several recent proposals, *Rate Control Protocol* (RCP) [5] is a simple method that achieves max-min fairness in the steady-state and maintains close-to-optimal average flow completion time (AFCT). RCP requires lower per-packet computation overhead compared to some of the traditional methods (such as XCP [11]) and has a smaller control header size compared to most other schemes (including JetMax [26], MKC [25], and XCP [11]). However, the existing literature lacks thorough analysis of RCP regarding its stability in general networks and transient dynamics.

A. Our Contributions

In this work, we aim to fill the void in understanding RCP and study its vulnerabilities in practice. Our results show that RCP can behave in an unstable manner in certain topologies under highly heterogeneous delays. We also find that in the pursuit of achieving lower AFCT, end-hosts in RCP use an overly aggressive controller that in certain cases drastically overshoots link capacity and requires huge buffers inside routers. To overcome the former drawback, we first investigate a method called *Queue Independent RCP* (QI-RCP) that addresses stability issues in RCP by removing the queue term from the router controller and properly normalizing its gain parameter. We derive stability conditions for QI-RCP under heterogeneous RTTs and show `ns2` [17] simulations that confirm the tightness of the obtained bounds.

Even though QI-RCP is stable in topologies where the original RCP was unstable, the new method still requires the same huge buffers inside routers. To address this issue, we next propose *Proportional Integral Queue Independent RCP* (PIQI-RCP)[1], which consists of a PI controller inside routers and an EWMA-type controller at end-users. We establish stability conditions for PIQI-RCP under heterogeneous RTTs and mild assumptions on the router control interval. Simulations show that PIQI-RCP's stability bounds derived from a theoretical model are tight in real networks and that the protocol remains stable in single- and multi-bottleneck topologies. We also

*Supported by NSF grants CCR-0306246, ANI-0312461, CNS-0434940, and CNS-0519442.

[1]Pronounced "Picky-RCP."

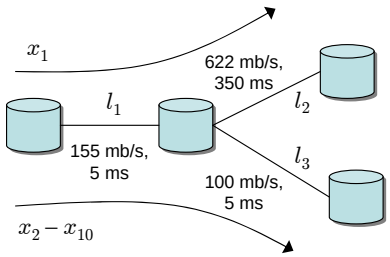
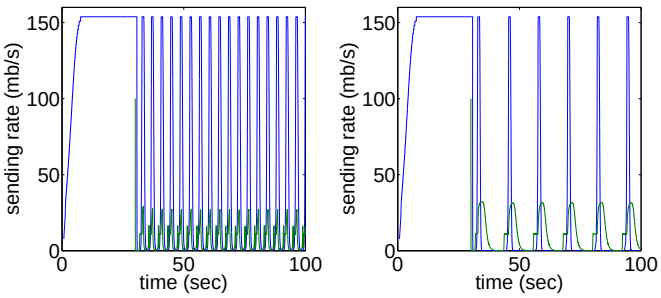


Fig. 1. Topology $\mathcal{T}_1$.



(a) oscillating bottlenecks (b) fixed bottlenecks

Fig. 2. Unstable rates of RCP flows in topology $\mathcal{T}_1$ (default router control interval $T = 10$ ms).

show using Linux experiments that PIQI-RCP significantly outperforms RCP in terms of transient capacity overshoot, peak queue size, and buffer space requirements. Since PIQI-RCP has stability conditions that can be easily satisfied inside routers, incurs small overhead inside routers, and exhibits smaller AFCT than TCP or XCP, it poses an appealing alternative to existing methods.

The rest of the paper is organized as follows. In Section II, we briefly review RCP and introduce common terminology. In Section III, we discuss RCP's strengths and weaknesses. Section IV focuses on the design and analysis of QI-RCP and PIQI-RCP. In Section V, we compare RCP and PIQI-RCP using `ns2` simulations and in Section VI using Linux experiments. We conclude the paper and suggest directions for future work in Section VII.

II. Background

Assume N flows in the network whose sending rates at time t are $x_1(t), \ldots, x_N(t)$. For max-min fairness, each flow responds only to feedback $p_l(t)$ of the most-congested router l of its path, which we call the *bottleneck link* of the flow. The forward/backward delays of flow i to/from its bottleneck link are denoted by $D_i^{\rightarrow}$ and $D_i^{\leftarrow}$, respectively. Then the RTT of each flow can be written as $D_i = D_i^{\rightarrow} + D_i^{\leftarrow}$ and the aggregate traffic arrival rate at router l can be expressed as $y_l(t) = \sum_{i \in l} x_i(t - D_i^{\rightarrow})$, where notation $i \in l$ denotes the set of flows passing through link l [15].

Rate Control Protocol (RCP) [5] is an explicit congestion control method based on max-min fairness. Each router l computes the desired sending rate $R_l(t)$ for flows bottlenecked at l using the following non-linear controller

$$R_l(t) = R_l(t - T)\left[1 + \frac{T}{d_l C_l}\left(\alpha(C_l - y_l(t)) - \beta\frac{q_l(t)}{d_l}\right)\right], \quad (1)$$

where α and β are constants, d_l is a moving average of flow RTTs sampled by router l, C_l is its link capacity, $q_l(t)$ is its instantaneous queue size at time t, and T is its control interval.

The goal of controller (1) is to converge the input traffic rate $y_l(t)$ to link capacity C_l and set the queue length $q_l(t)$ to zero. To accomplish this, rate $R_l(t)$ is fed back to end-flows that adjust their sending rates $x_i(t)$ by setting them to the smallest suggested rate of their paths

$$x_i(t) = \min_{l \in i} R_l(t - D_i^{\leftarrow}), \quad (2)$$

where $l \in i$ is the set of routers along the path of flow i [15]. Since all flows bottlenecked at l set their sending rate to the same received feedback $R_l(t)$, (2) ensures fairness among flows. Assuming infinite queues, (2) also guarantees that completion time of each flow is close to the minimum possible.

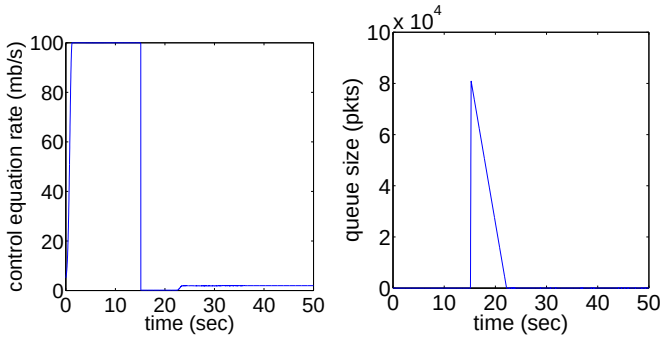
III. Analysis of RCP

In this section, we first demonstrate several drawbacks associated with RCP and then summarize its known strengths.

A. Limited Understanding of Stability

In [6], RCP has been analyzed for stability assuming all flows have *homogeneous* RTTs (i.e., $D_i = D$ for all i). For users with heterogeneous RTTs, stability results of RCP are available only using simulations for several choices of delays. The presence of the queue term $q_l(t)$ in the router control equation (1) makes rigorous stability analysis difficult, because the stationary queue size $q^* = 0$ lies in the region where $q_l(t)$ cannot be linearized. It is therefore unknown how to select parameters for arbitrary choices of flow delay and whether the RCP equation is stable in general networks. We next show one example where RCP behaves in an unstable manner and then explain what causes it.

