EVALUATION OF EXPLICIT CONGESTION CONTROL FOR HIGH-SPEED NETWORKS

A Thesis

by

SAURABH JAIN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2007

Major Subject: Electrical Engineering

EVALUATION OF EXPLICIT CONGESTION CONTROL FOR HIGH-SPEED NETWORKS

A Thesis

by

SAURABH JAIN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Co-Chairs of Committee,	Dmitri Loguinov
	A. L. N. Reddy
Committee Members,	Shankar Bhattacharyya
	Alexander Sprintson
Head of Department,	Costas Georghiades

May 2007

Major Subject: Electrical Engineering

ABSTRACT

Evaluation of Explicit Congestion Control

for High-Speed Networks. (May 2007)

Saurabh Jain, B.Tech., Indian Institute of Technology, Roorkee

Co–Chairs of Advisory Committee: Dr. Dmitri Loguinov
Dr. A. L. N. Reddy

Recently, there has been a significant surge of interest towards the design and development of a new global-scale communication network that can overcome the limitations of the current Internet. Among the numerous directions of improvement in networking technology, recent pursuit to do better flow control of network traffic has led to the emergence of several *explicit-feedback* congestion control methods. As a first step towards understanding these methods, we analyze the stability and transient performance of Rate Control Protocol (RCP). 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 simulations to show that the new method has significantly lower transient queue lengths, better transient dynamics, and tractable stability properties.

As a second step in understanding explicit congestion control, we experimentally evaluate proposed methods such as XCP, JetMax, RCP, and PIQI-RCP using their Linux implementation developed by us. Our experiments show that these protocols are scalable with the increase in link capacity and round-trip propagation delay. In steady-state, they have low queuing delay and almost zero packet-loss rate. We confirm that XCP cannot achieve max-min fairness in certain topologies. We find that

JetMax significantly drops link utilization in the presence of short flows with long flows and RCP requires large buffer size at bottleneck routers to prevent transient packet losses and is slower in convergence to steady-state as compared to other methods. We observe that PIQI-RCP performs better than RCP in most of the experiments.

To my parents

ACKNOWLEDGMENTS

I am sincerely grateful to Dr. Dmitri Loguinov for agreeing to guide my Master's thesis and allowing me to do research with him. This work would not have been possible without his constant motivation and guidance. Every bit of interaction with him has been a learning experience in some way or the other. His attitude and passion towards whatever he likes to do has always surprised and inspired me. I also thank him for making me part of the Internet Research Lab and the regular weekly seminars that helped me broaden my knowledge of computer networking.

I thank Dr. A. L. N. Reddy for agreeing to become the co-chair of my thesis committee and being so kind and patient whenever I needed his advice or help especially without prior appointments. I acknowledge Dr. Shankar Bhattacharyya and Dr. Alex Sprintson for being members of my thesis committee. I also thank all the faculty members with whom I had a chance to interact and gain knowledge.

I appreciate the help and company of all the members of the Internet Research Lab and friends in College Station. I am especially thankful to Rajanikant Maru for being a great friend and always being cheerfully available whenever I needed his help.

Last, but not least, I am indebted to my parents and family members for all their support and encouragement.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

The Internet has seen an explosive growth in the past two decades. Networking technologies that drive the Internet have improved significantly over the years. A number of industries, computational grids, and research laboratories own high-capacity networks spanning from one part of the globe to another for doing e-commerce, collaboration in research [48], data analysis [15], and data collection for geological and astronomical experiments. This enormous growth would not have been possible without the presence of congestion control in the Internet. Since the inception of the Internet, congestion control has always been a hot topic among researchers. The reasons being the importance and complexity of the problem, heterogeneity in networking environments, and the scope of more improvements.

The evolution of the Internet has brought concerns [12], [22], [28], [31], [53] among the research community on the effectiveness of currently deployed TCP-based congestion control methods in accelerating or maintaining the same level of growth as seen so far. A number of initiatives [43], [44] are currently under way towards the design and development of the *future* Internet that is scalable and more robust. As regard to this, recent efforts to design better congestion control have led to the origin of several *explicit-feedback* congestion control methods [6], [25], [51], [55], [56]. These methods solicit active multi-byte feedback from the routers to the end-hosts delivering a precise and timely congestion signal that is used to accurately adjust flow sending rates and hence achieve faster convergence, smaller packet loss rate, high link utilization, and better fairness between flows. For scalability, feedback is computed in

---

The journal model is *IEEE Transactions on Automatic Control*.

a distributed fashion with minimal processing and negligible data overheads without keeping per-flow state.

A. Objective

This thesis aims to advance research in explicit congestion control for high-speed networking. We strive to analyze the proposed explicit congestion control methods [6], [25], [51], [55], [56] based on max-min fairness [36] using theoretical insight and simulation tools. Our goal is to find performance deficiencies with the proposed methods and suggest modifications to improve their behavior. Apart from the steady-state properties, we also lay due emphasis on transient performance and system stability.

Current simulation studies [6], [25], [56] have shown that some of the proposed explicit congestion control methods have the potential to provide a scalable framework towards the design [43], [44] of a flexible and global-scale communication network that is better than the Internet today. However, experimental assessment and deployment experience with these approaches especially in high-capacity networks and multi-link settings is still unavailable in the current literature. Considering the importance of experimental studies, this thesis aims to fill this void by investigating the behavior of these methods in single and multi-link topologies involving practical systems and high-speed networks.

B. Contributions of This Work

As part of this work, we find that existing literature lacks a thorough analysis of Rate Control Protocol (RCP) [6] regarding its drawbacks. Our study shows that RCP can behave in an unstable manner in certain topologies and scenarios when there are flows with highly heterogeneous round-trip times (RTTs). We also find that

RCP has a very high buffering requirement inside routers in order to prevent a large number of packet losses. We propose *Queue Independent RCP* (QI-RCP) to address the stability issue with RCP. However, we find that QI-RCP also requires undesirably large buffer space inside routers. To address this drawback, we propose *Proportional Integral Queue Independent RCP* (PIQI-RCP) and mathematically prove its stability in single-bottleneck link topologies for flows with heterogeneous RTTs. Comparison of RCP and PIQI-RCP using ns-2 [42] simulations in both single and multiple-bottleneck link topologies show that PIQI-RCP is successful in reducing the peak queue size when there is a flash crowd of flows entering the system simultaneously and has stability conditions that can easily be satisfied inside routers.

This work also examines the performance of Explicit Control Protocol (XCP) [25], JetMax [56], RCP, and PIQI-RCP in an experimental environment using real systems and gigabit networks. Existing experimental studies in this regard are either completely missing (e.g., in the case of RCP) or have been limited to 10 mb/s bottleneck links [47], [54]. We implement these protocols in the existing Linux TCP/IP stack [33] in a manner that does not require modifications to existing TCP-based applications to use these protocols for flow control in the network. Normally, rate-based protocols have been implemented as application-level libraries using UDP [21], [56] or both UDP and TCP connections [46]. Instead, we develop rate-based data transmission functionality inside the Linux kernel required for JetMax, RCP, and PIQI-RCP to facilitate their deployment and fairly compare them with window-based XCP.

Our experiments in Emulab [9] using a variety of network configurations demonstrate behavior of these protocols. We confirm that XCP can maintain high link utilization and provides low queuing delay, but it cannot achieve max-min fairness in all topologies and has the highest number of per-packet computations inside the routers. JetMax performs well is almost all scenarios using long flows, has the least

buffer size requirement, and does the least number of per-packet computations inside the routers. However, JetMax looses link utilization in the presence of mice flows. RCP is able to maintain high link utilization in most of the traffic scenarios and does a reasonable amount of per-packet computations. However, it needs dramatically large buffer space to avoid large number of packet losses and shows significant oscillations in sending rate with abrupt increase or decrease in traffic demand. PIQI-RCP retains most of the strengths of RCP, has significantly lower transient queue lengths, and is more robust to abrupt surge in traffic demand.

C. Thesis Organization

The rest of the thesis is organized as follows. In Chapter II, we briefly describe the past developments in the field of congestion control. While Chapter III depicts the limitations of RCP, we design and analyze a new rate control protocol in Chapter IV. Chapter V and VI describe our Linux implementation methodology and the experimental results respectively. Finally, Chapter VII gives our conclusions and scope of future work.

CHAPTER II

BACKGROUND AND RELATED WORK

In this chapter, we first introduce congestion control and discuss its ideal properties. Since the field of congestion control has been an active area of research for the past two and half a decade, we next briefly highlight the past developments in this field. Finally, we describe the proposed congestion control methods that include the focus of this work.

A. Congestion Control

Assume $x_1(t), x_2(t), \ldots, x_N(t)$ be the sending rates of N flows with round-trip time (RTT) $D_1, D_2, \ldots, D_N$. The path of a flow in the network consists of a number of intermediate hops also called *routers* or *links*. The path of all the flows in the network can be expressed by *routing matrix* R, where $R_{il} = 1$ indicates that flow i passes through link l. The set of flows passing through the link l is represented by $i \in l$ while the set of links in the path of flow i is represented by $l \in i$. The forward/backward delays of flow i to/from the link l is denoted by $D_{il}^{\rightarrow}$ and $D_{il}^{\leftarrow}$ respectively. With these settings, *congestion control* can be defined as a way of regulating the sending rates of flows to operate the network within certain constraints. The ability to achieve these constraints determines the performance of a congestion control method. Some of the definitions required to understand the ideal properties of a congestion control method are described as follows:

- *Input Traffic Rate:* The input traffic rate $y_l(t)$ at link l is defined as the sum of

sending rates of flows passing through the link. It can be expressed as:

$$y_l(t) = \sum_{i \in l} x_i(t - D_{il}^{\rightarrow}), \tag{2.1}$$

where $i \in l$ is the set of flows passing through link l.

- *Link Capacity:* The capacity C_l of link l determines how fast the link can process or forward incoming packets of flows passing through the link.

- *Queue Length:* Any instantaneous increase in the input traffic rate compared to the link capacity causes queuing of packets. The instantaneous length of queue $q_l(t)$ at link l is called queue length.

- *Packet Loss Rate:* Once the queue is full, overshoot in input traffic rate compared to the outgoing link capacity causes loss of packets. The packet loss rate $p_l(t)$ at link l can be expressed as:

$$p_l(t) = \left[\frac{y_l(t) - C_l}{y_l(t)}\right]^+. \tag{2.2}$$

  The steady-state packet loss rate is the long term average number of packets lost in the network, while transient loss rate is the fraction of packets lost before flows converge to their steady state.

- *Efficiency:* Efficiency is the ability of a congestion control method to keep the average input traffic rate $\bar{y}_l$ at link l close to link capacity C_l.

$$\bar{y}_l = \lim_{T \to \infty} \frac{1}{T} \int_0^T y_l(t)dt \approx C_l \tag{2.3}$$

- *Fairness:* Fairness is a very broad term. However, for single-bottleneck link, fairness means that all flows have identical average sending rate in the steady-

state, i.e., $\bar{x}_i = \bar{y}_l/N$, where

$$\bar{x}_i = \lim_{T\to\infty} \frac{1}{T} \int_0^T x_i(t)dt. \tag{2.4}$$

For multiple-bottleneck links, fairness is usually complex to define. Different ways of defining fairness include majority fairness (Jain's fairness index) [4], proportional fairness [27], max-min fairness [36].

- *Stability:* For this work, we assume control-theoretic stability, i.e., the sending rate of flow i converges to its equilibrium value x_i^*.

$$\lim_{t\to\infty} x_i(t) = x_i^* \tag{2.5}$$

- *Convergence Rate:* Convergence rate is a measure of how fast the system converges to its steady state after any perturbations. Convergence to efficiency means how soon the input traffic rate converges to the link capacity. Convergence to fairness means how soon all the flows passing through the link equally share the link capacity.

B. Ideal Congestion Control

As shown in Fig. 1, an ideal congestion control method should be efficient and fair. It should maintain a high link utilization and ensure that all the flows are equally sharing the link capacity. It should try to maintain zero steady-state and transient packet loss rate in order to avoid unnecessary retransmissions. It is also desirable that the method has fast convergence rate independent of link capacity and round-tip propagation delay. Another important feature of an ideal congestion control method is to maintain stability in the system without any oscillations in the sending rate of the flows.

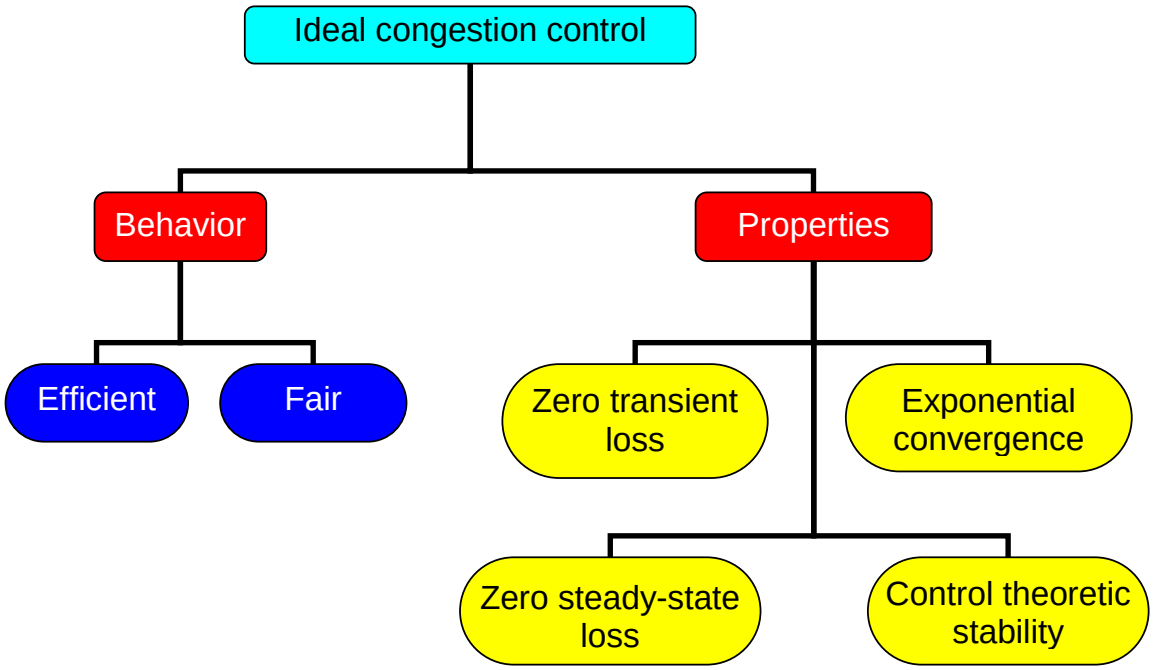


Fig. 1. Behavior and properties of ideal congestion control.

C. Feedback in Congestion Control

Congestion control is a closed-loop feedback control system, where flows in the network respond to the congestion feedback in order to adjust their sending rates. Congestion feedback can be *implicit* in nature such as detections of loss of a packet or increase in RTT due to larger queuing delays. Congestion feedback can also be *explicit* in nature with the support from the routers. Explicit feedback can be single-bit in nature using the ECN bit [45] in TCP/IP headers or multi-bit in nature such as change in the congestion window [25], traffic load factor [52], link price [51], desired sending rate [7], packet loss rate [55], and estimated fair rate [56].

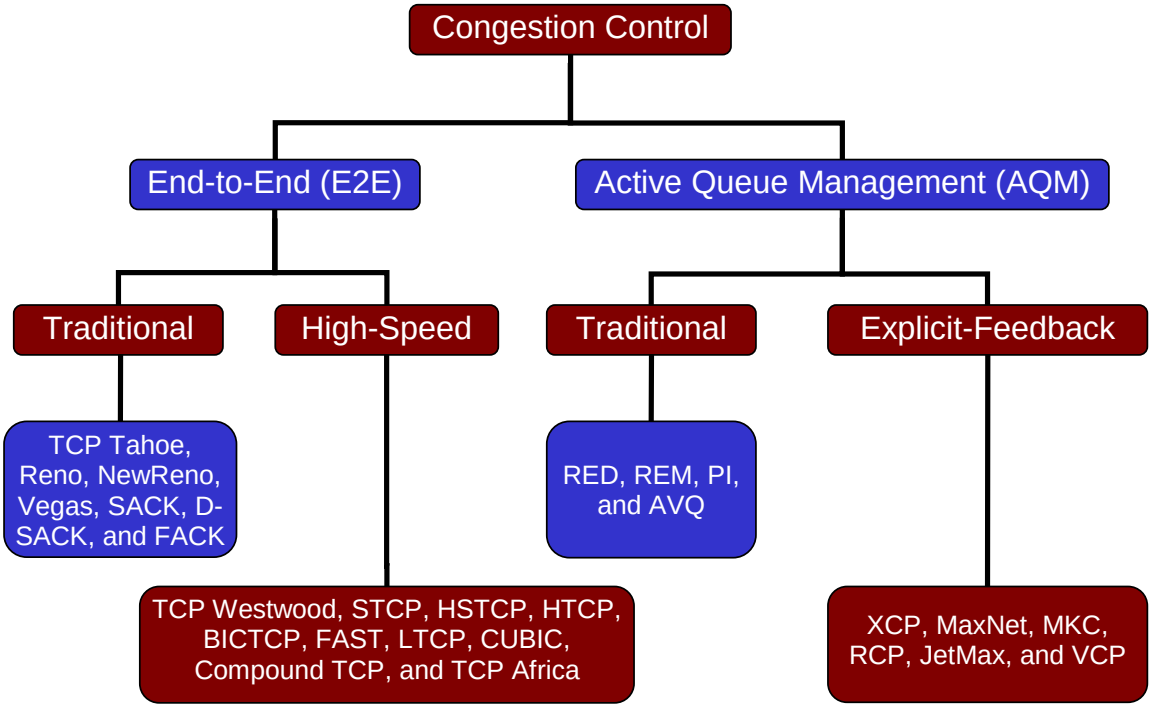


Fig. 2. Past developments in network congestion control.