Consider a multi-link topology $\mathcal{T}_1$ in Fig. 1, where flow x_1 with RTT 710 ms is bottlenecked at link l_1 and flows $x_2 - x_{10}$ with RTT 20 ms are bottlenecked at link l_3. Flow x_1 starts at time $t = 0$ and converges to 155 mb/s after a short delay. Then at time $t = 30$, the other nine flows join the system and instantly become unstable as demonstrated in Fig. 2(a) using `ns2` simulations. Similar unstable behavior of RCP has been observed in [1]; however, it was attributed to the oscillation of the bottleneck of flows $x_2 - x_{10}$ between links l_1 and l_3. As we discuss below, oscillating bottlenecks are the *effect* of a destabilized RCP equation rather than its *cause*. This is confirmed in Fig. 2(b) after fixing the bottleneck of flow x_1 to l_1 and that of the remaining flows to l_3. As seen in the figure, the system remains unstable even though the assignment of bottlenecks is correct and time-invariant. We next elaborate on how this unstable scenario arises and what causes it.



(a) rate x_1 (b) queue size

Fig. 3. Performance of RCP after abrupt increase in traffic demand at $t = 15$.

Recall that in the Internet, routers may carry a mix of flows some of which are bottlenecked locally (i.e., at the current router) and some remotely (i.e., at other routers). In such cases, the average RTT d_l computed by router l may not reflect the correct delay of flows that are being controlled by the router. To simplify discussion, define flows to be *responsive* with respect to router l if they adapt their rates based on feedback $R_l(t)$ (i.e., link l is their bottleneck). Further define flows to be *unresponsive* with respect to l if they are being controlled by some other router $r \neq l$. It then follows from common sense that metric d_l must include the RTTs of only responsive flows at l rather than all flows passing through it. In topology $\mathcal{T}_1$, link l_1 is controlling only flow x_1 since the other nine flows are bottlenecked at l_3; however, all 10 flows contribute to the computation of d_l. This leads to a large discrepancy between the flow response time at link l_1 and the average delay computed by the router (i.e., $d_l = 89$ ms instead of 710 ms) and causes instability.

As the rate of x_1 starts to oscillate, the queue in l_1 responds by fluctuating with each overshoot and undershoot of link capacity. This causes the RTT of flows $x_2 - x_{10}$ to fluctuate as well, which in turn destabilizes the controller in l_3 since it is closely coupled with flow delays, all of which leads to the result in Fig. 2. Note that a similar problem surfaces in other explicit-feedback controllers that rely on average delay, one notable example being XCP [11]. We omit simulations showing this result for brevity, but note that analysis of congestion control under heterogeneous delay is of fundamental importance in diverse networks such as the Internet.

B. Link Overshoot

Besides unknown stability conditions in certain cases, RCP is very aggressive during flow join and can significantly overshoot link capacity in the transient phase. Recall that RCP's source controller (2) sets the sending rate $x_i(t)$ of each joining flow i to the feedback received from the router. As a result, new flows entering the system directly use the current router control rate as their sending rate. For a router that is already in its steady state (i.e., $C_l = y_l(t)$), this causes the input traffic rate to overflow the link and create a sudden surge in queue size. The problem becomes severe when a large number of flows join the system simultaneously. This is illustrated by the `ns2` simulations we show next.

Consider a dumb-bell topology with a 100 mb/s bottleneck link and delay 50 ms. At $t = 0$, flow x_1 enters the system and quickly saturates the bottleneck link as shown in Fig. 3(a). At $t = 15$, flows $x_2 - x_{51}$ enter the system and obtain the current control rate $R_l(t) = 100$ mb/s from the router. For at least one RTT following the join, all 51 flows use the latest rate provided to them (i.e., $x_i(t) = C_l$) and produce a combined input load on the router equal to 5.1 gb/s. Assuming unlimited buffer space, this aggressive behavior increases the queue length to $80,868$ packets (i.e., 81 MB) as shown in Fig. 3(b), which takes the router 7.5 seconds to drain. Considering that the commonly deployed [2] rule is to allocate buffers equal to the bandwidth-delay product of the link (i.e., 1.25 MB in this case), RCP's overshoot requires 64 times more buffer space than used today in most commercial routers [4], [21].

Hence, unless unrealistically large buffers are provisioned inside routers, RCP will sustain significant packet loss, which may result in drastic rate reductions[2], retransmissions, slow convergence, and even instability. As we show later in this paper, RCP's buffering requirement increases proportionally to the number of flows entering the system and the delay to drain the queue increases dramatically with flow RTT. In highly dynamic scenarios where flows constantly join and depart the system, RCP may require more buffer space or even offer lower performance than the traditional TCP, which is highly undesirable in practice.

C. Strengths

Apart from the drawbacks identified above, RCP has certain strengths as well. First, RCP requires lower per-packet computation to compute the feedback signal inside routers than some of its counterparts (e.g., 2 additions and 2 multiplications compared to 6 additions and 3 multiplications in XCP [11]). Second, RCP has a smaller control header size (i.e., 16 bytes) compared to XCP's 20 bytes [8], JetMax's 32 bytes [26], and MKC's 20 bytes [25]. Third, unlike XCP [14], RCP's steady-state rates achieve max-min fairness in general network topologies. Finally, RCP [5] has a much smaller average flow completion time than XCP or TCP, which allows short flows to quickly utilize available bandwidth and finish their transfers. Considering these strengths, we next strive to improve upon the drawbacks of RCP.

IV. Improving RCP

In this section, we propose several modifications to RCP's router and user control equations (1)-(2) to address its stability and link-overshoot issues.

[2]Note that RCP does not specify how flows should react to packet loss or recover dropped packets ([5] uses infinite buffers for all simulations). However, a common technique [11] is to use TCP's recovery mechanisms (i.e., reduction of the rate in half) until all lost packets are recovered.

A. QI-RCP

We next introduce a simple modification to RCP, which we call *Queue Independent RCP* (QI-RCP), that decouples queue dynamics from the feedback computed by the router. This allows us to prove heterogeneous stability of the new controller and therefore achieve good performance in topology $\mathcal{T}_1$. We then improve QI-RCP by reducing its overshoot of link capacity and lowering buffering requirements at routers.

Define the error function of link l at time t to be

$$e_l(t) = 1 - \frac{y_l(t)}{\gamma_l C_l}, \tag{3}$$

where $y_l(t)$ is the combined input rate of all flows at router l, C_l is its capacity, and $0 < \gamma_l < 1$ is the desired link utilization in the steady-state. Then, the router controller of QI-RCP is given by

$$R_l(t) = R_l(t-T)[1 + \kappa e_l(t)], \tag{4}$$

where $R_l(t)$ is the control rate, T is the control interval, and κ is a constant whose range we determine below.

Control equation (4) is similar to RCP's version (1), but does not include the queue term inside the error function. In order to drain any possible queue build-up, the QI-RCP controller operates with a virtual link capacity $\gamma_l C_l$ instead of the physical capacity C_l. End-flow rates are still adjusted using (2), which coupled with (4) represents an integral controller that tries to converge the input traffic rate $y_l(t)$ to the virtual link capacity $\gamma_l C_l$ (see [16] for more discussion of integral controllers). Hence, in the steady state $y_l(t) = \gamma_l C_l$ and any packets buffered due to transient affects in the system are drained using spare bandwidth $(1-\gamma_l)C_l$.

We next analyze QI-RCP's stability. Although multiple packets may carry the same feedback value computed by the router during the last control interval, each end-user in QI-RCP responds to each feedback no more than once and may be viewed as operating on the time-scale of T units even if its RTT is smaller than T. Keeping this in mind and converting the delayed feedback loop of (4) to

$$R_l(t) = R_l(t-T)\left[1 + \kappa\Big(1 - \frac{1}{\gamma_l C_l}\sum_{i=1}^{N} R_i(t-D_i)\Big)\right], \tag{5}$$

we arrive at the following result.

Theorem 1: Assume N flows with heterogeneous RTTs and define $D = \max\{D_1, \ldots, D_N\}$, $D' = \lceil D/T \rceil$. Then, the discrete version (5) of QI-RCP is locally asymptotically stable if $0 < \kappa < \kappa^*$, where

$$\kappa^* = 2\sin\Big(\frac{\pi}{2(2D'-1)}\Big). \tag{6}$$

Furthermore, if flow RTTs are homogeneous (i.e., $D_i = D$ for all i), the above condition is also necessary.