D. The Big Picture

In 1983, the Internet moved to TCP/IP networking because of the many advantages associated with it. As documented in [39], Nagle observed congestion in the network as early as 1983. Since then and till today, congestion control has been an active area of research. During this vast period, a number of congestion control protocols have been proposed. As shown in Fig. 2, these protocols can be broadly divided into two different categories: a) End-to-End Congestion Control; b) Active Queue Management. They are briefly discussed in the following sections.

1. End-to-End Congestion Control

Congestion control methods that do not rely on routers and use implicit feedback to detect congestion are called *end-to-end* (E2E) congestion control methods. Some of

the traditional methods in this category are shown in Fig. 2 with TCP Reno being the most widely deployed in the Internet. These methods are known to lack scalability [12] with the increase in either bandwidth or delay (or both) in the network. Hence, as shown in Fig. 2 a number of methods for high-speed networking have recently been proposed. Some important ones include STCP [28], HSTCP [12], BIC-TCP [53], HTCP [31], FAST [22], [23], and LTCP [3]. Similar to TCP Reno, most of these methods are based on packet loss as a sign of congestion except for FAST that is based on increase in queuing delay or RTT. All of these protocols are based on end-to-end semantics and have only sender-side modifications. This facilitates their deployment in the Internet. In fact, most of these methods already have implementations that are part of the modern Linux kernel (starting with release 2.6.13). A comparative experimental evaluation available in [32] shows that these methods are able to provide high link utilization. However, considering other performance metrics of an ideal congestion control algorithm they are only slightly better or even worse compared to TCP Reno.

2. Active Queue Management

Active queue management (AQM) is a way of involving routers to aid the end-to-end congestion control methods. In AQM, a control algorithm runs at the routers that aims to provide more accurate and early congestion feedback to the end-hosts. As shown in Fig. 2, some of the proposed traditional AQM methods include Random Early Discard (RED) [13], [14], Random Early Marking (REM) [2], Proportional Integral Controller (PI) [18], [19], and Adaptive Virtual Queue (AVQ) [29], [30]. Most of these methods either implicitly drop packets or mark the ECN bit [45] in the TCP/IP headers to provide early congestion warning. However, it has been shown in [35] using control theory that these methods are prone to instability as capacity or

delay (or both) increases in the network. It has also been claimed that it is unlikely that any AQM scheme can operate in a stable manner over high-capacity and large-delay networks.

Another research direction using control-theoretic principles called *explicit* congestion control that has recently been considered is to use multi-byte explicit-feedback from the routers. In explicit congestion control, network devices in the path feedback more accurate multi-byte congestion information to the end-hosts so that they can adjust their congestion window size $W_i(t)$ or sending rate $x_i(t)$ more accurately. Each router l does per-packet processing to compute the combined traffic rate and apply this information in a control equation to generate feedback signal $p_l(t)$. Types of feedback suggested in the past include changes in the congestion window [25], traffic load factor [52], link price [51], desired sending rate [7], packet loss [55], and estimated fair rate [56]. Congestion feedback $\eta_i(t)$ received by flow i can be of the following two forms:

a. Additive Feedback

In the case of additive feedback, the congestion-feedback signal $\eta_i(t)$ received by flow i is the sum of feedback $p_l(t)$ generated by all the routers in the path of a flow.

$$\eta_i(t) = \sum_{l \in i} p_l(t - D_{il}^{\leftarrow}) \tag{2.6}$$

Additive feedback has been used in the traditional Kelly's model [27] where flow i adjusts its sending rate $x_i(t)$ as:

$$x_i(t) = x_i(t-1) + \kappa_i\Big(\omega_i - x_i(t - D_i)\sum_{l \in i} p_l(t - D_{il}^{\leftarrow})\Big), \tag{2.7}$$

where $x_i(t-1)$ is the previous sending rate, $x_i(t - D_i)$ is the sending rate one RTT

ago, ω_i is the price flow i is willing to pay, and κ_i is the control gain parameter. Such a system achieves *proportional* fairness [26] and has been proved in [27] to be globally asymptotically stable in the absence of delay in the network. Using discrete-time analysis, the necessary and sufficient conditions for local stability of the system of users (2.7) in the presence of homogeneous delays has been derived in [24] and a similar conjecture has been proposed for the sufficient condition for local stability in the presence of heterogeneous or diverse delays. A proof of the conjecture also using discrete-time analysis has been provided in [49]. Using continuous-time analysis and in the presence of heterogeneous delays, the conjecture in [24] has been proved in [50] by applying the generalized Nyquist stability criterion [5]. A slightly weaker version of the same conjecture and using continuous-time analysis has been proved in [37].

b. Max-min Feedback

In the case of max-min feedback, metric $p_l(t)$ of only the most congested router l in flow i's path is echoed back to the end-host as congestion feedback $\eta_i(t)$ and used to adjust the congestion window or sending rate.

$$\eta_i(t) = \max_{l \in i} p_l(t - D_{il}^{\leftarrow}) \tag{2.8}$$

Max-min fairness [36] is based on using max-min feedback. A number of explicit congestion control protocols for max-min fairness have been recently proposed. Some of them have been summarized in the following section.

E. Explicit Congestion Control for Max-min Fairness

In this section, we describe the proposed explicit congestion control methods for max-min fairness. These methods include the focus of this thesis.

1. XCP

Explicit Control Protocol (XCP) [25] is a window-based explicit congestion control method that uses a decoupled efficiency controller (EC) and fairness controller (FC) inside the router. EC generates the desired *aggregate* change $\phi(t) = \alpha d S(t) - \beta Q(t)$ in the congestion window for all flows, where α and β are constants, d is the average RTT, $S(t)$ is the available bandwidth, and $Q(t)$ is the persistent queue size at the bottleneck link. FC then translates $\phi(t)$ into per-packet feedback $H_i(k)$, which is conveyed in the k-th ACK of flow i. Upon arrival of each ACK, flow i sets its congestion window $W_i(k)$ according to:

$$W_i(k) = \max\big(W_i(k-1) + H_i(k), s\big), \tag{2.9}$$

where s is the packet size and $W_i(k)$ is flow i's window size after receiving ACK k. For homogeneous delay, it is shown in [25] that XCP is stable in single-bottleneck topology if $0 < \alpha < \pi/4\sqrt{2}$ and $\beta = \alpha^2\sqrt{2}$. The suggested values [25] of control parameters are $\alpha = 0.4$ and $\beta = 0.226$.

The stability analysis in the presence of flows with heterogeneous RTTs is not available for XCP. It has been proved in [34] that XCP may be arbitrarily max-min unfair in certain network topologies and an improper choice of α and γ may lead to low link utilization. In [54], it is shown that XCP can become unstable if there is inadequate buffer provisioning at routers in the path of a flow. It is also shown that lack of correct estimation of link capacity can prevent XCP from settling at zero steady-state error. As found in [6], the average flow completion time for XCP is higher than that of TCP. A recent study [56] shows that XCP can have a very high convergence time when there are flows with highly heterogeneous RTTs in networks with small bottleneck link capacity.

2. MaxNet

MaxNet [51] is another window-based explicit congestion control method. Each router l uses an integrator process to compute feedback $p_l(t)$ as:

$$p_l(t) = p_l(t-T) + \frac{y_l(t) - \gamma C_l}{\gamma C_l}, \tag{2.10}$$

where $y_l(t)$ is the input traffic rate, C_l is the link capacity, T is the control interval, and γ is the target link utilization. The sending rate of flow i is governed by an explicit demand function $D_i()$ of the received congestion-feedback $\eta_i(t) = \max_{l \in i} p_l(t - D_{il}^{\leftarrow})$. For a logarithmic utility function $U_i(x_i) = K_i \log(x_i)$, flow i updates its congestion window $W_i(k-1)$ upon receiving the k-th ACK as:

$$\xi_i(k) = \xi_i(k-1) + \beta_i \left(\frac{K_i}{W_i(k-1)} - \frac{\eta_i(k)}{D_i} \right) T \tag{2.11}$$

$$W_i(k) = W_{m,i} \exp\left(\xi_i(k) - \frac{\alpha_i \eta_i(k)}{M_i D_i} \right), \tag{2.12}$$

where K_i, α_i, β_i, and M_i are constants, $W_i(k)$ is the congestion window after receiving the k-th ACK, D_i is the RTT of flow i, $\eta_i(k)$ is the received congestion signal, T is the router control interval, $W_{m,i}$ is a large constant, and $\xi_i(k)$ is the value of a state-variable of flow i after receiving the k-th ACK.

3. MKC

Max-min Kelly Control [55] is obtained by modifying Kelly's equation (2.7) for max-min fairness. Flow i adjusts its sending rate $x_i(n)$ using

$$x_i(n) = (1 - \beta \eta_i(n)) x_i(n - D_i) + \alpha, \tag{2.13}$$

where α and β are constants, $x_i(n-D_i)$ is the sending rate one RTT ago, and received congestion-feedback $\eta_i(n)$ is:

$$\eta_i(n) = \max_{l \in i} p_l(n - D_{il}^{\leftarrow}). \tag{2.14}$$

MKC has been proved to be stable under arbitrary and including random delay in the network. Exponential MKC (EMKC) uses feedback $p_l(n)$ as the packet-loss rate at link l as:

$$p_l(n) = \frac{y_l(n) - C_l}{y_l(n)}, \tag{2.15}$$

where $y_l(n)$ is the input traffic rate and C_l is the link capacity. EMKC is stable in the case of flows with heterogeneous RTTs if $0 < \beta < 2$. However, the system has a non-zero steady state packet-loss rate and slow convergence speed.

4. RCP

Rate Control Protocol (RCP) [6] is a rate-based explicit congestion control scheme that aims at emulating processor sharing irrespective of traffic characteristics and network conditions. Each router l computes the desired sending rate $R_l(t)$ for flows bottlenecked at l using a controller

$$R_l(t) = R_l(t-T)\left[1 + \frac{T\left(\alpha(C_l - y_l(t)) - \beta\frac{q_l(t)}{d_l}\right)}{d_l C_l}\right], \tag{2.16}$$

where α and β are constants, $y_l(t)$ is the input traffic rate, d_l is the moving average of RTTs sampled by router l, C_l is its capacity, $q_l(t)$ is the instantaneous queue length at time t, and T is the router control interval. There is no mathematically tractable stability analysis of RCP for flows with homogeneous or heterogeneous RTTs. This is because of the presence of a queue term in the control equation that makes the model complicated. However, using MATLAB [38] and ns-2 [42] simulations and

under several choices of delays, the authors in [6] have developed a stability region for values of α and β to operate the system in a stable manner.

5. JetMax

JetMax [56] is another rate-based protocol, in which flow i adjusts its sending rate $x_i(n)$ using

$$x_i(n) = x_i(n - D_i) - \tau\big(x_i(n - D_i) - g_l(n - D_i^{\leftarrow})\big), \tag{2.17}$$

where $0 < \tau < 2$ is the gain parameter and network feedback $g_l(n)$ is the estimated fair rate at the bottleneck:

$$g_l(n) = \frac{\gamma_l C_l - u_l(n)}{N_l(n)}. \tag{2.18}$$

At time n, $N_l(n)$ is the total number of flows bottlenecked at l and $u_l(n)$ is the aggregate rate of flows receiving feedback from routers other than l. Mathematical analysis shows that for the single-link case, the number of control steps required to reach $(1-\epsilon)$-efficiency and $(1-\epsilon)$-fairness is $\lceil \log_{1-\tau} \epsilon \rceil$. Also, to guarantee monotonic convergence the condition $0 < \tau < 1$ must be satisfied. JetMax also uses a number of protocol enhancements such as *proposed rate* in order to prevent transient overshoot when a) a new flow enters the system; b) bottleneck switching has been detected; c) when a host demands a higher sending rate. It is shown in [56] that JetMax achieves max-min fairness, zero packet loss, and constant convergence speed to both efficiency and fairness.

6. Others

Other explicit congestion control methods include Variable-structure congestion Control Protocol VCP [52]. VCP uses the two explicit congestion notification (ECN) bits of the IP header. The control algorithm at the router samples input traffic rate

and queue level to designate whether the system is in low-load, high-load, or over-load conditions. This information is passed to the end-hosts using the ECN bits. Flows operate in the Multiplicative-Increase mode when the system is in low-load condition, Additive-Increase mode when the system is in high-load condition, and Multiplicative-Decrease mode when the system is in over-load condition.

CHAPTER III

ANALYSIS OF RCP

In this chapter, we demonstrate several drawbacks of Rate Control Protocol (RCP) [6] and suggest possible ways to eradicate them. We attribute the causes of these drawbacks to lack of mathematically-tractable stability analysis and an aggressive control equation at the end-host. We finally summarize the known strengths of RCP.

A. Drawbacks

We categorize the drawbacks of RCP in the following two sections: a) Instability; b) High Buffering Requirement

1. Instability

In RCP, the control equation at router l is given as

$$R_l(t) = R_l(t-T)\left[1 + \frac{T\left(\alpha(C_l - y_l(t)) - \beta\frac{q_l(t)}{d_l}\right)}{d_l C_l}\right], \tag{3.1}$$

where α and β are constants, $R_l(t)$ is the control rate, T is the control interval, C_l is the link capacity, $y_l(t)$ is the input traffic rate, $q_l(t)$ is the queue length, and d_l is the average round-trip time (RTT) of flows passing through the router. The simulation and stability analysis of RCP in the case of flows with only *homogeneous* RTTs is given in [7]. For flows with heterogeneous RTTs, the stability analysis is available only using simulations for various choices of delays. The presence of queue term $q_l(t)$ in (3.1) makes the analysis difficult because of the difficulty in modeling it. The evolution of queue $q_l(t)$ is discontinuous in nature (the discontinuity lies at $q_l(t) = 0$) and hence not differentiable. Also, the interaction of the queue size and average RTT

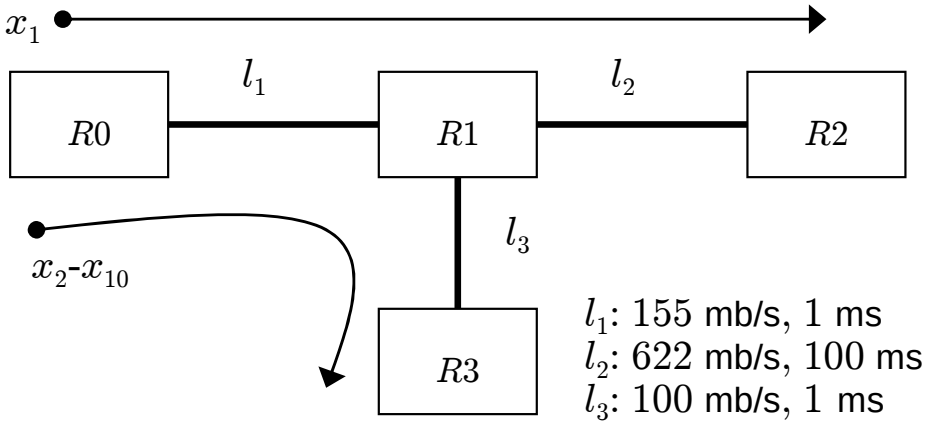


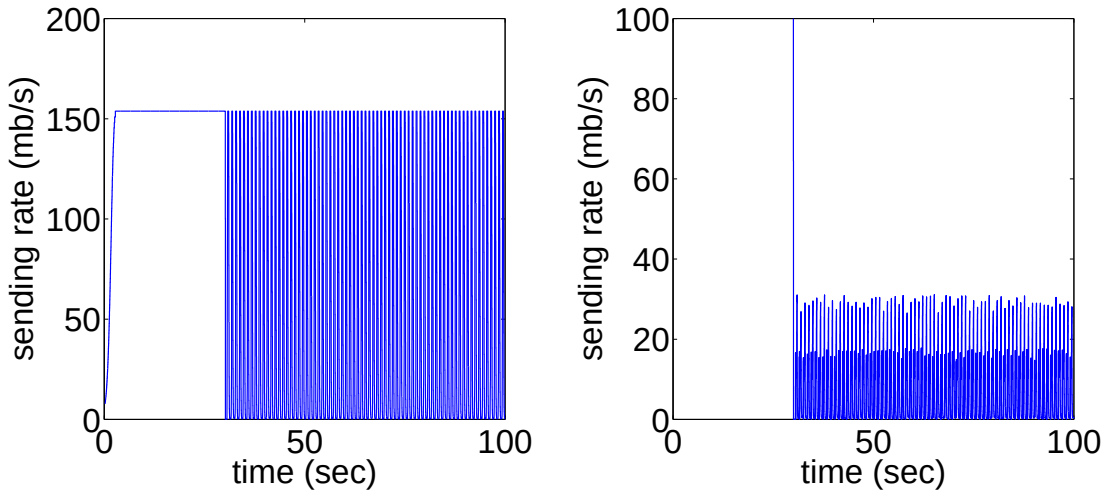
Fig. 3. Topology $\mathcal{T}_u$.

d_l in (3.1) is difficult to perceive. Some of these intricacies were neglected by the authors during their study in [7] and hence RCP can behave in an unstable manner in certain topologies and scenarios. One such scenario is shown next.