Proof: The z-transform of the linearized system (5) is given by

$$R_l(z) = R_l(z)\Big[z^{-T} - \frac{\kappa}{N}\sum_{i=1}^{N} z^{-D_i}\Big] + K, \tag{7}$$

where K is a constant. The system transfer function is then

$$R_l(z) = \frac{K/(1-z^{-T})}{1+G(z)}, \tag{8}$$

where

$$G(z) = \frac{\kappa}{N}\sum_{i=1}^{N}\frac{z^{-D_i}}{1-z^{-T}}. \tag{9}$$

The transfer function $G(z)$ in the frequency domain can be written as

$$G(e^{j\omega}) = \frac{\kappa}{N}\sum_{i=1}^{N}\frac{e^{-j\omega(D_i-T)}}{e^{j\omega T}-1}. \tag{10}$$

After expanding the exponentials, (10) can also be written as

$$\begin{aligned} G(e^{j\omega}) \;=\; & -\frac{\kappa}{2N\sin(\omega T/2)}\sum_{i=1}^{N}\left[\sin\frac{\omega(2D_i-T)}{2}\right. \\ & \left. +j\cos\frac{\omega(2D_i-T)}{2}\right]. \end{aligned} \tag{11}$$

Most discrete controllers inside a router must operate every T time steps, keeping the rate constant between the control intervals. This can be converted to the case of $T = 1$ by normalizing each delay using $D_i' = \lceil D_i/T \rceil$, where T and D_i are given in time units, while D_i' in router control-steps.

For $D_i' = D'$ and $T = 1$, (11) can be written as

$$\begin{aligned} G(e^{j\omega}) \;=\; & -\frac{\kappa}{2\sin(\omega/2)}\left[\sin\frac{\omega(2D'-1)}{2}\right. \\ & \left. +j\cos\frac{\omega(2D'-1)}{2}\right]. \end{aligned} \tag{12}$$

It can be seen that $G(e^{j\omega})$ crosses the real axis (i.e., $\mathrm{Im}[G(e^{j\omega})] = 0$) for $\omega_i = (2i+1)\pi/(2D'-1)$, where i is an integer. For $i = 0$, we have $\omega_0 = \pi/(2D'-1)$ and the real part of $G(e^{j\omega_0})$ is

$$\mathrm{Re}[G(e^{j\omega_0})] = \frac{-\kappa}{2\sin(\omega_0/2)} = \frac{-\kappa}{2\sin\left(\frac{\pi}{2(2D'-1)}\right)}. \tag{13}$$

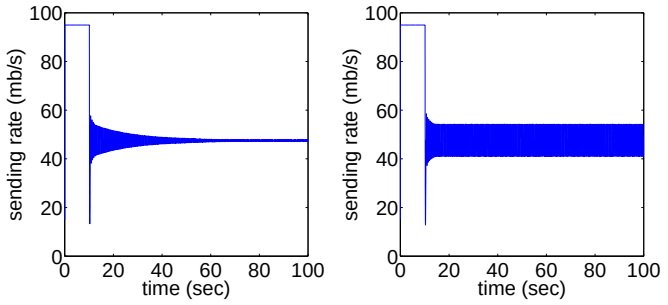
Applying Nyquist stability criterion, it can be seen that stability is ensured if and only if

$$0 < \kappa < 2\sin\left(\frac{\pi}{2(2D'-1)}\right). \tag{14}$$

However, when delays are heterogeneous $G(e^{j\omega})$ crosses the real axis (i.e., $\mathrm{Im}[G(e^{j\omega})] = 0$) when

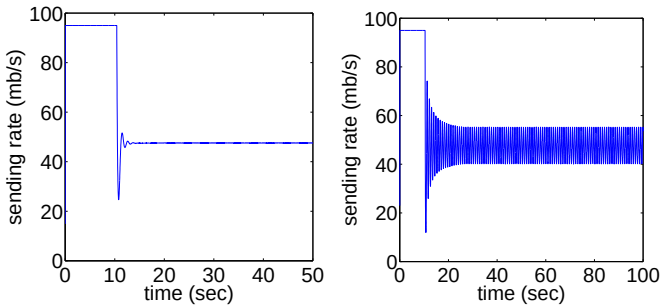
$$\sum_{i=1}^{N}\cos\frac{\omega(2D_i'-1)}{2} = 0. \tag{15}$$

It can be seen that none of the roots ω_i' of the above equation have absolute value smaller than $\omega_0 = \pi/(2D'-1)$. We prove this by contradiction. Assume that $0 \le \omega_0' < \omega_0$ is the smallest root of (15). Then, $0 \le \omega_0' D_i < \pi/2$, which means that *all* cosine terms in the summation are strictly positive, which contradicts the assumption that ω_0' is a root of (15). Since cosine is a symmetric function, we immediately obtain the



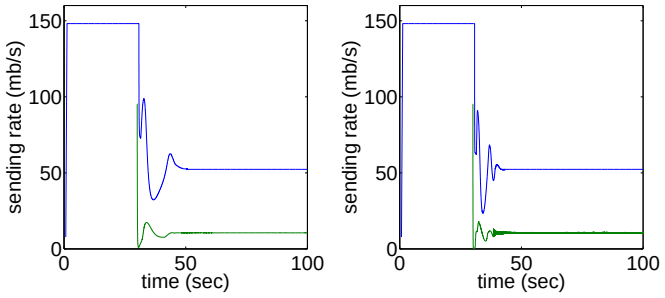
(a) $\eta = 0.99$ (b) $\eta = 1.01$

Fig. 4. Flow rate of QI-RCP for $\kappa = \eta\kappa^*$ in a single-link network with $T = 10$ ms and two flows with homogeneous RTTs $D_1 = D_2 = 120$ ms ($T = 10$ ms, $\gamma = 0.95$).



(a) $\eta = 0.99$ (b) $\eta = 1.8$

Fig. 5. Flow rate of QI-RCP for $\kappa = \eta\kappa^*$ in a single-link network with $T = 10$ ms and two flows with heterogeneous RTTs $D_1 = 122, D_2 = 306$ ms ($T = 10$ ms, $\gamma = 0.95$).



(a) $\eta = 0.5$ (b) $\eta = 0.99$

Fig. 6. Sending rate of QI-RCP in topology $\mathcal{T}_1$ for $\kappa = \eta\kappa^*$ ($T = 10$ ms, $\gamma = 0.95$).

same contradiction for $-\omega_0 \leq \omega_0' < 0$. Therefore, it follows that $|\omega_0'| \geq \omega_0$.

Also, due to the periodic nature (with period 2π) of the frequency domain of a discrete-time system, we can limit our attention to $\omega_i' \in [-\pi, \pi]$. Again, π is a solution to (15) since $(2D_i' - 1)$ is odd for any integer D_i'. Based on these arguments, the smallest root ω_0' of (15) should satisfy $0 < \omega_0' \leq \pi$ and $\omega_0' > \omega_0$. Again, because of the monotonicity of the sine function between 0 and $\pi/2$, the condition $\sin(\omega_0'/2) > \sin(\omega_0/2)$ holds. For ω_0', the real part of $G(e^{j\omega_0'})$ is

$$\begin{aligned} \mathrm{Re}[G(e^{j\omega_0'})] &= -\frac{\kappa}{2N\sin(\omega_0'/2)} \sum_{i=1}^{N} \sin\frac{\omega_0'(2D_i' - 1)}{2} \\ &\geq -\frac{\kappa}{2\sin(\omega_0'/2)} \\ &\geq -\frac{\kappa}{2\sin(\omega_0/2)} \\ &= -\frac{\kappa}{2\sin\left(\frac{\pi}{2(2D'-1)}\right)}. \end{aligned} \tag{16}$$

The above inequality is obtained by bounding the sines with 1, using $\sin(\omega_0'/2) > \sin(\omega_0/2)$, and remains valid even if $\omega_0' < 0$. Therefore, the magnitude of the point at which the real axis is crossed can only be *reduced* (i.e., moved closer to zero) in the heterogeneous case compared to that in the homogeneous case. Finally, noticing that the remaining ω_i' are larger than ω_0', it follows that they can only shift (16) further toward zero and thus lead to looser bounds on κ. Hence, using Nyquist stability criterion, a sufficient condition to ensure stability is

$$0 < \kappa < 2\sin\left(\frac{\pi}{2(2D' - 1)}\right), \tag{17}$$

which leads to the desired result. ■

Note that in the synchronized case (i.e., $D_i = T$ for all i), κ^* is simply 2. Furthermore, for small $T/D \approx 0$, stability margin $\kappa^* \approx \pi T/(2D)$, which can be directly obtained using continuous-time analysis of (5); however, in cases when flow RTTs are close to T, the two expressions are different.[3] Furthermore, as a rule of thumb, upper bound (6) can be considered necessary when delays are close to each other (i.e., $Var[D_i] \approx 0$); however, for highly varying delays, the system may be stable even when κ exceeds κ^* as we show below.