Consider the topology $\mathcal{T}_u$ as shown in Fig. 3. Link l_1 has capacity of 155 mb/s, delay 1 ms, and bottleneck id 0. Link l_2 has capacity of 622 mb/s, delay 100 ms, and bottleneck id 2. Link l_3 has capacity of 100 mb/s, delay 1 ms, and bottleneck id 4. Flow x_1 passes through links l_1 and l_2 while flows $x_2 - x_{10}$ traverse links l_1 and l_3. At $t = 0$, flow x_1 starts. At $t = 30$, flows $x_2 - x_{10}$ join the system. The simulation was performed using ns-2 simulation code and scripts for long flows as provided by the RCP authors.

a. RCP

The behavior of RCP in the simulation setup described above is shown in Fig. 4. Till $t = 30$, when there is only one flow x_1, with RTT 202 ms, in the network, the system behaves in a stable manner. The bottleneck link l_1 is completely utilized and the queue length is zero packets. The control rate at all the routers and the sending rate of x_1 are stable. However, the control rate and queue size at links l_1 and l_3 start



(a) x_1 (b) $x_2 - x_{10}$

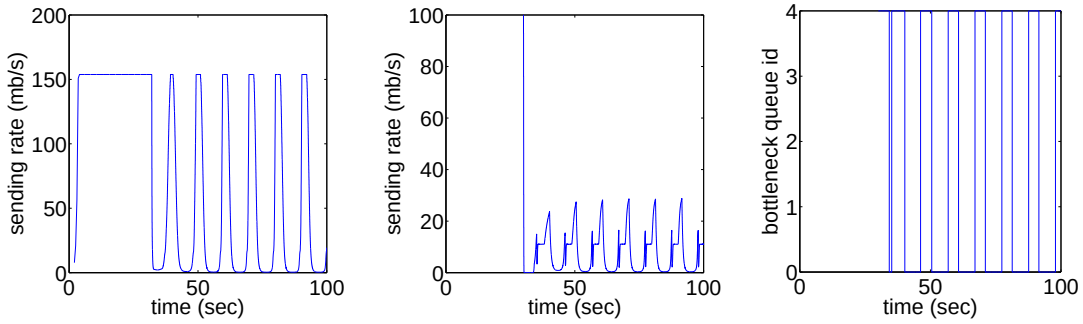
Fig. 4. Sending rate of flows $x_1 - x_{10}$ in the case of RCP for topology $\mathcal{T}_u$ confirming instability.

oscillating after 9 flows x_2 to x_{10}, with RTT 4 ms, enter the system at $t = 30$. This also causes the sending rate of flow x_1 and $x_2 - x_{10}$ to oscillate. From the figure, it is clear that the system is behaving in an unstable manner.

b. RCP with Higher Link Delays

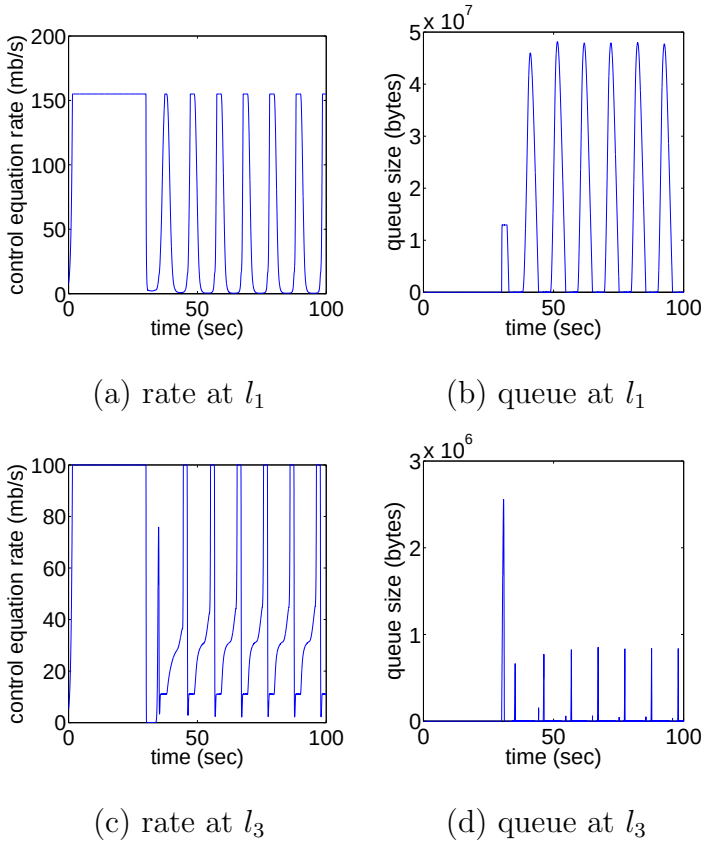
We increase the delay of all the links in topology $\mathcal{T}_u$ by a factor of 10, i.e., make the delay of link l_1 and l_3 to be 10 ms and that of link l_2 to be 1000 ms. As a result, the round-trip propagation delay of flow x_1 becomes 2.02 s and that of flow $x_2 - x_{10}$ becomes 40 ms. We repeat the simulation to confirm that instability in RCP is not an artifact of only one set of link delay. The sending rate of flows $x_1 - x_{10}$ shown in Fig. 5 and the control rate at links l_1 and l_3 shown in Fig. 6 keeps oscillating. Clearly, RCP again behaves in an unstable manner.

In this scenario, we examine average RTT d_l at bottleneck links l_1 and l_3 as shown in Fig. 7. We also examine the RTT of the end flows as shown in Fig. 7. It



(a) rate of x_1 (b) rate of $x_2 - x_{10}$ (c) bottleneck id of $x_2 - x_{10}$

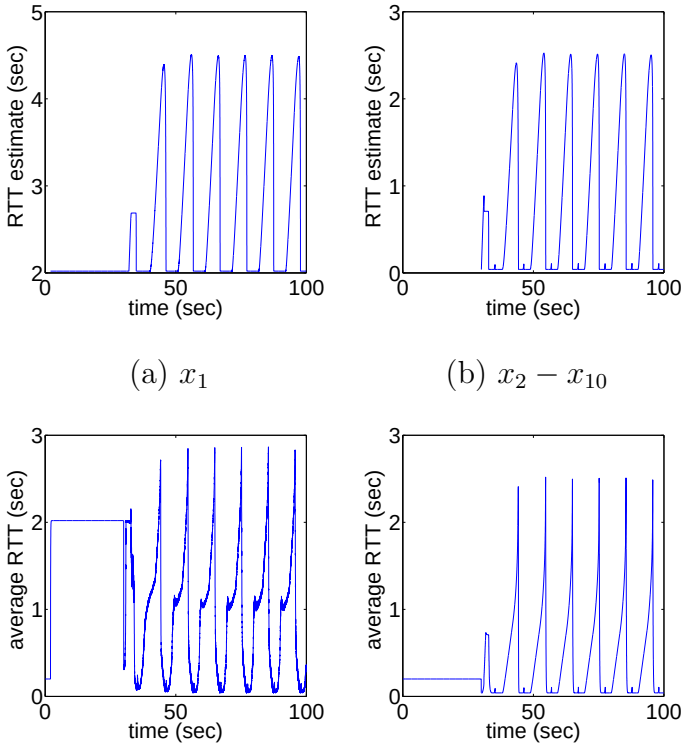
Fig. 5. Sending rate of flows $x_1 - x_{10}$ and bottleneck id of flows $x_2 - x_{10}$ in the case of RCP for topology $\mathcal{T}_u$ with higher link delay confirming instability.



(a) rate at l_1 (b) queue at l_1

(c) rate at l_3 (d) queue at l_3

Fig. 6. Control rate and queue size at links l_1 and l_3 in the case of RCP for topology $\mathcal{T}_u$ with higher link delay confirming instability.



(a) x_1 (b) $x_2 - x_{10}$

(c) l_1 (d) l_3

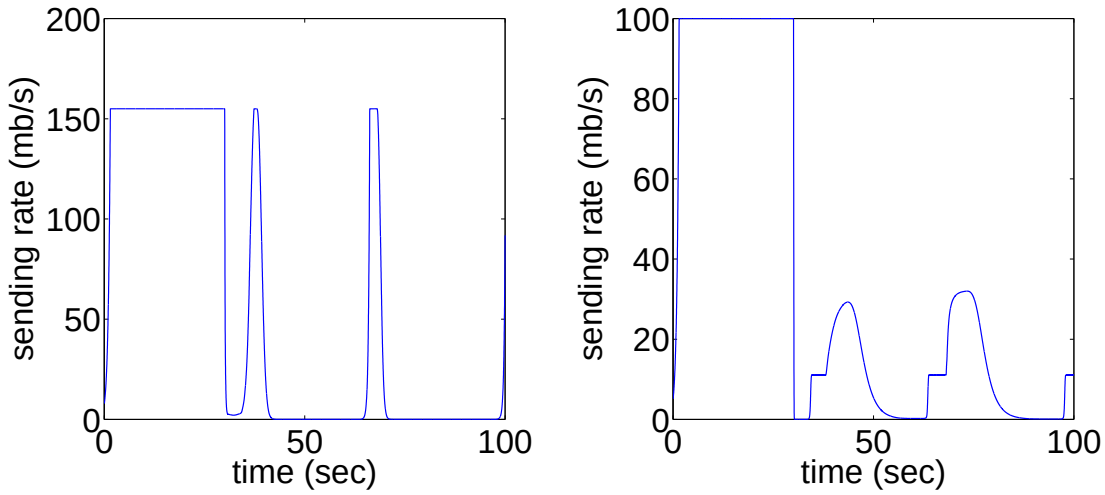
Fig. 7. RTT of flows $x_1 - x_{10}$ and average RTT at links l_1 and l_3 in the case of RCP for topology $\mathcal{T}_u$ with higher link delay.

turns out that because of the oscillating queue size at bottleneck links l_1 and l_3, the RTT of flows $x_1 - x_{10}$ also keeps fluctuating. For example, the RTT of flow x_1 varies between 2.02 s to 4.6 s and the RTT of flows $x_2 - x_{10}$ vary between 0.04 s to 2.62 s. Oscillations in the RTTs of flows $x_1 - x_{10}$ cause the average RTTs at the bottleneck links l_1 and l_3 to oscillate. The average RTT d_l at both l_1 and l_3 vary between 0.04 s to 2.62 s since majority of the input traffic passing through them is from flows $(x_2 - x_{10})$ that have similar RTT. It should be noted that the average RTT d_l at link l_1 is not only oscillatory, but at certain time instants it has a value much smaller than the maximum RTT (which is the RTT of x_1 varying between 2.02 s to 4.6 s) of flows passing through it. We suspect that this may be a reason that is violating the stability conditions at l_1 since average RTT d_l is closely coupled with the control gain parameters α and β.

During the simulation, we also study the bottleneck assignment of flow x_1 and flows $x_2 - x_{10}$. We find that flow x_1 always remains bottlenecked at link l_1 but flows $x_2 - x_{10}$ keep switching their bottleneck link between l_1 (bottleneck id = 0) and l_3 (bottleneck id = 4) as shown in Fig. 5. In order to verify whether bottleneck oscillation is a cause or effect of instability in RCP we next repeat the simulation with fixed bottleneck assignment.

c. RCP with Fixed Bottlenecks

We configure the router controller of link l_1 to assign feedback only to packets from flow x_1 and configure the router controller at link l_3 to always assign feedback to packets from flows $x_2 - x_{10}$. We do the simulation with fixed bottleneck assignment and the sending rate of flows $x_1 - x_{10}$ in this scenario is shown in Fig. 8. The system still behaves in an unstable manner confirming that bottleneck oscillation is the effect of an unstable controller rather than the cause as claimed in [1].



(a) x_1 (b) $x_2 - x_{10}$

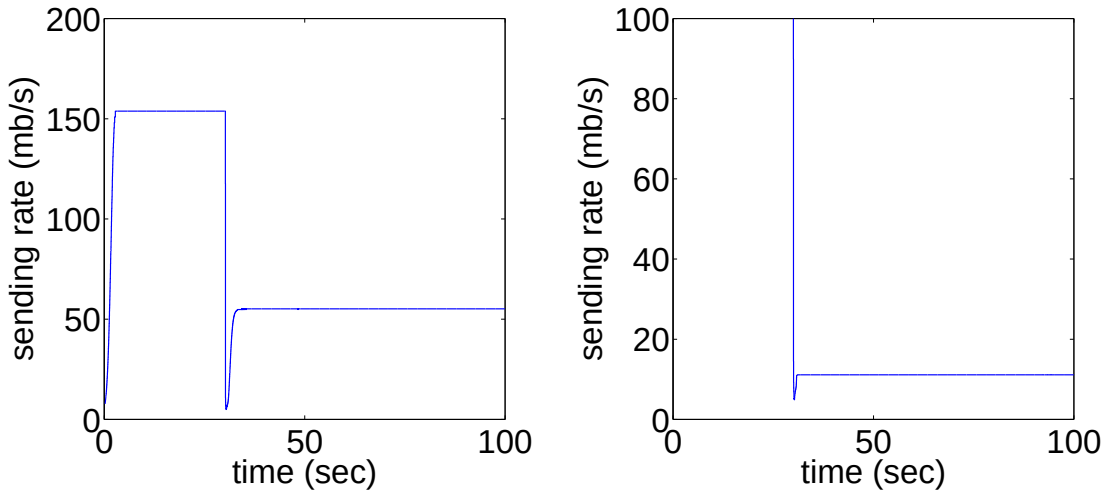
Fig. 8. Sending rate of flows $x_1 - x_{10}$ in the case of RCP with fixed bottleneck assignment indicating instability.

d. RCP-1

We modify equation (3.1) by dropping the queue term $q_l(t)$. The corresponding equation is

$$R_l(t) = R_l(t - T)\left[1 + \frac{T\alpha\left(C_l - y_l(t)\right)}{d_l C_l}\right] \tag{3.2}$$

and has been referred to as RCP-1. This equation has tractable stability conditions (see chapter IV). The behavior of RCP-1 for the simulation setup as considered previously is shown in Fig. 9. At $t = 0$, flow x_1 enters the system and quickly saturates the bottleneck link l_1. At $t = 30$, 9 flows $x_2 - x_{10}$ join the system. The network quickly recovers from the transient state and converges to a stable steady state. Clearly the system is stable with no oscillations in the sending rate of any of the flows. However, a side effect of using control equation (3.2) is a non-zero queue at link l_3. This mainly occurs because at equilibrium (3.2) only provides $y_l(t) = C_l$ and not $q_l(t) = 0$. The question whether stability or zero queue is more important for a



(a) x_1 (b) $x_2 - x_{10}$

Fig. 9. Sending rate of flows $x_1 - x_{10}$ in the case of RCP-1 (3.2) for topology $\mathcal{T}_u$ indicating stability.

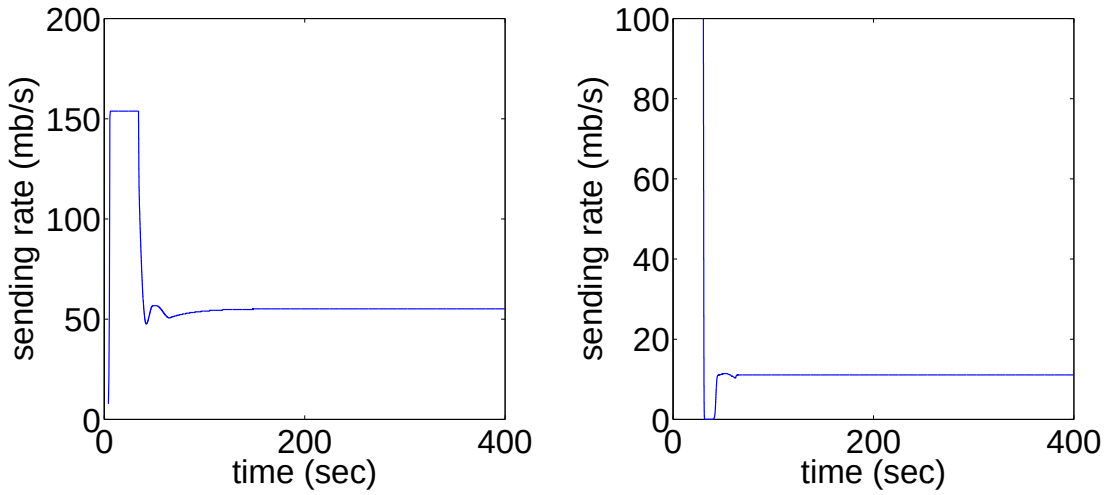
congestion control scheme is beyond the scope of this work. However, in practice it is always desirable to have zero or low queue size to absorb traffic burst, reduce RTT of flows and jitter for voice applications.

e. RCP-2

Consider the following control equation at router l

$$R_l(t) = R_l(t-T)\left[1 + \frac{T\alpha\left(C_l - y_l(t)) - \beta\frac{q_l(t)}{D}\right)}{DC_l}\right] \tag{3.3}$$

where D is the maximum RTT of all flows passing through the router. Equation (3.3) has been referred to as RCP-2. It is identical to the router control equation (3.1) of RCP with average RTT d_l replaced by maximum RTT D. We next study the behavior of RCP-2 in topology $\mathcal{T}_u$ with higher link delays. The simulation result is shown in Fig. 10 indicating stability. During the simulation, we find that unlike the



(a) x_1 (b) $x_2 - x_{10}$

Fig. 10. Sending rate of flows $x_1 - x_{10}$ in the case of RCP-2 (3.3) for topology $\mathcal{T}_u$ with higher link delay indicating stability.

case of RCP, the values of average RTT at links l_1 and l_3 are not very far from the maximum RTT of flows passing through the links. Also, the values of average RTT do not oscillate significantly. Substituting average RTT d_l with maximum RTT D has the effect that whatever change in traffic rate $C_l - y_l(t)$ or drain in queue $q_l(t)$ that router l wants to undergo is done over a larger period of time since $D > d_l$. Hence, when maximum RTT D is used in the control equation as against average RTT d_l, a smaller fraction of the net change is being applied (since $T/d_l > T/D$) during each control interval T. This reduces the responsiveness of the controller but improves its stability. Because of the difficulty in modeling RCP, the exact conditions of stability cannot be found accurately. We speculate here without proof that the conditions for stability in the case of RCP are also a function of d_l and D. We show this for a modified RCP like controller in the next chapter.

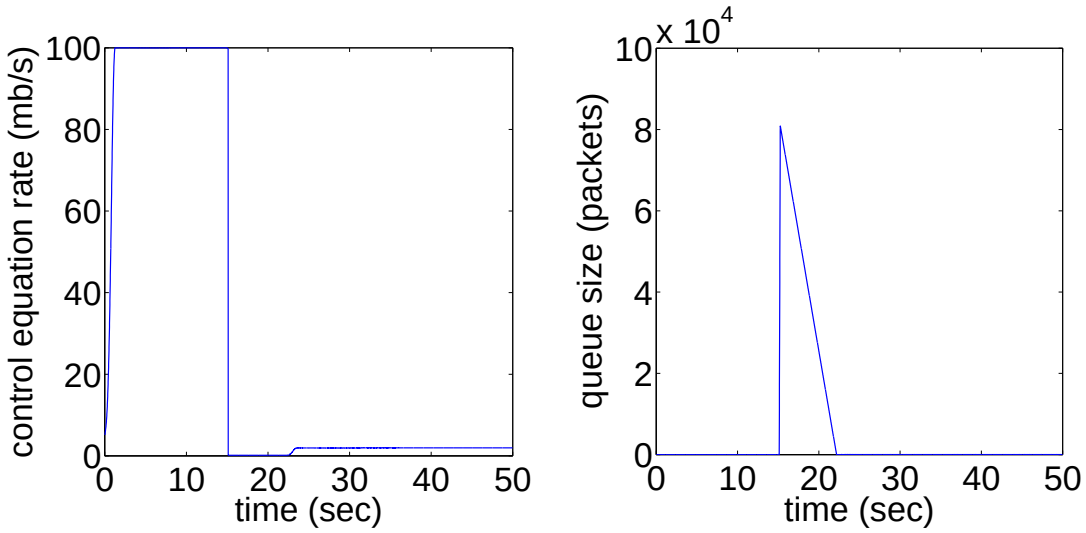
2. High Buffering Requirement

In this section, we show the high buffering requirement in the case of RCP to avoid a large number of packet losses. The end-host control equation in RCP sets the sending rate $x_i(t)$ of flow i equal to the received feedback $R(t - D_i^{\leftarrow})$ from the router as

$$x_i(t) = R(t - D_i^{\leftarrow}). \tag{3.4}$$

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 capacity and an increase in the queue occupancy. The problem becomes extremely severe when a large number of flows join the system simultaneously as indicated by the simulation result shown next.

Consider a dumb-bell topology with bottleneck link capacity 100 mb/s and link delay of 50 ms. The system dynamics for this simulation scenario is shown in Fig. 11. At $t = 0$, flow x_1 enters the system. It saturates the bottleneck link with a sending rate equal to the router control rate of 100 mb/s. At $t = 15$, 50 flows $x_2 - x_{51}$ enter the system simultaneously. New flows entering the system start sending packets at the rate given to them by the router, i.e., 100 mb/s. The input traffic rate at the bottleneck link overshoots the link capacity by 51 times since all the 51 flows are now sending data at the router control rate of 100 mb/s. As shown in Fig. 11, this increases the queue length to a value of 80868 packets and it takes nearly 7.5 s for the system to recover and reach its steady state. Hence, unless a huge buffer is provisioned inside the routers, a large number of packets would be lost. The buffering requirement increases significantly with the increase in the number of flows entering the system simultaneously.



(a) control rate (b) queue size

Fig. 11. Performance of RCP in a dumb-bell topology with bottleneck link capacity 100 mb/s and delay 50 ms with abrupt increase in traffic demand at $t = 15$.

B. How to Fix RCP?

The question that now arises is how to fix the drawbacks of RCP as discussed in the previous sections? How to modify the control equations (at router, at source, or both) to have a tractable stability and lower buffering requirement inside the routers?

- We should remove the queue term from the router control equation in order to avoid modeling difficulties associated with it. However, mechanisms should also be incorporated to drain queue built up due to any transient effects in the system.

- We should modify the control equation at the end-hosts so that new flows joining the system increase their sending rate gradually. This would allow the router control equation to converge to a new steady-state without significantly overflowing the queue.

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 [25]). Second, RCP has a smaller control header size (i.e., 16 bytes) compared to XCP's 20 bytes [10], JetMax's 32 bytes [56], and MKC's 20 bytes [55]. Third, unlike XCP [34], RCP's steady-state rates achieve max-min fairness in general network topologies. Finally, RCP [6] 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 strive to improve upon the drawbacks of RCP in the next chapter.

CHAPTER IV

NEW RATE CONTROL PROTOCOL

In this chapter, we strive to develop a new congestion control framework based on modifications to Rate Control Protocol (RCP) [6] to eradicate its weaknesses as demonstrated in the previous chapter and simultaneously provide a mathematically tractable stability analysis of the system with single-bottleneck links. Closed form stability analysis is missing in the case of RCP as studied in [7].

Consider the feedback control system model of explicit congestion control as shown in Fig. 12. Let $G(s)$ be the *plant* consisting of N users or flows each with sending rate $x_i(t)$ and RTT $D_i = D_i^{\rightarrow} + D_i^{\leftarrow}$. Let $C(s)$ be the router *controller* whose goal is to operate the closed loop system in a stable manner within certain constraints. The output of the plant is the total sending rate $y(t)$ arriving at the router controller and the input being the per user or flow sending rate $R(t)$ generated by the router. Consider the router control equation

$$R(t) = R(t-T)\left[1 + \frac{\frac{T}{d}\left(\alpha(C - y(t)) - \beta\frac{q(t)}{d}\right)}{C}\right], \tag{4.1}$$

where $R(t)$ is the calculated rate, T is the control interval, α and β are constants, C is the link capacity, $y(t)$ is the input traffic rate, $q(t)$ is the queue length, and d is the average RTT of flows passing through the router. The control equation at the router is referred to as *router controller*. Also, consider the control equation for flow i referred to as *source controller* and given as:

$$x_i(t) = R(t - D_i^{\leftarrow}). \tag{4.2}$$

Equation (4.1) and (4.2) together form an RCP system that has been studied in

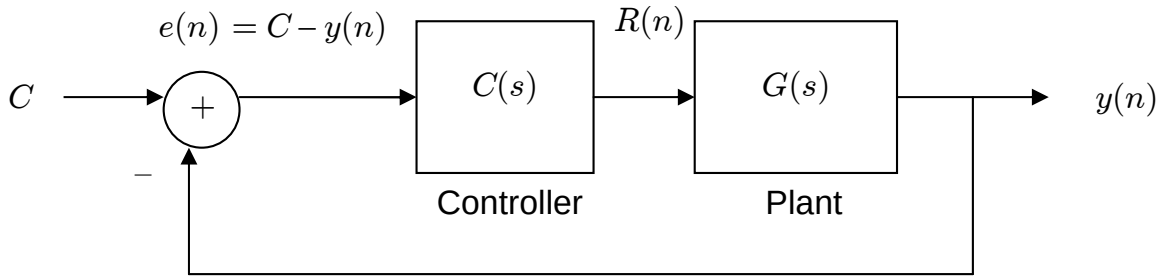


Fig. 12. Feedback control system model of explicit congestion control.

detail in [7].

A. Router Controller

Consider the controller at the router given as:

$$R(t) = R(t-T) + \frac{T\alpha(\gamma C - y(t))}{Nd}, \tag{4.3}$$

where γ is a constant and N is the number of flows in the system. Using $\gamma C/R(t-T)$ as an estimate of N, (4.3) can be written as:

$$R(t) = R(t-T)\left[1 + \frac{T\alpha(\gamma C - y(t))}{\gamma C d}\right]. \tag{4.4}$$

Equation (4.4) is non-linear and can be linearized to obtain equation (4.3). It is identical to the router control equation (4.1) of RCP but without the queue term and link capacity scaled by γ. The absence of the queue term eradicate modeling difficulties associated with it and helps in carrying out a tractable analysis. Also working with virtual link capacity γC helps to drain a non-zero queue for $0 < \gamma < 1$.

Theorem 1. *Control equation given by* (4.3) *represents an Integral controller. The Proportional and Derivative components are absent from the control system.*

Proof. We prove the theorem first using continuous analysis and then using discrete

analysis. The input to the controller is error signal $e(t) = \gamma C - y(t)$ and its output is sending rate $R(t)$. As stated in [40], the transfer function $C(s) = R(s)/e(s)$ in the Laplace domain of a controller that has Proportional, Integral, and Derivative components can be expressed as:

$$C(s) = \frac{R(s)}{e(s)} = K_P + \frac{K_I}{s} + K_D s, \tag{4.5}$$

where K_P, K_I, and K_D are the corresponding gains. Hence, the output $R(s)$ of the controller in the Laplace domain can be written as:

$$R(s) = K_P e(s) + \frac{K_I}{s} e(s) + K_D s e(s). \tag{4.6}$$

Converting equation (4.3) in the form of a differential equation, we get:

$$\dot{R}(t) = \frac{\alpha e(t)}{Nd} \tag{4.7}$$

Taking the Laplace transform of both sides of the above equation, we have:

$$C(s) = \frac{R(s)}{e(s)} = \frac{\alpha}{sNd} \tag{4.8}$$

On comparing the above equation with (4.5), it can be seen that $K_P = K_D = 0$ and $K_I = \alpha/(Nd)$. Hence (4.3) is an Integral controller.

In the discrete-time domain with T as the sampling period, (4.3) can be written as:

$$R(n) = R(n-1) + \frac{\alpha(\gamma C - y(n))}{Nd} \tag{4.9}$$

Rewriting equation (4.6) in the z-domain using the transformation $s = (1 - z^{-1})/T$, we have:

$$R(z) = K_P e(z) + \frac{K_I T e(z)}{1 - z^{-1}} + \frac{K_D}{T}(1 - z^{-1}) e(z). \tag{4.10}$$

Multiplying both sides of the above equation by $(1 - z^{-1})$, we get:

$$(1 - z^{-1})R(z) = (1 - z^{-1})K_P e(z) + K_I T e(z) + \frac{K_D}{T}(1 - z^{-1})^2 e(z). \tag{4.11}$$

Transforming the above equation to the time domain, we have:

$$R(n) - R(n-1) = K_P(e(n) - e(n-1)) + K_I T e(n) + \frac{K_D}{T}(e(n) - 2e(n-1) + e(n-2)). \tag{4.12}$$

After rearranging terms, the output $R(n)$ of the controller is given by:

$$\begin{aligned} R(n) &= R(n-1) + \left(K_P + K_I T + \frac{K_D}{T}\right) e(n) - \left(K_P + 2\frac{K_D}{T}\right) e(n-1) \\ &\quad + \frac{K_D}{T} e(n-2) \\ &= R(n-1) + \left(K_P + K_I T + \frac{K_D}{T}\right) (\gamma C - y(n)) \\ &\quad - \left(K_P + 2\frac{K_D}{T}\right) (\gamma C - y(n-1)) + \frac{K_D}{T}(\gamma C - y(n-2)) \end{aligned} \tag{4.13}$$

On comparison with (4.9), the gain parameters can be evaluated as $K_P = K_D = 0$ and $K_I = \alpha/(NTd)$. Hence, (4.9) is an Integral controller whose goal is to converge error $e(n) = (\gamma C - y(n)) \to 0$ or $y(n) \to \gamma C$ in the steady state. □

Consider the following router controller:

$$R(t) = R(t - T)\left[1 + \frac{\alpha_1 T(\gamma C - y(t))}{\gamma C d} + \frac{\alpha_2 T(\gamma C - y(t - T))}{\gamma C d}\right]. \tag{4.14}$$

The above equation can be linearized to obtain:

$$R(t) = R(t - T) + \frac{\alpha_1 T(\gamma C - y(t))}{N d} + \frac{\alpha_2 T(\gamma C - y(t - T))}{N d}. \tag{4.15}$$

Using the continuous analysis in the previous theorem, the above linearized controller has gain parameters $K_D = 0$, $K_P = -\alpha_2 T/(Nd)$, and $K_I = (\alpha_1 + \alpha_2)/(Nd)$. Hence, equation (4.15) is a Proportional Integral (PI) controller with no Derivative

component. Apart from adding the Proportional component, it also has a higher Integral gain (for $\alpha_1 = \alpha_2 = \alpha$) as compared to (4.3). This helps in improving the system response time and limiting the queue to lower levels. It was also found using simulations that including the Derivative component does not improve the system dynamics further.

B. Source Controller

The system dynamics $G(s)$ of RCP has the affect of overflowing the router queue when new flows join the system. This is undesirable in practice and instead we want system dynamics that does not significantly overshoot the queue. Intuitively, new flows should not directly use the sending rate given to them by the router since it may be incorrect. Because of the usage of a PI controller, we know the system would converge to a stable steady state with zero error provided the controller is stable. However, it is desirable to have a transient phase where the router queues are small. Hence, new flows joining the system should gradually increase their sending rate. Consider the following Exponentially Weighted Moving Average (EWMA) type source controller, where upon receiving feedback $R(n)$ the sending rate $x_i(n)$ of flow i is updated as:

$$x_i(n) = x_i(n-1) - \tau_1 \left(x_i(n-1) - R(n - D_i^{\leftarrow}) \right) \tag{4.16}$$

This source controller is also used by JetMax and has the property that in the steady state, the sending rate of a flow converges to the rate received from the router as feedback. How fast the source rate converges to its steady state depends upon the choice of τ_1.

Theorem 2. *For the source controller* (4.16) *with the router controller in its steady*

state, the number of control steps required to converge to $(1-\epsilon)$*-efficiency is independent of the link capacity and the steady state value.*

Proof. For the router already in its steady-state phase, the subsequent feedback values K are all the same. By expanding the equation (4.16) recursively, the sending rate can be written as:

$$\begin{aligned} x_i(n) &= (1-\tau_1)^n x_i(0) + K\tau_1 \left[1 + (1-\tau_1) + (1-\tau_1)^2 + \cdots + (1-\tau_1)^{n-1}\right] \\ &= (1-\tau_1)^n x_i(0) + K\tau_1 \left[\frac{1-(1-\tau_1)^n}{1-(1-\tau_1)}\right] \\ &= (1-\tau_1)^n x_i(0) + K\left[1-(1-\tau_1)^n\right] \end{aligned} \tag{4.17}$$

After rearranging terms, the above equation can be written as:

$$(K - x_i(n)) = (1-\tau_1)^n (K - x_i(0)). \tag{4.18}$$

Defining $\epsilon = (K - x_i(n))/(K - x_i(0))$, the above equation is equivalent to $\epsilon = (1-\tau_1)^n$. Hence, the number of control steps required to converge to K is $n = \lceil \log_{1-\tau_1} \epsilon \rceil$, which depends only upon τ_1 and ϵ and is independent of the link capacity. □

Consider the following source controller:

$$x_i(n) = x_i(n-1) - \tau_1(x_i(n-1) - R(n - D_i^{\leftarrow})) + \tau_2(R(n - D_i^{\leftarrow}) - R(n-1-D_i^{\leftarrow})), \tag{4.19}$$

where $R(n - D_i^{\leftarrow})$ and $R(n-1-D_i^{\leftarrow})$ are the two most recent feedback received from the router. If $R(n - D_i^{\leftarrow}) > R(n-1-D_i^{\leftarrow})$ then the router is under-utilized and wants to encourage the flows to increase their sending rate. On the contrary, if $R(n - D_i^{\leftarrow}) < R(n-1-D_i^{\leftarrow})$ then the router is over-utilized and wants the flows to decrease their sending rate. The term associated with τ_2 makes the system more responsive. It should be noted here that this gain comes by just saving the last received feedback at the source side without incurring any network overhead.

Another information to note is that τ_2 affects only when the system is in transient state, i.e., when the successive received feedback are different. In case, the router has already reached its steady state, i.e., $y(n) = \gamma C$, successive feedback values would all be the same and the source dynamics is mainly governed by τ_1.

Another question that now arises is how often should the source controller be invoked? Some of the possible options are:

- Every Ack: The source responds to feedback generated during every router control interval and for multiple number of times.
- Every New Feedback: The source responds to feedback generated by the router and only once per the control interval. It also requires a field in the congestion header so that flows can identify the new feedback.
- Once per RTT: The source responds to received feedback only once per RTT.

C. QI-RCP

In this section, we propose a congestion control method called *Queue Independent RCP* (QI-RCP) consisting of router controller (4.4) and source controller (4.2). We next prove its stability in both continuous and discrete case using Nyquist stability criterion.

1. Continuous Case

The linearized QI-RCP system in continuous case can be expressed as:

$$\dot{R}(t) = \frac{\alpha(\gamma C - y(t))}{Nd} = \frac{\alpha}{Nd}\left(\gamma C - \sum_{i=1}^{N} R(t - D_i)\right). \tag{4.20}$$

Taking the Laplace transform of the above system, we have:

$$sR(s) = -\frac{\alpha}{Nd}\sum_{i=1}^{N} R(s)e^{-sD_i} + K, \tag{4.21}$$

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

$$R(s) = \frac{K/s}{1+G(s)}, \tag{4.22}$$

where

$$G(s) = \frac{\alpha}{Nd}\sum_{i=1}^{N} \frac{e^{-sD_i}}{s}. \tag{4.23}$$

In the frequency domain, the above transfer function can be written as:

$$G(j\omega) = \frac{\alpha}{Nd}\sum_{i=1}^{N} \frac{e^{-j\omega D_i}}{j\omega} \tag{4.24}$$

Theorem 3. *For a QI-RCP system consisting of flows with homogeneous RTTs D, the necessary and sufficient condition for local stability is $0 < \alpha < \pi/2$.*

Proof. For $D_i = D = d$, we have:

$$G(j\omega) = \alpha\frac{e^{-j\omega D}}{j\omega D}, \tag{4.25}$$

We start with $\alpha > 0$. The values of ω_i where $G(j\omega)$ crosses the real axis can be found by solving the following equation:

$$\mathrm{Im}[G(j\omega)] = \mathrm{Im}\left[\alpha\frac{\cos(\omega D) - j\sin(\omega D)}{j\omega D}\right] = 0. \tag{4.26}$$

This immediately reduces to $\cos(\omega D) = 0$, which has roots $\omega_i = \pi(1+2i)/(2D)$, where i is an integer. For $i = 0$, we have $\omega_0 = \pi/(2D)$ and the real part of $G(j\omega_0)$ is:

$$\mathrm{Re}\left[\alpha\frac{\cos(j\omega_0 D) - j\sin(\omega_0 D)}{j\omega_0 D}\right] = -\alpha\frac{2}{\pi}. \tag{4.27}$$

To ensure stability, we must satisfy $\mathrm{Re}[G(j\omega_0)] > -1$, which leads to the con-

dition $\alpha < \pi/2$. The remaining steps are to show that for $i \neq 0$ (both positive and negative values), the condition on α becomes looser and diverges to $+/-\infty$. Taking the intersection of all conditions, we find the narrowest that guarantees stability, which is $0 < \alpha < \pi/2$.