B. Tightness of the QI-RCP Stability Condition

We next examine QI-RCP's stability in `ns2` and investigate how good the derived bounds on κ are. Fig. 4 shows the sending rate of QI-RCP in a single-link network with two flows, both having steady-state RTT equal to 120 ms. The router adjusts its κ by keeping it equal to $\eta\kappa^*$, where η is a scale parameter and κ^* is the upper bound in (6). Observe in the figure that $\eta = 0.99$ keeps the system stable and $\eta = 1.01$ makes it unstable, confirming the sufficiency and necessity of condition $\kappa < \kappa^*$. In Fig. 5, we show an example of a system with heterogeneous RTTs. Observe that the system is stable for $\eta = 0.99$ and only becomes unstable when η reaches 1.8. This confirms that when delays are highly variable, $\kappa < \kappa^*$ is sufficient, but not necessary.

We finish our `ns2` simulations with QI-RCP by showing in Fig. 6 its performance in topology $\mathcal{T}_1$ where RCP was unstable. As seen in the figure, at $t = 0$ flow x_1 enters the

[3]Note that RCP's stability for homogeneous RTT has been established only for the continuous approximation to (1). While gain parameters used in [5] are conservative enough to keep the discrete system (1) stable, the exact stability region of discrete RCP even for homogeneous delays is unknown.

system and quickly saturates its bottleneck link l_1. At $t = 30$, nine additional flows $x_2 - x_{10}$ join the system, which causes small oscillations from which the network quickly recovers and converges to a stable steady state. Note that QI-RCP remains stable for η as high as 0.99 (see Fig. 6(b)), where larger η actually speeds up convergence since routers make more aggressive changes to their $R_l(t)$.

QI-RCP has mathematically tractable stability conditions that can be easily satisfied since routers have access to the RTT of each flow using QI-RCP packet headers, which are similar to those in RCP. Knowing the maximum flow RTT D in every control interval, QI-RCP dynamically tunes the control parameter κ so as to comply with the stability condition $\kappa < \kappa^*$. This keeps the system stable even when D is time-varying due to the changes in the queuing delay as seen in simulations above. However, the main drawback of QI-RCP is that it uses the same aggressive source controller (2), which leads to similar queue build-up as in RCP. We next focus on modifying the user controller to eliminate this drawback.

C. PIQI-RCP

In this section, we propose a new congestion control framework called *Proportional Integral Queue Independent RCP* (PIQI-RCP) that significantly improves the transient behavior of the methods studied earlier in this paper. We start with the router controller of the form

$$R_l(t) = R_l(t-T)[1 + \kappa_1 e_l(t) + \kappa_2 e_l(t-T)], \qquad (18)$$

where error $e_l(t)$ is given by (3) and κ_1, κ_2 are some constants. It can be seen from the equation that (18) is a simple PI controller [16], where the proportional component helps improve system response time and limit the queue to lower levels (see simulations below).

We next discuss the rationale for the new source controller. The user equation in RCP and QI-RCP causes the input traffic rate to significantly overshoot link capacity and skyrocket the queue size when a flash crowd of flows joins the system. Instead, we desire a controller that does not immediately jump to the current fair bandwidth share in the network and uses caution during rate increase. To achieve this, new flows joining the system should gradually increase their sending rate towards the feedback provided by their bottleneck router.

For the discussion that follows, define $e_i(t) = R_l(t - D_i^{\leftarrow}) - x_i(t-T)$ to be the error between the previous rate of user i and the last rate suggested by the network, assuming the current bottleneck of user i is router l. Also denote by $\delta_i(t) = R_l(t - D_i^{\leftarrow}) - R_l(t - T - D_i^{\leftarrow})$ the difference between the two most-recent rates provided by the network to flow i. Then, the PIQI-RCP source controller is given by

$$x_i(t) = x_i(t-T) + \tau_1 e_i(t) + \tau_2 \delta_i(t), \qquad (19)$$

where τ_1 and τ_2 are constants that we determine below.

The first two terms in (19) constitute an *Exponentially Weighted Moving Average* (EWMA) controller. Rather than directly setting the sending rate $x_i(t)$ to the received feedback $R(t - D_i^{\leftarrow})$ as in (2), this controller changes the current sending rate in steps that take $x_i(t)$ toward $R_l(t)$. The $\delta_i(t)$ term performs a comparison of two successive feedback values to understand what the bottleneck router wants flows to do: if $\delta_i(t) > 0$, then the router is under-utilized and wants to encourage the flows to increase their sending rate; if $\delta_i(t) < 0$, then the router is over-utilized and wants the flows to decrease their sending rate. Including this information in the source controller makes it more responsive and reduces queue overshoot in the transient state.

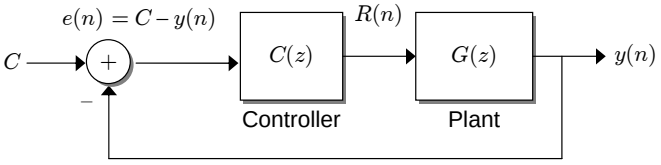


Fig. 7. Feedback control system model of explicit congestion control

It should be noted that comparison of successive feedback values is achieved by simply saving the last received feedback from the bottleneck and does not incur any network overhead. Also note that τ_2 affects the behavior of the system when the bottleneck link is in a transient state, i.e., when the successive feedback values are different. After the bottleneck controller has reached its steady state, $\delta_i(t) = 0$ and the performance of the source is governed only by τ_1.

Using the Nyquist stability criterion, we next show that PIQI-RCP can be easily stabilized assuming the maximum RTT D is known at the router. We also replace the general router controller with a simpler version that we use in practice by setting $\kappa_1 = \kappa_2 = \kappa$.

Theorem 2: Assume N flows with heterogeneous RTTs and define $D = \max\{D_1, \ldots, D_N\}$, $D' = \lceil D/T \rceil$. Then, the discrete PIQI-RCP system (18)-(19) with sufficiently small T is locally asymptotically stable if $0 < \tau_1 < 1$, $0 < \tau_1 + 2\tau_2 < 2$, and $0 < \kappa < \kappa^*$, where

$$\kappa^* = \sin\Big(\frac{\pi}{2(2D' - 1)}\Big). \qquad (20)$$

Furthermore, if flow RTTs are homogeneous (i.e., $D_i = D$ for all i), condition $0 < \kappa < \kappa^*$ is also necessary.

Proof: We first develop a linear control-theoretic model of PIQI-RCP and then analyze the model for stability using Nyquist stability criterion. Consider the feedback control system model of explicit congestion control as shown in Fig. 7. Let $G(z)$ be the *plant* consisting of N flows each with sending rate $x_i(t)$ and round-trip time (RTT) $D_i = D_i^{\rightarrow} + D_i^{\leftarrow}$. Let $C(z)$ be the router *controller* whose goal is to operate the closed loop system in a stable manner. The output of the plant is the total sending rate $y(n)$ arriving at the router controller and the input being the per-flow sending rate $R(n)$ generated by the router. The individual blocks, i.e., the controller and the plant, are analyzed below.