Next, for $\alpha = 0$, we have a marginally stable system $\dot{R}(t) = 0$ with a single pole at $s = 0$. For $\alpha < 0$, we have to reverse the condition on $\mathrm{Re}[G(j\omega)]$, which leads to $\alpha > \pi/2$, which contradicts the assumption that $\alpha < 0$. Therefore, no value of $\alpha < 0$ can keep the system stable. □

Theorem 4. *For a QI-RCP system consisting of flows with heterogeneous RTTs, a sufficient condition for local stability is* $0 < \alpha < \pi d/(2D)$*, where* $D = \max\{D_1, D_2, \ldots, D_N\}$ *and* d *is the average RTT of the system.*

Proof. We are interested in roots ω_i' of the following equation:

$$\mathrm{Im}[G(j\omega)] = \frac{\alpha}{Nd} \sum_{i=1}^{N} \cos(\omega D_i) = 0. \tag{4.28}$$

Observe that (4.28) cannot have roots in $[-\omega_0, \omega_0]$, where $\omega_0 = \pi/(2D)$, *unless all delays are equal*. We prove this by contradiction. Assume that $0 \leq \omega_0' < \omega_0$ is the smallest root of (4.28). Then, $0 \leq \omega_0' D_i < \pi/2$, which means that *all* cosine terms in the summation are strictly positive, which contradicts the assumption that ω_0' is a root of (4.28). Since cosine is a symmetric function, we immediately obtain the same contradiction for $-\omega_0 \leq \omega_0' < 0$. Therefore, it follows that $|\omega_0'| \geq \omega_0$.

Next, consider the value of $\mathrm{Re}[G(j\omega_0')]$ where ω_0' as before is the smallest root of (4.28):

$$\mathrm{Re}[G(j\omega_0')] = -\frac{\alpha}{Nd\omega_0'} \sum_{i=1}^{N} \sin(\omega_0' D_i). \tag{4.29}$$

Bounding all sines with 1 and leveraging our prior observation on the relationship

to the homogeneous case, we have assuming $\omega_0' > 0$:

$$\mathrm{Re}[G(j\omega_0')] \geq -\frac{\alpha}{\omega_0' d} \geq -\frac{\alpha}{\omega_0 d} = -\frac{2D\alpha}{\pi d}. \tag{4.30}$$

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. If $\omega_0' < 0$, observe that $\sin(\omega_0')/\omega_0' = \sin(-\omega_0')/(-\omega_0')$, which can be converted to the case of positive ω_0' to produce identical results to those in (4.30). Finally, noticing that the remaining ω_i' are larger than ω_0', it follows that they can only shift (4.30) further toward zero and thus lead to looser bounds on α. Hence, a sufficient condition for stability is $0 < \alpha < \pi d/(2D)$ □

2. Discrete Case

In this section, we analyze the stability of QI-RCP in the discrete case. We first consider the ideal undelayed scenario and then a more practical setting with delays in the network.

Theorem 5. *For a QI-RCP system in an undelayed scenario, the necessary and sufficient condition for local stability is* $0 < \alpha < 2$

Proof. For a QI-RCP system in an undelayed scenario, the source controller is:

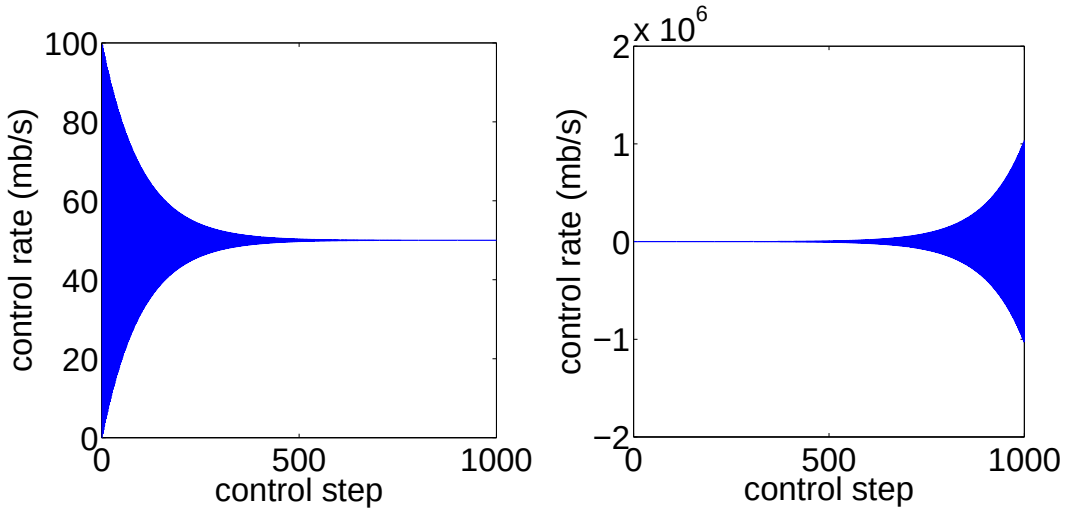
$$x_i(n) = R(n-1)$$

and the linearized router controller is:

$$R(n) = R(n-1) + \frac{\alpha(\gamma C - y(n))}{N}.$$

At equilibrium,

$$R(n) = x_i(n) = R^* = \frac{\gamma C}{N}. \tag{4.31}$$



(a) $\alpha = 1.99$ (b) $\alpha = 2.01$

Fig. 13. Verification of undelayed stability conditions for QI-RCP.

The input traffic rate $y(n)$ seen at the router is:

$$y(n) = \sum_{i=1}^{N} x_i(n) = \sum_{i=1}^{N} R(n-1) = NR(n-1). \tag{4.32}$$

Substituting the value of $y(n)$ in the router control equation, we have:

$$\begin{aligned} R(n) &= R(n-1) + \frac{\alpha\gamma C}{N} - \frac{\alpha}{N} NR(n-1) \\ &= R(n-1)(1-\alpha) + \frac{\alpha\gamma C}{N}. \end{aligned} \tag{4.33}$$

The above system dynamics has only 1 eigenvalue $\lambda = (1-\alpha)$. For a stable system, $|\lambda| < 1$ should be satisfied, which leads to $0 < \alpha < 2$. □

We verify the stability conditions as derived above using MATLAB. They are shown in Fig. 13. Clearly the system is stable for $\alpha = 1.99$ but becomes unstable for $\alpha = 2.01$.

Theorem 6. *For a QI-RCP system consisting of flows with RTTs D_1, D_2, ..., D_N, the necessary and sufficient condition for local stability is that the roots of the following*

characteristic equation expressed in the z-domain must lie within the unit circle.

$$1 - z^{-T} + \frac{T\alpha}{Nd} \sum_{i=1}^{N} z^{-D_i} = 0 \tag{4.34}$$

Proof. For a QI-RCP system in the discrete-time domain, the source controller is:

$$x_i(n) = R(n - D_i^{\leftarrow}) \tag{4.35}$$

and the linearized router controller is:

$$R(n) = R(n - T) + \frac{\alpha T(\gamma C - y(n))}{Nd}, \tag{4.36}$$

At equilibrium,

$$R(n) = x_i(n) = R^* = \frac{\gamma C}{N}. \tag{4.37}$$

The input traffic rate $y(n)$ seen at the router is:

$$y(n) = \sum_{i=1}^{N} x_i(n - D_i^{\rightarrow}) = \sum_{i=1}^{N} R(n - D_i). \tag{4.38}$$

Substituting the value of $y(n)$ in the router control equation, we have:

$$R(n) = R(n - T) + \frac{T\alpha\gamma C}{Nd} - \frac{T\alpha}{Nd} \sum_{i=1}^{N} R(n - D_i). \tag{4.39}$$

Near the equilibrium point R^* and using $X(n) = R(n) - R^* = R(n) - \gamma C/N$, the above equation can be written as:

$$X(n) = X(n - T) - \frac{T\alpha}{Nd} \sum_{i=1}^{N} X(n - D_i). \tag{4.40}$$

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

$$X(z) = \frac{K}{1 - z^{-T} + \frac{T\alpha}{Nd} \sum_{i=1}^{N} z^{-D_i}}, \tag{4.41}$$

where K is a constant. For stability of the system, the poles of the transfer func-

tion must lie within the unit circle. Hence, the location of roots of the following characteristic equation must be within the unit circle.

$$1 - z^{-T} + \frac{T\alpha}{Nd} \sum_{i=1}^{N} z^{-D_i} = 0 \tag{4.42}$$

□

While implementing a QI-RCP system, the control algorithm is invoked every T intervals. The calculated rate is kept constant for the period T after which a new rate is calculated again. Hence, the only discrete case we need to analyze is $T = 1$ and $D_i \leftarrow \lceil D_i/T \rceil$. Using this information, the QI-RCP system transfer function (4.41) becomes:

$$X(z) = \frac{K/(1 - z^{-1})}{1 + G(z)}, \tag{4.43}$$

where,

$$G(z) = \frac{\alpha}{Nd} \sum_{i=1}^{N} \frac{z^{-D_i}}{1 - z^{-1}}. \tag{4.44}$$

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

$$G(e^{j\omega}) = \frac{\alpha}{Nd} \sum_{i=1}^{N} \frac{e^{-j\omega(D_i-1)}}{e^{j\omega} - 1}. \tag{4.45}$$

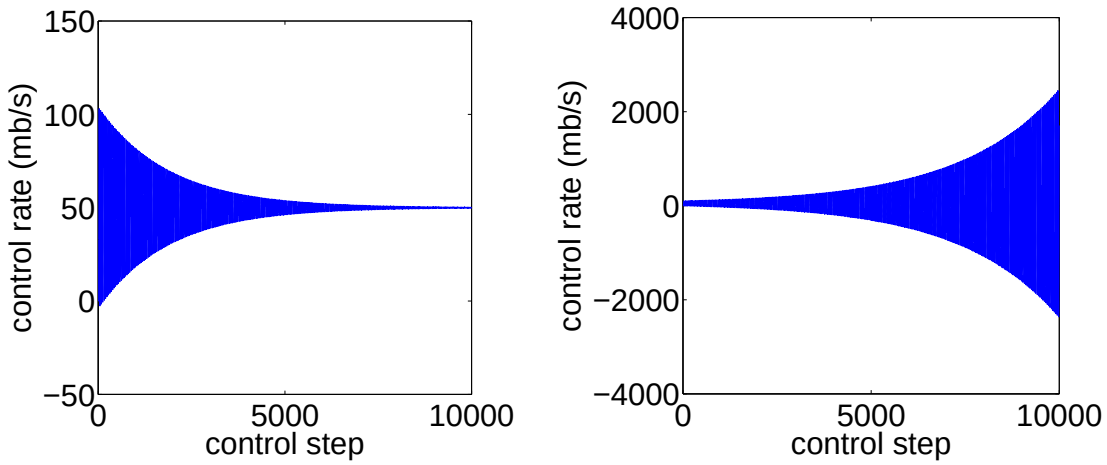
After expanding the exponentials, (4.45) can also be written as:

$$G(e^{j\omega}) = -\frac{\alpha}{2Nd \sin(\omega/2)} \sum_{i=1}^{N} \left[\sin \frac{\omega(2D_i - 1)}{2} + j \cos \frac{\omega(2D_i - 1)}{2} \right] \tag{4.46}$$

Theorem 7. *For a QI-RCP system consisting of flows with homogeneous RTT D, the necessary and sufficient condition for local stability is:*

$$0 < \frac{\alpha}{D} < 2 \sin \left(\frac{\pi}{2(2D - 1)} \right). \tag{4.47}$$

Proof. When RTTs of all the flows are equal to D (hence $d = D$), $G(e^{j\omega})$ crosses the



(a) $\alpha = 1.64$ (b) $\alpha = 1.66$

Fig. 14. Verification of stability condition for QI-RCP in the case of flows with homogeneous RTT $D = 10$. The necessary and sufficient condition for stability is $\alpha < 1.6523$.

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:

$$\mathrm{Re}[G(e^{j\omega_0})] = \frac{-\alpha}{2D\sin(\omega_0/2)} = \frac{-\alpha}{2D\sin\left(\frac{\pi}{2(2D-1)}\right)}. \tag{4.48}$$

Using arguments in the proof of theorem 3, stability is ensured if and only if:

$$0 < \frac{\alpha}{D} < 2\sin\left(\frac{\pi}{2(2D-1)}\right). \tag{4.49}$$

□

We verify the stability condition as derived above using MATLAB. Consider the case when $D = 10$. For stability, the necessary and sufficient condition is $\alpha < 1.6523$. It can be seen from Fig. 14 that for $\alpha = 1.64$, the system is stable but becomes unstable for $\alpha = 1.66$.

Theorem 8. *For a QI-RCP system consisting of flows with heterogeneous RTTs, a*

sufficient condition for local stability is:

$$0 < \frac{\alpha}{d} < 2\sin\left(\frac{\pi}{2(2D-1)}\right), \tag{4.50}$$

where $D = \max\{D_1, D_2, \ldots, D_N\}$ *and* d *is the average RTT of the system.*

Proof. Using (4.46), we find that $G(e^{j\omega})$ crosses the real axis (i.e., $\mathrm{Im}[G(e^{j\omega})] = 0$) when:

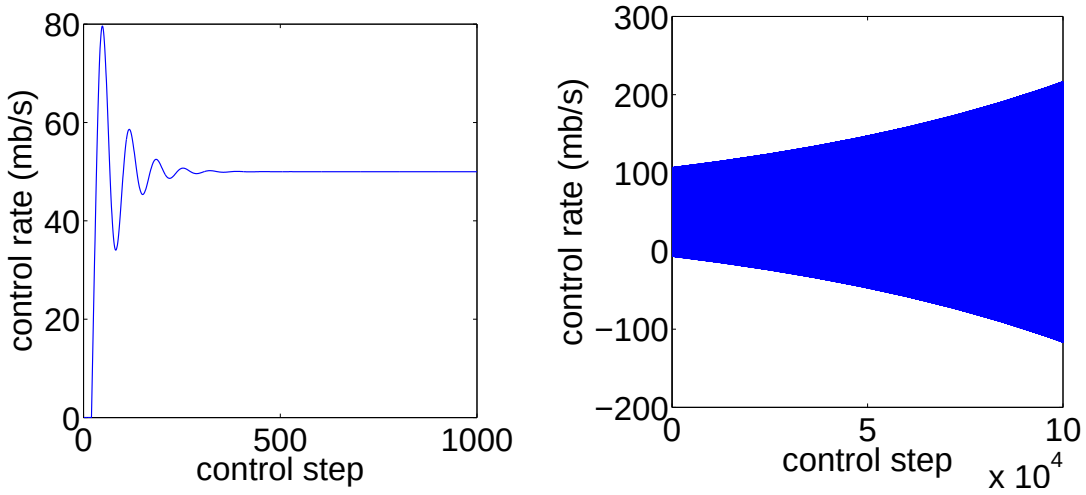
$$\sum_{i=1}^{N} \cos\frac{\omega(2D_i - 1)}{2} = 0. \tag{4.51}$$

It can be seen that none of the roots ω_i' of the above equation have absolute value smaller than $\omega_0 = \pi/(2D-1)$. Due to the periodic nature (with period 2π) of the frequency domain of a discrete-time system, we can limit our attention to $\omega_i' \in [-\pi, \pi]$. Also, π is a solution to (4.51) since $(2D_i' - 1)$ is odd for any integer D_i. Based on these arguments, the smallest root ω_0' of (4.51) 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:

$$\begin{aligned}
\mathrm{Re}[G(e^{j\omega_0'})] &= -\frac{\alpha}{2Nd\sin(\omega_0'/2)} \sum_{i=1}^{N} \sin\frac{\omega_0'(2D_i - 1)}{2} \\
&\geq -\frac{\alpha}{2d\sin(\omega_0'/2)} \\
&\geq -\frac{\alpha}{2d\sin(\omega_0/2)} \\
&= -\frac{\alpha}{2d\sin\left(\frac{\pi}{2(2D-1)}\right)}
\end{aligned} \tag{4.52}$$

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 $\omega_0' < 0$. Based on arguments as in the proof of theorem 4, a sufficient condition for stability is:

$$0 < \frac{\alpha}{d} < 2\sin\left(\frac{\pi}{2(2D-1)}\right) \tag{4.53}$$



(a) $\alpha = 1.2$ (b) $\alpha = 1.896$

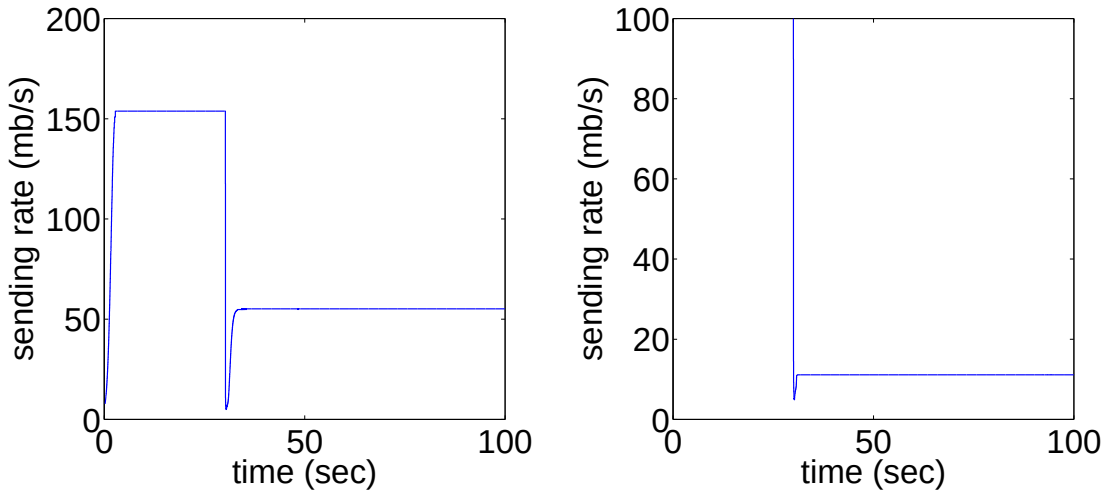
Fig. 15. Verification of stability condition for QI-RCP in the case of flows with heterogeneous RTTs $D_1 = 10$, $D_2 = 20$. The sufficient condition for stability is $\alpha < 1.2080$.

□

We verify the stability condition as derived above using MATLAB. Consider the case of two flows with $D_1 = 10$ and $D_2 = 20$. A sufficient condition for stability is $\alpha < 1.2080$. It can be seen from Fig. 15 that for $\alpha = 1.2$, the system is stable but becomes unstable only for $\alpha \geq 1.896$ indicating that the condition is not necessary.

We verify that QI-RCP is stable in the topology shown in Fig. 3 where RCP was unstable as shown in chapter III. The corresponding plots for sending rate of flow x_1 and $x_2 - x_{10}$ are shown in Fig. 16.

For $T/D \approx 0$, the conditions derived in continuous and discrete case become equivalent. Though QI-RCP has mathematically tractable stability conditions, it still uses the aggressive source controller (4.2) used in the case of RCP. This has the effect of input traffic rate significantly overshooting the link capacity and hence overflowing the queue size when a flash crowd of long flows join the system simultaneously. We



(a) x_1 (b) $x_2 - x_{10}$

Fig. 16. Sending rate of flows $x_1 - x_{10}$ in the topology shown in Fig. 3 for QI-RCP indicating stability.

next strive to improve this.

D. PIQI-RCP

In this section, we propose *Proportional Integral Queue Independent RCP* (PIQI-RCP) consisting of router controller (4.14) and source controller (4.19). We next prove its stability in both continuous and discrete case.

1. Continuous Case

In this section, we analyze the stability of PIQI-RCP in the continuous case. Consider the control system as shown in Fig. 12. The individual blocks, i.e., the controller and the plant in the context of PIQI-RCP are analyzed below.

The linearized router controller can be written as:

$$R(t) = R(t-T) + \frac{T\alpha_1(\gamma C - y(t))}{Nd} + \frac{T\alpha_2(\gamma C - y(t-T))}{Nd}. \tag{4.54}$$

For simplicity, we assume $\alpha_1 = \alpha_2 = \alpha$. We define the error signal $e(t) = \gamma C - y(t)$ and hence $e(t-T) = \gamma C - y(t-T)$. After substituting the error term and converting the above equation in the form of differential equation, we have:

$$\dot{R}(t) = \frac{\alpha}{Nd}\left[e(t) + e(t-T)\right]. \tag{4.55}$$

Taking the Laplace transform of both sides of the above equation, the transfer function $C(s) = R(s)/e(s)$ can be written as:

$$C(s) = \frac{\alpha}{sNd}\left[1 + e^{-sT}\right]. \tag{4.56}$$

For small values of sT, we can approximate $e^{-sT} = 1 - sT$. With this approximation, the transfer function of the controller can be written as

$$C(s) = \frac{\alpha}{sNd}(2 - sT). \tag{4.57}$$

The plant consists of N flows and each flow i adjusts its sending rate $x_i(t)$ as:

$$\begin{aligned} x_i(t) &= x_i(t-T) - \tau_1(x_i(t-T) - R(t - D_i^{\leftarrow})) + \tau_2(R(t - D_i^{\leftarrow}) - R(t - T - D_i^{\leftarrow}) \\ &= (1-\tau_1)x_i(t-T) + (\tau_1 + \tau_2)R(t - D_i^{\leftarrow}) - \tau_2 R(t - T - D_i^{\leftarrow}) \end{aligned} \tag{4.58}$$

The total input traffic rate observed at the router is given by:

$$\begin{aligned} y(t) &= \sum_{i=1}^{N} x_i(t - D_i^{\rightarrow}) \\ &= \sum_{i=1}^{N} \left[x_i(t - T - D_i^{\rightarrow})(1-\tau_1) + (\tau_1 + \tau_2)R(t - D_i) - \tau_2 R(t - T - D_i)\right] \\ &= (1-\tau_1)y(t-T) + (\tau_1 + \tau_2)\sum_{i=1}^{N} R(t - D_i) - \tau_2 \sum_{i=1}^{N} R(t - T - D_i) \end{aligned} \tag{4.59}$$

Converting the above equation in the form of a differential equation, we have:

$$\dot{y}(t) = \frac{-\tau_1}{T} y(t-T) + \frac{\tau_1+\tau_2}{T} \sum_{i=1}^{N} R(t-D_i) - \frac{\tau_2}{T} \sum_{i=1}^{N} R(t-T-D_i). \tag{4.60}$$

Taking Laplace transform of both sides of the above equation, we get:

$$sY(s) = \frac{-\tau_1}{T} e^{-sT} Y(s) + \frac{\tau_1+\tau_2}{T} R(s) \sum_{i=1}^{N} e^{-sD_i} - \frac{\tau_2}{T} R(s) \sum_{i=1}^{N} e^{-s(T+D_i)}. \tag{4.61}$$

Hence, the transfer function of the plant $G(s) = Y(s)/R(s)$ can be written as:

$$G(s) = \frac{Y(s)}{R(s)} = \frac{1}{T} \left[\frac{(\tau_1+\tau_2) - \tau_2 e^{-sT}}{s + \frac{\tau_1}{T} e^{-sT}} \right] \sum_{i=1}^{N} e^{-sD_i}. \tag{4.62}$$

Using the approximation $e^{-sT} = 1 - sT$ for small values of sT, $G(s)$ can be written as:

$$G(s) = \frac{1}{T} \left[\frac{\tau_1 + \tau_2 sT}{s + \frac{\tau_1}{T}(1-sT)} \right] \sum_{i=1}^{N} e^{-sD_i}. \tag{4.63}$$

The overall open loop transfer function combining the controller and the plant can be obtained from equation (4.57) and (4.63) as given below:

$$\begin{aligned} T_f(s) &= C(s)G(s) = \frac{\alpha}{sNd}(2-sT)\frac{1}{T}\left[\frac{\tau_1+\tau_2 sT}{s+\frac{\tau_1}{T}(1-sT)} \right] \sum_{i=1}^{N} e^{-sD_i} \\ &= \frac{\alpha}{sNdT} \left[\frac{(2-sT)(\tau_1+\tau_2 sT)}{s+\frac{\tau_1}{T}(1-sT)} \right] \sum_{i=1}^{N} e^{-sD_i} \\ &= \frac{\sum_{i=1}^{N} e^{-sD_i}}{sNd} \left[\frac{\alpha(-T^2\tau_2 s^2 + s(2\tau_2 T - T\tau_1) + 2\tau_1)}{sT(1-\tau_1)+\tau_1} \right] \end{aligned} \tag{4.64}$$

The transfer function $T_{cl}(s)$ of the closed loop system is given as:

$$T_{cl}(s) = \frac{T_f(s)}{1+T_f(s)}, \tag{4.65}$$

where the characteristic equation is $1 + T_f(s) = 0$ or $T_f(s) = -1$. We next study the stability of PIQI-RCP using the model developed above.

Theorem 9. *For a PIQI-RCP system with $T/D \approx 0$, the necessary and sufficient*

condition for local stability in the case of flows with homogeneous RTTs D is $0 < \alpha < \pi/4$.

Proof. Using (4.64), the open loop transfer function $T_f(s)$ of PIQI-RCP in the case of flows with homogeneous RTT D (hence average RTT $d = D$) can be written as:

$$T_f(s) = \frac{e^{-sD}}{sD}\left[\frac{\alpha(-T^2\tau_2 s^2 + s(2\tau_2 T - T\tau_1) + 2\tau_1)}{sT(1-\tau_1)+\tau_1}\right] = T_D(s)T(s), \tag{4.66}$$

where

$$T_D(s) = \frac{e^{-sD}}{sD} \tag{4.67}$$

$$T(s) = \left[\frac{\alpha(-T^2\tau_2 s^2 + s(2\tau_2 T - T\tau_1) + 2\tau_1)}{sT(1-\tau_1)+\tau_1}\right] \tag{4.68}$$

Rewriting the open loop transfer function and the equations given above in the frequency domain using $s = j\omega$, we get:

$$T_f(j\omega) = \frac{e^{-j\omega D}}{j\omega D}\left[\frac{\alpha(T^2\tau_2\omega^2 + j\omega(2\tau_2 T - T\tau_1) + 2\tau_1)}{j\omega T(1-\tau_1)+\tau_1}\right] \tag{4.69}$$

$$T_D(j\omega) = \frac{e^{-j\omega D}}{j\omega D} \tag{4.70}$$

$$T(j\omega) = \left[\frac{\alpha(T^2\tau_2\omega^2 + j\omega(2\tau_2 T - T\tau_1) + 2\tau_1)}{j\omega T(1-\tau_1)+\tau_1}\right] \tag{4.71}$$

We next study the points where $T_D(j\omega)$ and $T(j\omega)$ cross the real axis, i.e., their imaginary parts are equal to zero. After expanding the exponential term in $T_D(j\omega)$, we get:

$$T_D(j\omega) = \frac{e^{-j\omega D}}{j\omega D} = \frac{-1}{\omega D}(\sin\omega D + j\cos\omega D). \tag{4.72}$$

From the above equation, we observe that $T_D(j\omega)$ crosses the real axis (i.e., $\mathrm{Im}[T_D(j\omega)] = 0$) for $\omega_i = (i + 1/2)\pi/D$. While for $i = 0$, $\mathrm{Re}[T_D(j\omega)] = -2/\pi$ and for $i \neq 0$, $\mathrm{Re}[T_D(j\omega)]$ converges to ± 0 with increase in i. The function $T(j\omega)$ crosses the

real axis for $\omega_1' = 0$ at $\mathrm{Re}[T(j\omega)] = 2\alpha\tau_1/\tau_1 = 2\alpha$ and for ω_2' given by:

$$\omega_2' = \left[\frac{\tau_1^2 - 2\tau_1 + 2\tau_1\tau_2}{T^2\tau_2(1-\tau_1)}\right]^{\frac{1}{2}}. \tag{4.73}$$

However, for $0 < \tau_1 < 1$ the value of ω_2' is imaginary for $0 < \tau_1 + 2\tau_2 < 2$. Hence, selecting small values of τ_1 and τ_2 (such as $\tau_1 = 0.01$ and $\tau_2 = 0.1$) that we want in order to restrict overflowing the queue significantly, we can enforce that $T(j\omega)$ crosses the real axis only for $\omega_1' = 0$.

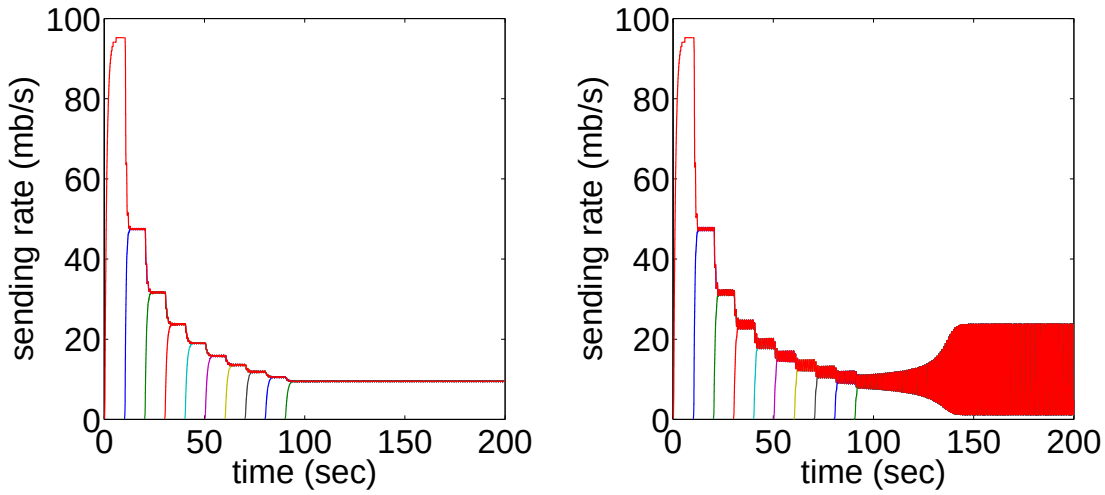
For small ωT (i.e.,$T/D \approx 0$), $T_f(j\omega)$ can be reduced to:

$$T_f(j\omega) = \frac{e^{-j\omega D}}{j\omega D}\frac{2\alpha\tau_1}{\tau_1} = 2\alpha\frac{e^{-j\omega D}}{j\omega D}, \tag{4.74}$$

which using Nyquist stability criterion ensures stability if and only if $0 < \alpha < \pi/4$. □

We next verify the above stability condition using ns-2 simulation. Consider a dumb-bell topology with bottleneck link capacity 100 mb/s and delay 50 ms. A new flow enters the system every 10 seconds and remains in the system for the entire duration of the simulation. The access links of the flows have capacity of 1 gb/s. The the access links of all the flows have identical delay of 10 ms. Hence, all the flows have identical RTT equal to 120 ms. As indicated in the previous theorem, a necessary and sufficient condition for stability in the case of flows with homogeneous RTTs is $\alpha < \pi/4 \approx 0.78571$. We carry out the simulations for $\alpha = 0.77$ and $\alpha = 0.80$. The corresponding plots are shown in Fig. 17(a) and 17(b) respectively. The system clearly becomes unstable for $\alpha = 0.80$ but is stable for $\alpha = 0.77$.

Theorem 10. *For a PIQI-RCP system with $T/D \approx 0$, a sufficient condition for local stability in the case of flows with heterogeneous RTTs is $0 < \alpha < \pi d/(4D)$, where $D = \max\{D_1, D_2, \ldots, D_N\}$ and d is the average RTT of the system.*



(a) $\alpha = 0.77$ (b) $\alpha = 0.80$

Fig. 17. Verification of delayed stability condition for PIQI-RCP in the case of flows with homogeneous RTT $D = 120$ ms. The necessary and sufficient condition for stability is $\alpha < 0.78571$.

Proof. For $T/D \approx 0$, the open loop transfer function (4.64) can be approximated as:

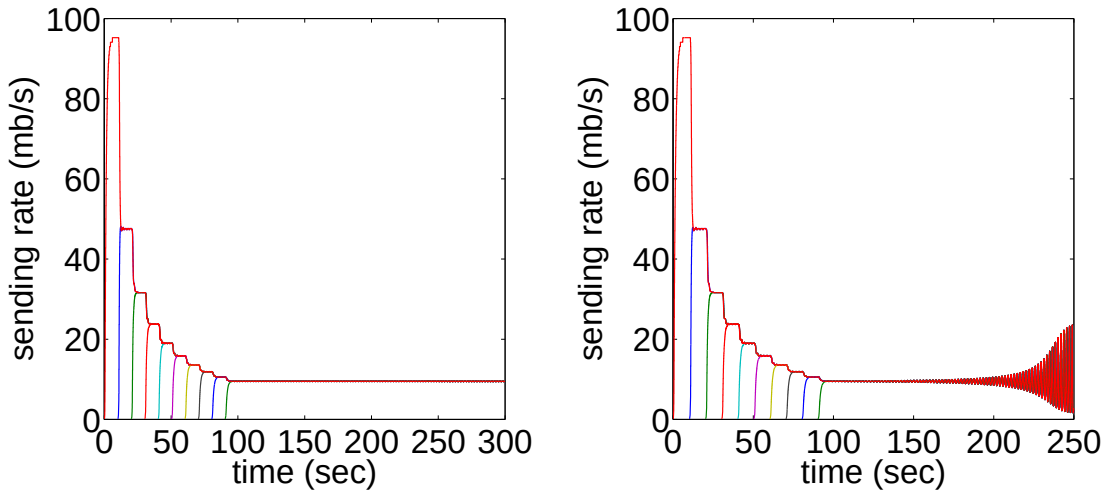
$$T_f(s) = \frac{\sum_{i=1}^{N} e^{-sD_i}}{sNd} \frac{2\alpha\tau_1}{\tau_1} = 2\alpha \frac{\sum_{i=1}^{N} e^{-sD_i}}{sNd}. \tag{4.75}$$

In the frequency domain, the above equation can be written as:

$$T_f(j\omega) = \frac{2\alpha}{Nd} \sum_{i=1}^{N} \frac{e^{-j\omega D_i}}{j\omega}. \tag{4.76}$$

Using the analysis in the proof of theorem 4, a sufficient condition for stability is $0 < \alpha < \pi d/(4D)$. □

It should be noted here that the router controller has knowledge of both average RTT d and maximum RTT D through the packet congestion header. Hence, the stability condition can easily be satisfied. Also within the router control equation, we can replace average RTT d with maximum RTT D to ensure stability condition to be independent of RTTs of flows in the system. However this may reduce the



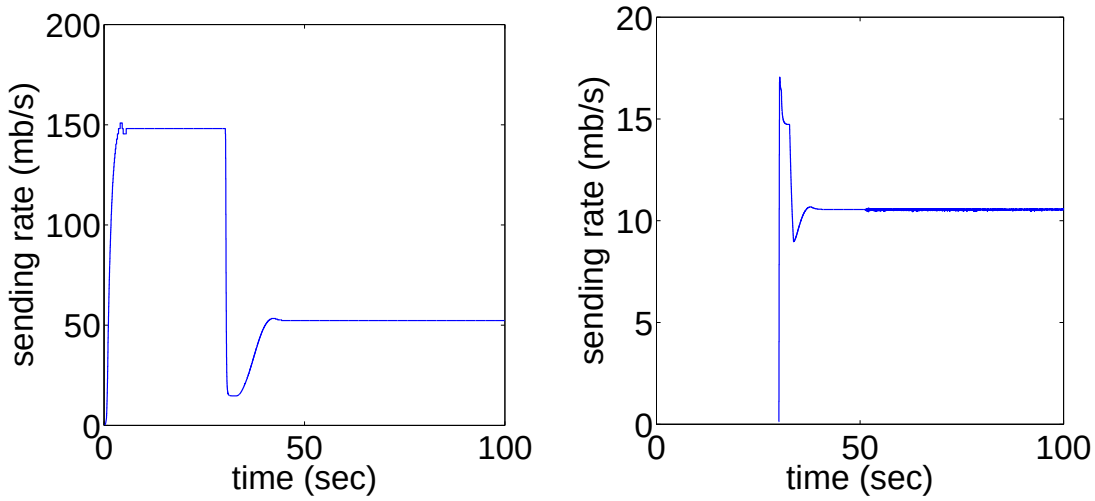
(a) $\alpha = 0.26$ (b) $\alpha = 0.27$

Fig. 18. Verification of delayed stability condition for PIQI-RCP in the case of flows with heterogeneous RTTs. A sufficient condition for stability is $\alpha < 0.261905$.

responsiveness of the controller.