The linearized controller (18) in the discrete-time domain can be written as

$$R(n) = R(n-T) + \frac{\kappa e_l(n)}{N} + \frac{\kappa e_l(n-T)}{N}. \qquad (21)$$

Taking the z-transform of both sides of the equation, we get

$$R(z)[1-z^{-T}] \;=\; \frac{\kappa}{N}\left[1+z^{-T}\right]e(z). \tag{22}$$

After re-arranging the terms in the above equation, the transfer function $C(z) = R(z)/e(z)$ is

$$C(z) \;=\; \frac{\kappa(1+z^{-T})}{N(1-z^{-T})}, \tag{23}$$

The plant consists of N flows, each of which adjusts its sending rate using (19). In the discrete-time domain and after expanding the error terms, (19) can be written as

$$\begin{aligned} x_i(n) &= x_i(n-T) - \tau_1[x_i(n-T) - R_l(n-D_i^{\leftarrow})] \\ &+ \tau_2[R_l(n-D_i^{\leftarrow}) - R_l(n-T-D_i^{\leftarrow})] \\ &= (1-\tau_1)x_i(n-T) + (\tau_1+\tau_2)R_l(n-D_i^{\leftarrow}) \\ &- \tau_2 R_l(n-T-D_i^{\leftarrow}). \end{aligned} \tag{24}$$

The total input traffic rate $y_l(n)$ observed at the router is given by

$$\begin{aligned} y_l(n) &= \sum_{i=1}^{N} x_i(n-D_i^{\rightarrow}) = \sum_{i=1}^{N}[x_i(n-T-D_i^{\rightarrow})(1-\tau_1) \\ &+ (\tau_1+\tau_2)R_l(n-D_i) - \tau_2 R_l(n-T-D_i)] \\ &= (1-\tau_1)y(n-T) + (\tau_1+\tau_2)\sum_{i=1}^{N} R_l(n-D_i) \\ &- \tau_2 \sum_{i=1}^{N} R_l(n-T-D_i). \end{aligned} \tag{25}$$

Taking the z-transform of both sides of the above equation, the transfer function $G(z) = Y(z)/R(z)$ of the plant can be written as

$$G(z) = \frac{(\tau_1+\tau_2) - \tau_2 z^{-T}}{1-(1-\tau_1)z^{-T}} \sum_{i=1}^{N} z^{-D_i}. \tag{26}$$

The overall open loop transfer function $T_f(z) = C(z)G(z)$ combining the controller and the plant can be obtained from (23) and (26) as

$$\begin{aligned} T_f(z) &= \left[\frac{\tau_1+\tau_2+\tau_1 z^{-T} - \tau_2 z^{-2T}}{1-(1-\tau_1)z^{-T}}\right] \sum_{i=1}^{N} \frac{\kappa}{N} \frac{z^{-D_i}}{1-z^{-T}} \\ &= T(z)T_D(z), \end{aligned} \tag{27}$$

where

$$T(z) \;=\; \left[\frac{\tau_1+\tau_2+\tau_1 z^{-T} - \tau_2 z^{-2T}}{1-(1-\tau_1)z^{-T}}\right], \tag{28}$$

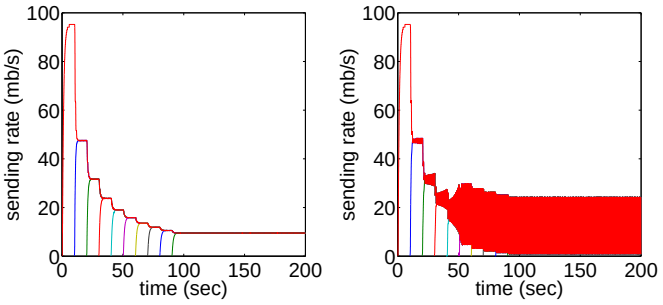
$$T_D(z) \;=\; \sum_{i=1}^{N} \frac{\kappa}{N} \frac{z^{-D_i}}{1-z^{-T}}. \tag{29}$$

In the frequency domain, we have

$$T_f(e^{j\omega}) \;=\; T_D(e^{j\omega})T(e^{j\omega}), \tag{30}$$

$$T(e^{j\omega}) \;=\; \left[\frac{\tau_1+\tau_2+\tau_1 e^{-j\omega T} - \tau_2 e^{-j\omega 2T}}{1-(1-\tau_1)e^{-j\omega T}}\right], \tag{31}$$

$$T_D(e^{j\omega}) \;=\; \sum_{i=1}^{N} \frac{\kappa}{N} \frac{e^{-j\omega D_i}}{1-e^{-j\omega T}}. \tag{32}$$



(a) $\eta = 0.99$ (b) $\eta = 1.01$

Fig. 8. Verification of stability of PIQI-RCP for flows with homogeneous RTT $D = 120$ ms and $\kappa = \eta\kappa^*$ ($T = 10$ ms, $\tau_1 = 0.005$, $\tau_2 = 0.5$, $\gamma = 0.95$).

Function $T(e^{j\omega})$ crosses the real axis when $\omega = \omega_1' = 0$ or $\omega = \omega_2'$ given by

$$\omega_2' = \left[\frac{\tau_1^2 - 2\tau_1 + 2\tau_1\tau_2}{T^2\tau_2(1-\tau_1)}\right]^{\frac{1}{2}}. \tag{33}$$

In the former case, $\text{Re}[T(e^{j\omega_1'})] = 2\kappa\tau_1/\tau_1 = 2\kappa$. In the latter case, for $0 < \tau_1 < 1$ and $\tau_1 + 2\tau_2 < 2$, the value of ω_2' is imaginary, which ensures that for this set of constants $T(e^{j\omega_2'})$ does not cross the real axis. Hence, selecting sufficiently small values of τ_1 and τ_2 that satisfy both inequalities above, one can enforce that $T(e^{j\omega})$ crosses the real axis only when ω equals ω_1'.

Further assuming small ωT (i.e., $T/D \approx 0$), $T_f(e^{j\omega})$ can be reduced to

$$T_f(e^{j\omega}) = \sum_{i=1}^{N} \frac{2\kappa\tau_1}{N\tau_1} \frac{e^{-j\omega D_i}}{1-e^{-j\omega T}} = \sum_{i=1}^{N} \frac{2\kappa}{N} \frac{e^{-j\omega D_i}}{1-e^{-j\omega T}}, \tag{34}$$

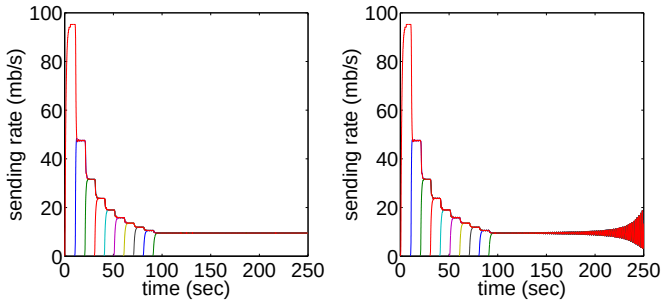
which using the analysis in the proof of Theorem 1 gives the upper bound on the stability region in (20). ■

Note that (20) is very similar to the condition on QI-RCP. The reduction by a factor of two is easy to explain since (18) has two error terms rather than one. The result in Theorem 2 shows that with τ_1, τ_2 sufficiently small, stability of PIQI-RCP can be reduced to that of the bottleneck controller (18).