We next verify the stability condition derived above using ns-2 simulation. Consider a dumb-bell topology with bottleneck link capacity 100 mb/s and delay 50 ms. A new flow enters the system every 10 seconds and remains in the system for the entire duration of the simulation. The access links of all flows have different delays. Consider the average RTT d to be 100 ms and the maximum RTT D among all the flows to be 300 ms. As indicated in the previous theorem, a sufficient condition for stability in the case of flows with heterogeneous RTTs is $\alpha < \pi d/4D$. For the current scenario, this leads to $\alpha < \pi/12 \approx 0.261905$. We carry out the simulations for $\alpha = 0.26$ and $\alpha = 0.27$. The corresponding plots are shown in Fig. 18(a) and 18(b) respectively. The system clearly becomes unstable for $\alpha = 0.27$ but is stable for $\alpha = 0.26$.

We verify that PIQI-RCP is stable in the topology shown in Fig. 3, where RCP is unstable as shown in chapter III. The corresponding plot for sending rate of flows



(a) x_1 (b) $x_2 - x_{10}$

Fig. 19. Sending rate of flows $x_1 - x_{10}$ in the topology shown in Fig. 3 for PIQI-RCP indicating stability.

$x_1 - x_{10}$ is shown in Fig. 19.

2. Discrete Case

In this section, we analyze the stability of PIQI-RCP in the discrete case. We first consider the ideal undelayed scenario and then a more practical setting with delays in the network.

Theorem 11. *For a PIQI-RCP system in an undelayed scenario, the condition for local stability is independent of link capacity C and number of flows N in the system.*

Proof. In the case of PIQI-RCP, the source controller is:

$$x_i(n) = x_i(n-1) - \tau_1(x_i(n-1) - R(n-1)) + \tau_2(R(n-1) - R(n-2))$$

and the linearized router controller is:

$$R(n) = R(n-1) + \frac{\alpha_1(\gamma C - y(n))}{N} + \frac{\alpha_2(\gamma C - y(n-1))}{N}.$$

At equilibrium,

$$R(n) = x_i(n) = R^* = \frac{\gamma C}{N}. \tag{4.77}$$

The input traffic rate $y(n)$ seen at the router is:

$$\begin{aligned} y(n) &= \sum_{i=1}^{N} x_i(n) \\ &= (1-\tau_1)\sum_{i=1}^{N} x_i(n-1) + N(\tau_1+\tau_2)R(n-1) - \tau_2 N R(n-2) \\ &= (1-\tau_1)y(n-1) + N(\tau_1+\tau_2)R(n-1) - \tau_2 N R(n-2) \end{aligned} \tag{4.78}$$

Substituting the value of $y(n)$ in the router control equation and assuming $\alpha_1 = \alpha_2 = \alpha$, we have:

$$\begin{aligned} R(n) &= (1-\alpha(\tau_1+\tau_2))R(n-1) - \left[\frac{\alpha(1-\tau_1)+\alpha}{N}\right] y(n-1) \\ &\quad +\alpha\tau_2 R(n-2) + \frac{\gamma\alpha C}{N} \end{aligned} \tag{4.79}$$

Hence, the overall system dynamics in matrix form can be expressed as:

$$\begin{bmatrix} R(n) \\ y(n) \\ R(n-1) \end{bmatrix} = \begin{bmatrix} (1-\alpha(\tau_1+\tau_2)) & \frac{-\alpha(1-\tau_1)-\alpha}{N} & \alpha\tau_2 \\ N(\tau_1+\tau_2) & 1-\tau_1 & -N\tau_2 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} R(n-1) \\ y(n-1) \\ R(n-2) \end{bmatrix} \tag{4.80}$$

Using symbolic analysis toolbox in MATLAB, we find that the above system dynamics has 3 eigenvalues λ_1, λ_2, λ_3, which are a complicated polynomial function of α, τ_1, and τ_2. The eigenvalues and hence the stability conditions are independent of link capacity C and number of flows N in the system. □

Since, we are mainly interested in stability analysis in the presence of delay, we next carry out the same. The linearized controller (4.14) in the discrete-time domain

can be written as:

$$R(n) = R(n-T) + \frac{\alpha T e(n)}{Nd} + \frac{\alpha T e(n-T)}{Nd}. \tag{4.81}$$

Taking the z-transform of both sides of the equation, the transfer function $C(z) = R(z)/e(z)$ is:

$$C(z) = \frac{\alpha T(1+z^{-T})}{Nd(1-z^{-T})}. \tag{4.82}$$

The plant consists of N flows and each adjusts its sending rate using (4.19). The total input traffic rate $y(n)$ observed at the router is given by:

$$\begin{aligned} y(n) &= \sum_{i=1}^{N} x_i(n - D_i^{\rightarrow}) \\ &= (1-\tau_1)y(n-T) + (\tau_1+\tau_2)\sum_{i=1}^{N} R_l(n-D_i) \\ &\quad -\tau_2 \sum_{i=1}^{N} R_l(n-T-D_i). \end{aligned} \tag{4.83}$$

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:

$$G(z) = \frac{(\tau_1+\tau_2) - \tau_2 z^{-T}}{1-(1-\tau_1)z^{-T}} \sum_{i=1}^{N} z^{-D_i}. \tag{4.84}$$

The overall open loop transfer function $T_f(z) = C(z)G(z)$ combining the controller and the plant can be obtained from equation (4.82) and (4.84) as given below:

$$\begin{aligned} T_f(z) &= \left[\frac{\tau_1+\tau_2+\tau_1 z^{-T} - \tau_2 z^{-2T}}{1-(1-\tau_1)z^{-T}}\right] \sum_{i=1}^{N} \frac{\alpha T}{Nd} \frac{z^{-D_i}}{1-z^{-T}} \\ &= T(z)T_D(z), \end{aligned} \tag{4.85}$$

where,

$$\begin{aligned} T(z) &= \left[\frac{\tau_1+\tau_2+\tau_1 z^{-T}-\tau_2 z^{-2T}}{1-(1-\tau_1)z^{-T}}\right] \\ T_D(z) &= \sum_{i=1}^{N}\frac{\alpha T}{Nd}\frac{z^{-D_i}}{1-z^{-T}}. \end{aligned} \tag{4.86}$$

In the frequency domain, we have:

$$\begin{aligned} T_f(e^{j\omega}) &= T_D(e^{j\omega})T(e^{j\omega}) \\ T(e^{j\omega}) &= \left[\frac{\tau_1+\tau_2+\tau_1 e^{-j\omega T}-\tau_2 e^{-j\omega 2T}}{1-(1-\tau_1)e^{-j\omega T}}\right] \\ T_D(e^{j\omega}) &= \sum_{i=1}^{N}\frac{\alpha T}{Nd}\frac{e^{-j\omega D_i}}{1-e^{-j\omega T}} \end{aligned} \tag{4.87}$$

The function $T(e^{j\omega})$ crosses the real axis for $\omega_1' = 0$ at $\mathrm{Re}[T(e^{j\omega})] = 2\alpha\tau_1/\tau_1 = 2\alpha$ and for ω_2' given by:

$$\omega_2' = \left[\frac{\tau_1^2 - 2\tau_1 + 2\tau_1\tau_2}{T^2\tau_2(1-\tau_1)}\right]^{\frac{1}{2}}. \tag{4.88}$$

However, for $0 < \tau_1 < 1$ the value of ω_2' is imaginary for $0 < \tau_1 + 2\tau_2 < 2$ and so $T(e^{j\omega})$ cannot cross the real axis for ω_2'. Hence, selecting small values of τ_1 and τ_2 (such as $\tau_1 = 0.01$ and $\tau_2 = 0.1$) that we want in order to restrict overflowing the queue significantly, we can enforce that $T(e^{j\omega})$ crosses the real axis only for $\omega_1' = 0$.

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

$$T_f(e^{j\omega}) = \sum_{i=1}^{N}\frac{2\alpha\tau_1 T}{Nd\tau_1}\frac{e^{-j\omega D_i}}{1-e^{-j\omega T}} = \sum_{i=1}^{N}\frac{2\alpha T}{Nd}\frac{e^{-j\omega D_i}}{1-e^{-j\omega T}}. \tag{4.89}$$

Theorem 12. *For a PIQI-RCP system with $T/D \approx 0$, the necessary and sufficient condition for local stability in the case of flows with homogeneous RTTs D is:*

$$0 < \frac{\alpha}{D} < \sin\left(\frac{\pi}{2(2D-1)}\right). \tag{4.90}$$

Proof. Using $T_f(e^{j\omega})$ in (4.89) and the analysis in the proof of theorem 7 gives the

condition for stability. □

Theorem 13. *For a PIQI-RCP system with $T/D \approx 0$, a sufficient condition for local stability in the case of flows with heterogeneous RTTs is:*

$$0 < \frac{\alpha}{d} < \sin\left(\frac{\pi}{2(2D-1)}\right), \tag{4.91}$$

where $D = \max\{D_1, D_2, \ldots, D_N\}$ and d is the average RTT of the system.

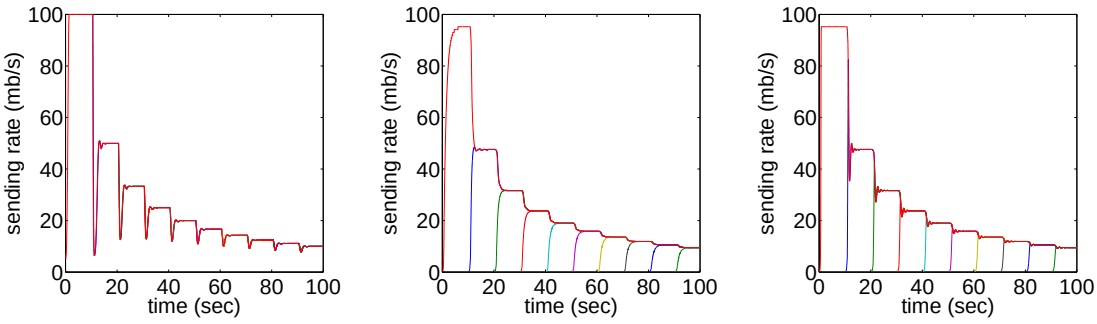
Proof. Using $T_f(e^{j\omega})$ in (4.89) and the analysis in the proof of theorem 8 gives the condition for stability. □

E. Simulations

In this section, we study the performance of PIQI-RCP in various simulation setups and also compare it with RCP. PIQI-RCP with source controller invoked per ACK is referred to as PIQI-RCP-ACK. The different parameters selected during the simulation unless specified otherwise are $\tau_1 = 0.01$, $\tau_2 = 0.1$, $\alpha = 0.5$, $\beta = 1$, $T = 10$ ms, and $\gamma = 0.95$. All simulations involve packet size of 1000 bytes. We unsynchronize the control interval of routers by randomizing the time when the first control interval starts.

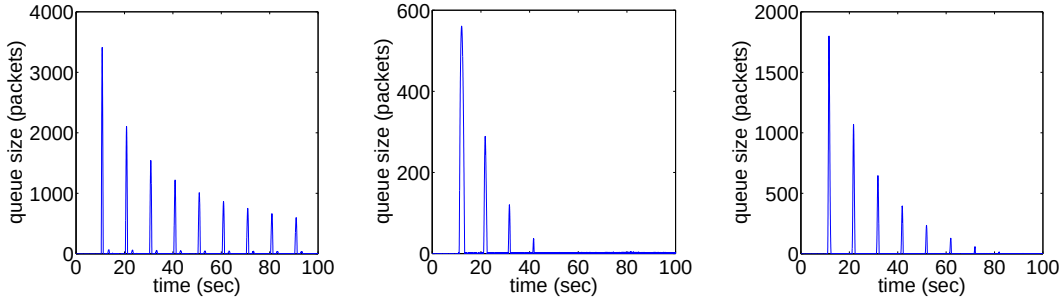
1. Single-Bottleneck Topology

Consider a dumb-bell topology with bottleneck link of capacity 100 mb/s and delay 50 ms. The access links have capacity 1 gb/s. Every 10 second, a new flow enters the system. The access link of flow x_1 has a delay of 10 ms while the access links of remaining flows $x_2, \ldots, x_{10}$ have a delay of 100 ms. Hence, the RTT of x_1 is 120 ms while the RTT of the remaining flows is 300 ms. The sending rates of flows are shown in Fig. 20 while the queue size at the bottleneck link is shown in Fig. 21. In the case of



(a) RCP (b) PIQI-RCP (c) PIQI-RCP-ACK

Fig. 20. Sending rate in the case of single-bottleneck topology.



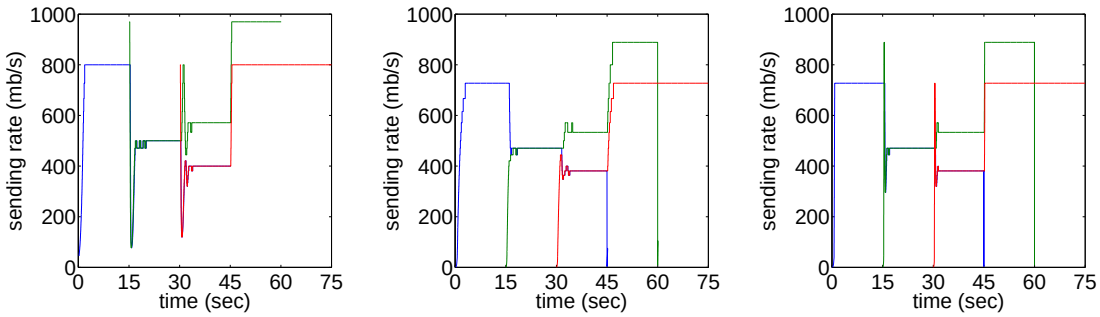
(a) RCP (b) PIQI-RCP (c) PIQI-RCP-ACK

Fig. 21. Queue size in the case of single-bottleneck topology.

RCP, the peak queue size shoots to nearly around 3500 packets with overflow in queue for every new flow entering the system. While for PIQI-RCP, the peak queue size is only 550 packets with the queue remaining close to 0 after $t = 40$. PIQI-RCP-ACK has a better convergence time as compared to PIQI-RCP at the expense of higher peak queue size. A similar behavior is also observed for bottleneck link capacity 1 gb/s and 10 gb/s.

2. Multiple-Bottleneck Topology

Consider a parking-lot topology, 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



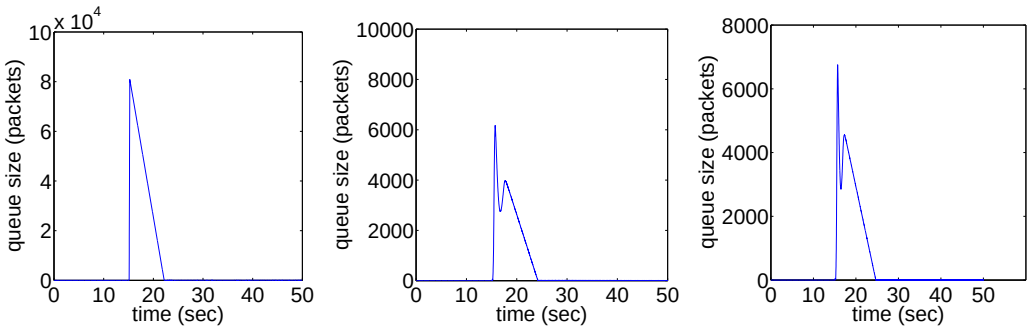
(a) RCP (b) PIQI-RCP (c) PIQI-RCP-ACK

Fig. 22. Comparison in multi-link topology.