D. Tightness of the PIQI-RCP Stability Condition

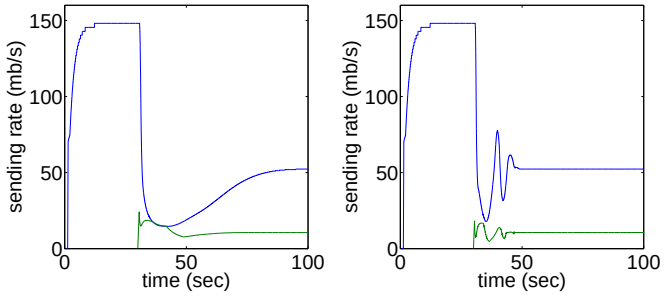
We next use `ns2` simulations to study how well the discrete model (18)-(19) of PIQI-RCP resembles the actual system. As before, each router keeps track of the maximum RTT in every control interval and sets $\kappa = \eta\kappa^*$, where κ^* is given by (20). As we discuss in the next section, the sine function in κ^* can be approximated with a much simpler equation and in our implementation none of the routers have to compute (20). However, to verify that the derived stability condition is accurate, this section uses the exact value of κ^*.

We start with the homogeneous case. Consider a dumbbell topology with a 100-mb/s bottleneck, delay 50 ms, and 1-gb/s access links. A new flow enters the system every 10 seconds and remains in the system for the entire duration of the simulation. The steady-state RTT of each flow is 120



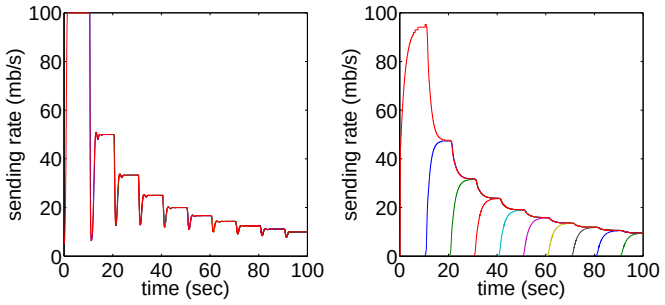
(a) $\eta = 0.99$ (b) $\eta = 1.01$

Fig. 9. Verification of stability of PIQI-RCP under heterogeneous RTTs $D_1 = 120, D_2 = \ldots = D_9 = 300$ ms and $\kappa = \eta\kappa^*$ ($T = 10$ ms, $\tau_1 = 0.005$, $\tau_2 = 0.5$, $\gamma = 0.95$).



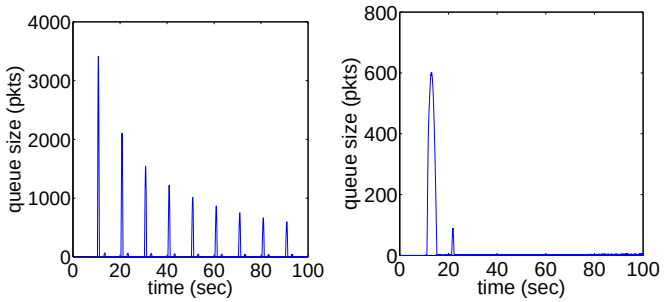
(a) $\eta = 0.5$ (b) $\eta = 0.99$

Fig. 10. Sending rate of PIQI-RCP in topology $\mathcal{T}_1$ for $\kappa = \eta\kappa^*$ ($T = 10$ ms, $\tau_1 = 0.005$, $\tau_2 = 0.5$, $\gamma = 0.95$).



(a) RCP (b) PIQI-RCP

Fig. 11. Sending rate of RCP and PIQI-RCP in a single-bottleneck network with heterogeneous RTTs $D_1 = 120, D_2 = \ldots = D_9 = 300$ ms and $\kappa = 0.95\kappa_*$ ($T = 10$ ms, $\tau_1 = 0.005$, $\tau_2 = 0.5$, $\gamma = 0.95$).



(a) RCP (b) PIQI-RCP

Fig. 12. Comparison of queue size for RCP and PIQI-RCP in a single-bottleneck network with heterogeneous RTTs $D_1 = 120, D_2 = \ldots = D_9 = 300$ ms and $\kappa = 0.95\kappa_*$ ($T = 10$ ms, $\tau_1 = 0.005$, $\tau_2 = 0.5$, $\gamma = 0.95$).

ms. Fig. 8 shows that PIQI-RCP is stable for $\eta = 0.99$ and unstable for $\eta = 1.01$, just as predicted by Theorem 2. For the heterogeneous case, access links have different delays such that $D_1 = 120$ ms and the other nine flows have RTT 300 ms. The corresponding plots are shown in Fig. 9 where the condition $\kappa < \kappa^*$ is both sufficient and necessary. This confirms our earlier observation that if RTTs are clustered close to D, the derived stability conditions become necessary.

We finally examine whether PIQI-RCP is stable in topology $\mathcal{T}_1$. Fig. 10 shows the sending rates of PIQI-RCP flows in $\mathcal{T}_1$ for both $\eta = 0.5$ and $\eta = 0.99$, where the former case is slower in convergence, but much smoother in terms of sending-rate dynamics.

V. Simulations

In this section, we compare the performance of RCP and PIQI-RCP in various simulation setups using `ns-2`. In most cases, RCP's parameters are set as in the default implementation (i.e., $\alpha = 0.4, \beta = 1$), all router intervals T are 10 ms, and PIQI-RCP's $\kappa_1 = \kappa_2$. In order to avoid computing the sine function inside routers, we replace the upper bound κ^* with a much simpler upper bound

$$\kappa_* = \frac{T}{2(T+D)} \leq \kappa^*, \tag{35}$$

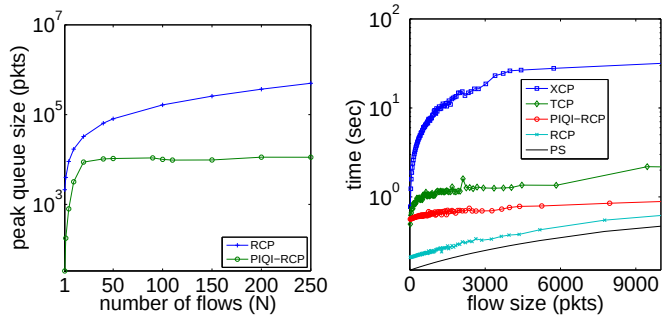
where D is the maximum RTT of end flows.

A. Single-Bottleneck Topology

Consider a dumb-bell topology with heterogeneous flows discussed in the previous section. The performance of RCP and PIQI-RCP with $\kappa = 0.95\kappa_*$ is shown in Fig. 11. In the steady state, all flows share the bottleneck link fairly regardless of their RTTs. For RCP, new flows join the system with a sending rate equal to the current control rate at the router, which leads to an overshoot of link capacity for every new flow entering the system. This in turn results in often drastic reductions in rate as seen in Fig. 11(a). The peak queue size for RCP is nearly $3,500$ packets, while that for PIQI-RCP is only 550 packets as shown in Fig. 12. Also observe that PIQI-RCP keeps the queue close to zero after the second flow has joined the system. Similar behavior is observed for bottleneck capacity of 1 and 10 gb/s (not shown for brevity).

B. Peak Queue Size

In this section, we compare the peak queue size of RCP with that of PIQI-RCP in the same simulation setup, except a single join event at $t = 15$ has N flows entering the system simultaneously. Fig. 13(a) shows that RCP performs significantly worse than PIQI-RCP, reaching over $501,000$ queued packets for $N = 250$ and scaling its buffer requirement super-linearly (note the log-scale of the y-axis). On the other hand, PIQI-RCP scales much better and stabilizes the queue at $10,000$ packets after $N = 25$. This implies that as N increases



(a) peak queue size (b) AFCT

Fig. 13. Comparison of PIQI-RCP to other methods ($T = 10$ ms, $\tau_1 = 0.005$, $\tau_2 = 0.5$, $\gamma = 0.95$).

beyond 250, the difference between RCP and PIQI-RCP will be more noticeable.

C. Average Flow Completion Time

We next compare RCP, PIQI-RCP, TCP, and XCP using average flow completion time (AFCT) as the main performance metric. AFCT is often compared to the smallest achievable average flow completion time known as *Processor Sharing* (PS) [6]. Intuitively, RCP will fare better in this scenario since new flows entering the system directly use the current control rate at the router, while new flows in PIQI-RCP, TCP, and XCP gradually increase their sending rate and hence require more time to complete. We confirm this in a simulation scenario from [5] consisting of a single-bottleneck link of capacity 2.4 gb/s and delay of 50 ms. New flows enter the system as Poisson arrivals with Pareto-distributed flow sizes with mean 30 packets (1000 bytes/pkt) and shape 1.4. The offered load is 90% of link capacity.

The corresponding AFCT for all studied methods are shown in Fig. 13(b), where the results are obtained from `ns2` simulations with infinite buffers.[4] Observe in the figure that PIQI-RCP is indeed slower than RCP, but nevertheless is faster than TCP and XCP. The difference between RCP and PIQI-RCP can be bridged by selecting a higher value of τ_1, which can be viewed as a tuning knob that controls the tradeoff between AFCT and the buffering requirement.

D. Multi-Bottleneck Link Topology

We finish simulations with a parking-lot topology in Fig. 14, where flow x_1 traverses two links of capacity 970 and 800 mb/s, respectively, and delay 50 ms each. Flow x_2 only traverses the first link and flow x_3 only the second. The three flows enter the system at $t = 0, 15, 30$ seconds, respectively. The sending rate of all flows for RCP and PIQI-RCP is shown in Fig. 15. Until $t = 15$, flow x_1 is bottlenecked at link l_2. At $t = 15$ when x_2 enters the system, the bottleneck of flow x_1 switches to link l_1 and both x_1 and x_2 have identical sending rates equal to 485 mb/s. At $t = 30$ when x_3 enters the system, x_1 switches its bottleneck to l_2 again, after which x_1 and

[4]RCP with finite buffers loses a lot of packets and performs much worse in terms of AFCT, which we do not show for brevity.

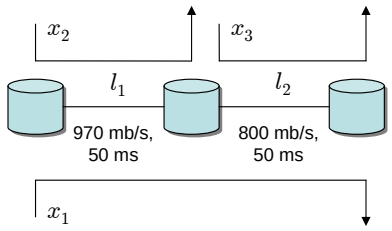
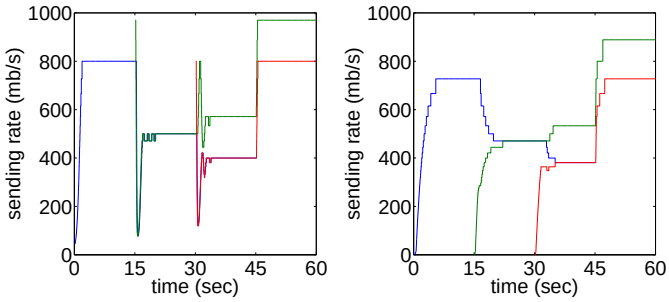


Fig. 14. Topology $\mathcal{T}_2$.



(a) RCP (b) PIQI-RCP

Fig. 15. Performance comparison of RCP and PIQI-RCP in parking-lot topology $\mathcal{T}_2$ under heterogeneous delay $D_1 = 200, D_2 = D_3 = 100$ ms and $\kappa = 0.95\kappa_*$ ($T = 10$ ms, $\tau_1 = 0.005$, $\tau_2 = 0.5$, $\gamma = 0.95$).

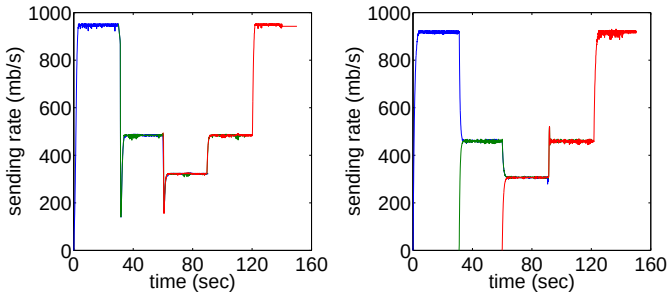
x_3 equally share link l_2 (i.e., rate 400 mb/s each). Flow x_2 captures the remaining bandwidth at link l_1, which is the max-min allocation of rates for this topology. As seen from the figures, the magnitude of transient oscillations is much smaller in PIQI-RCP compared to RCP, while the convergence time is almost the same.

VI. Linux Experiments

We also compare RCP and PIQI-RCP using our own Linux implementation in a gigabit Emulab [7] network. We use Linux kernel 2.6.12 to develop the end-host and router functionality of both RCP and PIQI-RCP, where each protocol is implemented as a module that can be dynamically plugged in/out of the running kernel without rebooting the system. We modify the TCP/IP stack to convert its original window-based data transfer to rate-based operation required by these protocols. We tune the default kernel parameters in order to support gigabit throughput for a wide-range of RTTs and fairly evaluate the limitations of each protocol. Specifically, we increase the maximum size of socket read/write buffers, per-connection memory space, backlog queue in the receive path, transmit queue in the forward path, and transmit/receive ring buffers of the network interface card. We set control parameters $\alpha = 0.1, \beta = 1$ for RCP and $\kappa = \kappa_*, \tau_1 = 0.01, \tau_2 = 0.1, \gamma = 0.95$ for PIQI-RCP. All routers use $T = 10$ ms for their control equations.

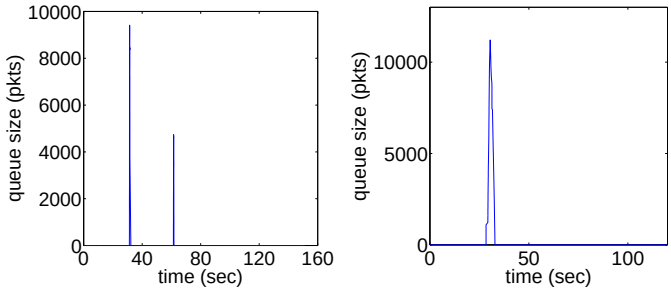
A. Single-bottleneck Topology

Consider a scenario where three flows pass through a single-bottleneck link of capacity 1 gb/s and round-trip propagation delay of 50 ms. Each flow is connected to the bottleneck link



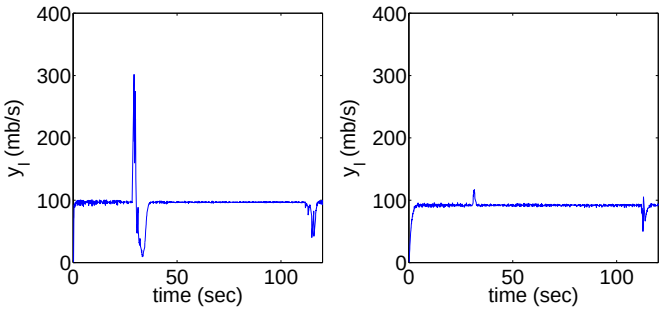
(a) RCP (b) PIQI-RCP

Fig. 16. Linux performance of three flows sharing a single-bottleneck link of capacity 1 gb/s and RTT 50 ms.



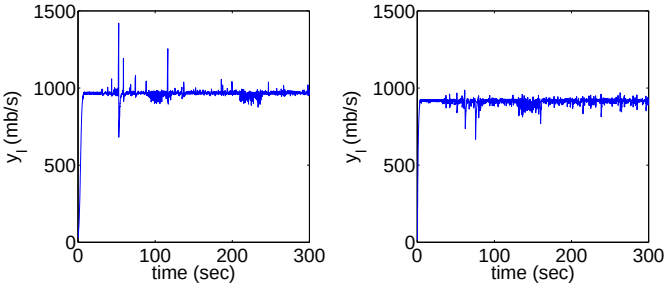
(a) single-bottleneck topology (b) abrupt change in traffic demand

Fig. 17. Linux queuing dynamics in RCP.



(a) RCP (b) PIQI-RCP

Fig. 18. Linux performance with abrupt change in traffic demand.



(a) RCP (b) PIQI-RCP

Fig. 19. Linux performance in the presence of background mice entering the system at $t = 30$.

through a different access link of capacity 1 gb/s and negligible delay. These flows start with a 30-second delay and each lasts for 90 seconds. The sending rates[5] of the flows are illustrated in Fig. 16, where both methods are able to maintain high throughput and keep CPU utilization below 30% at the router. During the experiment, we also monitor the IP layer queue size inside the bottleneck router. As shown in Fig. 17(a), RCP experiences significant queue buildup (up to $9,415$ packets) whenever a new flow joins the system. In contrast, PIQI-RCP maintains queue size of at most 1 packet (not shown in the figure), which can be explained by the gradual increase of user rates in the source controller. This allows the router enough time to converge to a new steady state without significantly overshooting link capacity.

B. Abrupt Change in Traffic Demand

In this experiment, we examine the performance of RCP and PIQI-RCP with an abrupt increase or decrease in traffic demand. We use a dumb-bell topology with gigabit access links and a bottleneck link with capacity 100 mb/s and delay 25 ms. At $t = 0$, one long flow is started for a duration of 120 seconds. At $t = 30$, another 10 flows abruptly join the network and continue to remain in the system until $t = 113$, at which time they all suddenly exit.

In RCP, at time $t = 30$ the average input traffic increases to 300 mb/s and the queue size jumps to around $11,000$ packets as shown in Fig. 17(b), which is very similar to the behavior of RCP in `ns2`. Next, with the rise in both the input traffic rate and queue size, the router's control equation overcompensates and drives the combined sending rate to 10 mb/s as shown in 18(a). This explains the drop in link utilization at $t = 30$. The system remains in this transient state for about 7 seconds before eventually recovering. As can be seen from Fig. 18(b), PIQI-RCP has a very small overshoot of C_l and all excess traffic gets absorbed in network device queues without increasing the router queue (i.e., the peak IP-layer queue is again 1 packet).

C. Performance with Mice Traffic

In this section, we study RCP and PIQI-RCP when the input traffic is a combination of both short-lived (i.e., mice) and long-lived flows. We start the long flow at $t = 0$ from one of the two sender machines and generate mice traffic starting at $t = 30$ from the other machine. All flows pass through a common bottleneck link of capacity 1 gb/s and delay of 25 ms. The pattern of mice traffic follows Poisson arrivals with the mean inter-arrival time of 0.2 seconds and Pareto-distributed duration (i.e., number of transferred packets) with shape parameter 1.4 and mean 100 packets.

The results of this experiment are shown in Fig. 19. In RCP, link utilization remains generally very high, but experiences occasional overshoots by 40% and undershoots by 25%. PIQI-RCP, on the other hand, demonstrates much less rate fluctuation, a virtually non-existent queue, and the same high link utilization as in RCP. This shows that PIQI-RCP is not

[5]The reported sending rate is the throughput at the IP layer using packet size of 1500 bytes.

only a stable protocol with good transient properties in theory, but also a practical approach suitable for GENI-style [19] networks of the future.

VII. Conclusion and Future Work

In this work, we found that RCP could become unstable in certain cases and required unrealistically large buffers to absorb transient overshoots of link capacity. As an alternative to RCP, we proposed a new controller called PIQI-RCP and showed that its heterogeneous stability could be easily established using common control-theory tools. We further demonstrated in simulations and experiments that PIQI-RCP required much smaller buffers at routers and had a lower average flow completion time compared to TCP and XCP.

Future work involves analysis of multi-link stability of max-min congestion control and deployment of explicit-feedback methods in large-scale testbeds with millions of flows.

References

[1] L. H. Andrew, K. Jacobsson, S. H. Low, M. Suchara, R. Witt, and B. P. Wydrowski, "MaxNet: Theory and Implementation," Netlab, Caltech, Tech. Rep., 2006. [Online]. Available: http://netlab.caltech.edu/maxnet/MaxNet_Implementation_TechReport.pdf.

[2] G. Appenseller, I. Keslassy, and N. McKeown, "Sizing Router Buffers," in *Proc. ACM SIGCOMM*, Aug. 2004, pp. 281–292.

[3] S. Bhandarkar, S. Jain, and A. L. N. Reddy, "Improving TCP Performance in High Bandwidth High RTT Links Using Layered Congestion Control," in *Proc. PFLDnet*, Feb. 2005.

[4] Cisco 12000 Series Gigabit Ethernet Line Cards. [Online]. Available: http://www.cisco.com/en/US/products/hw/modules/ps2710/ps186/.

[5] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown, "Processor Sharing Flows in the Internet," in *Proc. IEEE IWQoS*, Jun. 2005.

[6] N. Dukkipati and N. McKeown, "Processor Sharing Flows in the Internet," High Performance Networking Group, Stanford Univ., Tech. Rep. TR04-HPNG-061604, Jun. 2004.

[7] Emulab. [Online]. Available: http://www.emulab.net/.

[8] A. Falk, Y. Pryadkin, and D. Katabi, "Specification for the Explicit Control Protocol (XCP)," Oct. 2005, IETF Internet-draft.

[9] S. Floyd, "High-speed TCP for Large Congestion Windows," *IETF RFC 3649*, Dec. 2003.

[10] C. Jin, D. Wei, and S. H. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 2490–2501.

[11] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth Delay Product Networks," in *Proc. ACM SIGCOMM*, Aug. 2002, pp. 89–102.

[12] T. Kelly, "Scalable TCP: Improving Performance in High-speed Wide Area Networks," *Computer Communication Review*, vol. 33, no. 2, pp. 83–91, Apr. 2003.

[13] D. Leith and R. Shorten, "H-TCP Protocol for High-Speed Long Distance Networks," in *Proc. PFLDnet*, Feb. 2004.

[14] S. H. Low, L. L. H. Andrew, and B. P. Wydrowski, "Understanding XCP: Equilibrium and Fairness," in *Proc. IEEE INFOCOM*, Mar. 2005, pp. 1025–1036.

[15] L. Massoulié, "Stability of Distributed Congestion Control with Heterogeneous Feedback Delays," *IEEE Trans. Automat. Contr.*, vol. 47, no. 6, pp. 895–902, Jun. 2002.

[16] I. J. Nagrath and M. Gopal, *Control Systems Engineering*. John Wiley & Sons, 2004.

[17] ns2, "Network Simulator." [Online]. Available: http://www.isi.edu/nsnam/ns/.

[18] NSF FIND, "Future INternet Design." [Online]. Available: http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=12765&org=CNS.

[19] NSF GENI, "Global Environment for Network Innovations." [Online]. Available: http://www.nsf.gov/cise/cns/geni/.

[20] I. Rhee and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," in *Proc. PFLDnet*, Feb. 2005.

[21] Juniper M Series Multiservice Edge Routers. [Online]. Available: http://www.juniper.net/products_and_services/m_series_routing_portfolio/.

[22] B. P. Wydrowski, L. L. H. Andrew, and I. M. Y. Mareels, "MaxNet: Faster Flow Control Convergence," *Networking*, vol. 3042, pp. 588–599, May 2004.

[23] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One More Bit Is Enough," in *Proc. ACM SIGCOMM*, Aug. 2005, pp. 37–48.

[24] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control (BIC) for Fast, Long Distance Networks," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 2514–2524.

[25] Y. Zhang, S.-R. Kang, and D. Loguinov, "Delayed Stability and Performance of Distributed Congestion Control," in *Proc. ACM SIGCOMM*, Aug. 2004, pp. 307–318.

[26] Y. Zhang, D. Leonard, and D. Loguinov, "JetMax: Scalable Max-Min Congestion Control for High-Speed Heterogeneous Networks," in *Proc. IEEE INFOCOM*, Apr. 2006.