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 is shown in Fig. 22. 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 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 figure, the magnitude of transient oscillations is much smaller in PIQI-RCP compared to RCP, while the convergence time is almost the same.

3. Abrupt Increase in Traffic Demand

Consider a dumb-bell topology with bottleneck link capacity 100 mb/s and delay 50 ms. The access links have capacity 10 gb/s and delay 10 ms. Flow x_1 enters the system at $t = 0$ and flows $x_2, \ldots, x_{51}$ join the system simultaneously at $t = 15$. The queuing dynamics at the router is shown in Fig. 23. In the case of RCP, the router queue jumps to 80868 packets. While for PIQI-RCP and PIQI-RCP-ACK, this value



(a) RCP (b) PIQI-RCP (c) PIQI-RCP-ACK

Fig. 23. Queue size at the router in the case of abrupt increase in traffic demand at $t = 15$.

is 6173 and 6756 packets respectively.

4. Peak Queue Size

In this section, we compare the peak queue size for RCP, PIQI-RCP, and PIQI-RCP-ACK for the topology considered in the previous simulation setup when about N flows enter the system simultaneously at $t = 15$. The comparison is shown in Fig. 24. PIQI-RCP and PIQI-RCP-ACK have a significantly smaller peak queue size as compared to RCP. For example, when 250 flows enter the system simultaneously, RCP has a peak queue size of 501014 packets while PIQI-RCP and PIQI-RCP-ACK have a peak queue size of nearly 14000 packets, i.e., lower by a factor of 36. Also, the peak queue size in RCP keeps on increasing with N but stabilizes for PIQI-RCP and PIQI-RCP-ACK.

5. Average Flow Completion Time

In this section, we compare the performance of RCP, XCP, TCP, PIQI-RCP, and PIQI-RCP-ACK considering Average Flow Completion Time (AFCT) as the metric. Intuitively, RCP will fare better in this case since new flows entering the system are

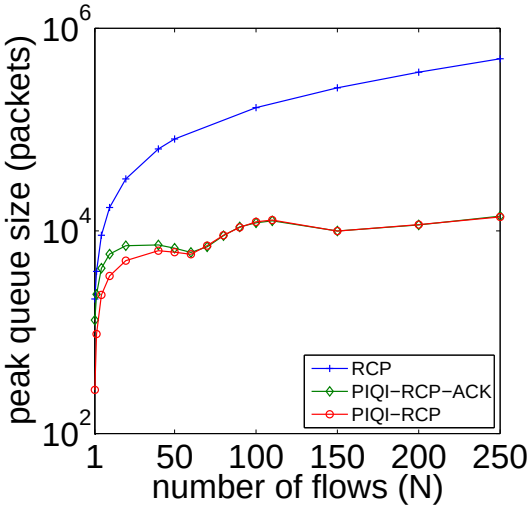
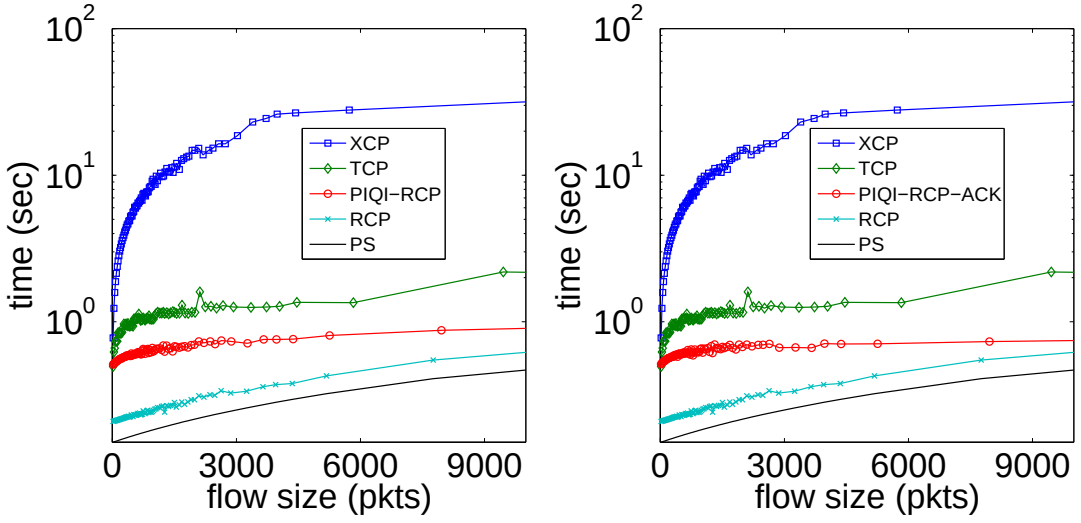


Fig. 24. Comparison of peak queue size.



(a) PIQI-RCP (b) PIQI-RCP-ACK

Fig. 25. Comparison of average flow completion time.

given a rate equal to the current control rate at the bottleneck router while new flows in TCP, XCP, PIQI-RCP, and PIQI-RCP-ACK increase their sending rate gradually. Consider a simulation scenario with a single bottleneck link of capacity 2.4 gb/s and round-trip propagation delay of 100 ms. New flows enter the system as Poisson arrivals with Pareto distributed flow sizes of mean 30 pkts and shape parameter 1.4. The offered load is 0.9. The comparison is shown in Fig. 25. It can be seen that PIQI-RCP and PIQI-RCP-ACK have a lower (higher) AFCT as compared to TCP and XCP (RCP).

F. Summary of Results

From the analysis and simulation results as shown in this chapter, we infer that PIQI-RCP overcomes the stability issue of RCP and simultaneously has a significantly lower buffering requirement at the routers. The parameters τ_1, τ_2 in the source controller of PIQI-RCP can be tuned to further reduce the buffering requirement at the expense of convergence time. PIQI-RCP compromises slightly on average flow completion time which still happens to be better compared to TCP and XCP.

CHAPTER V

LINUX IMPLEMENTATION

The first step in experimental evaluation is to design an efficient implementation that can perform well in high-speed networks. We present implementation details of XCP, JetMax, RCP, and PIQI-RCP in this chapter and show their performance in gigabit networks in the following chapter.

For our implementation, we use Linux kernel 2.6.12 available with Fedore Core 4 [11] Linux distribution. To compute feedback more accurately, we recompile the kernel by making relevant changes to the *Makefile* to support floating point operations. Starting Linux kernel 2.6.0, kernel threads can be pre-empted. Whenever a kernel thread is pre-empted, the floating point registers are not saved due to additional memory overheads. Hence, portions of our code that require floating point computations have been protected by locking the CPU to prevent kernel thread preemption. CPU locking and unlocking can be done using the get_cpu() and put_cpu() kernel routines. Linux, similar to various other UNIX variants such as FreeBSD [16], also supports the concept of modules, which are pieces of binary code that can be dynamically plugged in (out) the running kernel to support (unsupport) different features and protocols without rebooting the system. We implement most of the end-host and AQM-router functionality as modules and their implementation details follow next.

A. End-Host

End-hosts apply changes to their congestion window or sending rate using the feedback received from their bottleneck router. To facilitate the computation of feedback at the routers, end-hosts provide them congestion-related information such as their

current RTT estimate, congestion window or sending rate. Inspired by the discussions in [10], we decide to employ a new header called *congestion header* that can be placed between the transport (e.g., TCP) and the IP headers of a packet. Fields within the congestion header are used to communicate congestion-related information and feedback between end-hosts and routers. The congestion header is placed (removed) in every outgoing (incoming) packet by the end-host module. Other possibilities that have been suggested to communicate congestion-related information and feedback involve using TCP Options [47], [54] or IP Options. More about the drawbacks of these possibilities can be found in [10].

We assign different dummy protocol numbers in the IP header for XCP, JetMax, RCP, and PIQI-RCP. The actual protocol numbers need to be assigned by Internet Assigned Numbers Authority (IANA) [20] when these protocols are deployed. We choose these numbers to be 201, 202, 203, and 204 for XCP, JetMax, RCP, and PIQI-RCP respectively. End-host modules register (unregister) themselves with the TCP/IP stack corresponding to their protocol number whenever they are loaded (unloaded) into the kernel. Every incoming packet from the Network Interface Card (NIC) driver is passed to the IP layer for processing. Depending upon the protocol number, the IP layer passes the packet to the corresponding module that has registered for it. The end-host module captures the feedback information in the congestion header and applies the corresponding changes using its control equation. If the end-host is a data receiver, the module saves the feedback in the congestion header and returns it back to the sender as part of acknowledgement packets. Similarly, every outgoing packet from the application layer is processed by the TCP layer, which delegates the packet transmission to the default end-host module for appending the congestion header and filling in the congestion-related information. The end-host module then hands over the packet to the IP layer for further processing and finally it is passed to the NIC

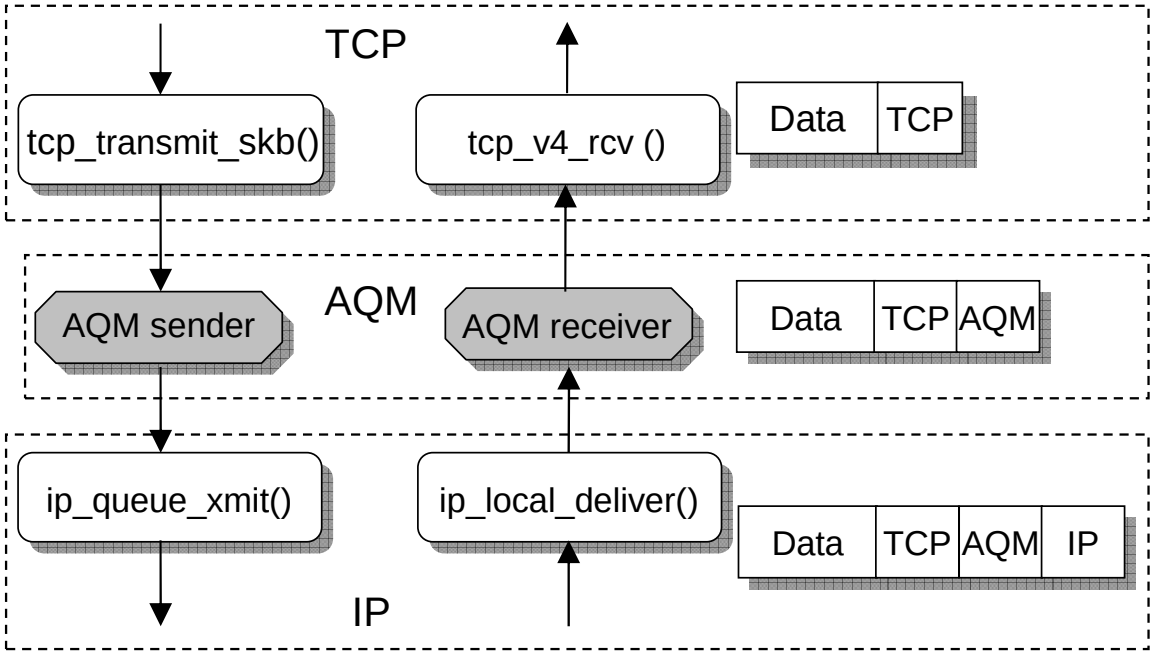


Fig. 26. Implementation methodology of explicit-feedback congestion control.

driver for transmission. This mechanism and the corresponding function entry/exit points at different layers are shown in Fig. 26.

Since most of the operations as mentioned above are done inside the core kernel and modules, they are transparent to the applications. This design facilitates deployment of these protocols in future networks without requiring any change to the current applications. We next provide more details about the implementation of window-based and rate-based schemes for end-hosts.

1. Window-Based Schemes

Since TCP is a window-based transport protocol, very few changes are required to the core kernel to implement window-based congestion control algorithms such as XCP. We implement most of the end-host congestion control functionality for XCP as part of a kernel module. TCP's slow start and congestion avoidance mode are

disabled so that they do not interfere with XCP's control algorithm. To implement rate-based congestion control algorithms in the window-based TCP/IP stack, changes are required to the kernel source code. More information about these changes follow next.

2. Rate-Based Schemes

Implementation of rate-based schemes (e.g., JetMax, RCP, and PIQI-RCP) utilize a similar methodology as described above. Additionally, we modify the TCP/IP stack to make its original window-based data transfer operation rate-based. The basic idea is to pace the transmission of data packets at the rate computed by the AQM module based on the network feedback from the routers. To accomplish this goal, it is necessary to first understand how data is transmitted in the original TCP/IP stack. Data sent by the application layer is sliced into chunks of MSS (Maximum Segment Size) and queued into a buffer by the transport layer. To each chunk is appended a transport header. The normal behavior is to transmit them *instantaneously* when the number of packets in the flight is less than the congestion window and the advertised receiver window. We disable this immediate transmission for data packets, but allow control packets (such as SYN, FIN, and RST) to be transmitted immediately.

In addition, we implement a control timer function to periodically process the queue holding data packets. When the timer expires, the timer interval and the number of data packets to transfer per instance are recalculated based on the designated data rate. Since a queued data packet cannot be sliced for transfer, to maintain the exact sending rate, the control interval is adjusted properly to take this into account. Moreover, even though Linux supports delaying an event to a resolution of 1 μs, we chose the minimum timer interval to be 1 ms, since these small resolutions are achieved using CPU busy waiting and waste a lot of CPU cycles. This also helps to reduce the

load on the system when thousands of connections are invoked concurrently.

B. Router

Routers in explicit congestion control algorithms need to provide feedback to end-hosts indicating the actual level of congestion. They gather information about the input traffic rate and queue size by per-packet processing and then feed this information into a control equation at regular intervals to generate feedback that is inserted into every packet's congestion header. To realize this functionality, methods [54] suggested in the past involve using Qdisc. However, for a flexible implementation, we take advantage of netfilter [41], which is an excellent packet filtering framework in the Linux network stack commonly used to develop firewall software such as *Iptables*. Using *netfilter*, custom user defined functions known as "hook" can be invoked at five different places in the IP and Route module of the network stack. These points are clearly illustrated in Fig. 27 and are described below:

- *NF_IP_PRE_ROUTING:* Incoming packets entering the IP layer but before getting processed in the Route module can be intercepted by registering hook functions at this point.

- *NF_IP_LOCAL_IN:* Incoming packets, to be delivered to the local host, after being processed at the Route module can be intercepted at this point before delivering them to the transport layer higher up.

- *NF_IP_FORWARD:* Incoming packets, which should be forwarded to another host after being processed at the Route module can be intercepted at this point.

- *NF_IP_LOCAL_OUT:* Outgoing packets from the local host can be intercepted at this point before they get processed by the Route module.

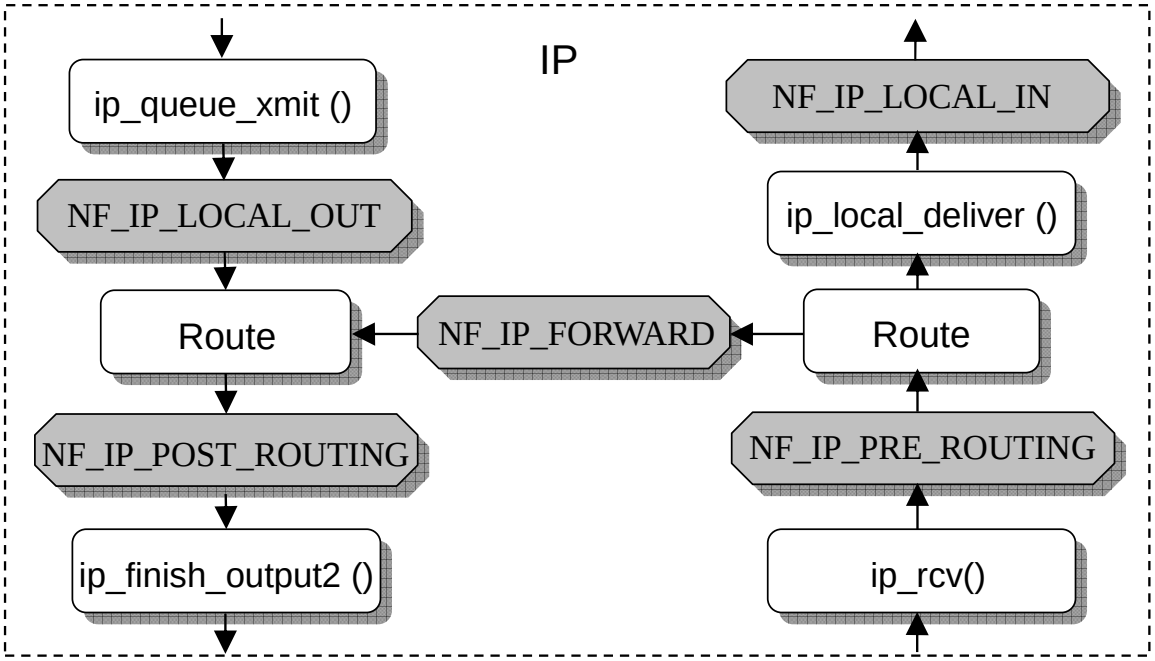


Fig. 27. Illustration diagram for different netfilter hooks in Linux TCP/IP stack.

- *NF_IP_POST_ROUTING:* Outgoing packets from the local host or incoming packets that have to be forwarded can be intercepted at this point after they have been processed by the Route module.

We develop modules to implement the AQM functionality of XCP, JetMax, RCP, and PIQI-RCP. Each module has a hook function to intercept packets at *NF_IP_POST_ROUTING*. At this location, outgoing packets from the local machine or incoming packets from other network interfaces that need to be forwarded are processed. We assign the hook function the lowest priority so that they are invoked only after all the kernel routines for processing the packet have been finished. The hook function collects the information present in the congestion header of the packets, update the module's data structure with this information, and insert feedback into the packets. A timer function is used to invoke the AQM-router's control equation at regular intervals and generate the feedback signal that would be inserted into the

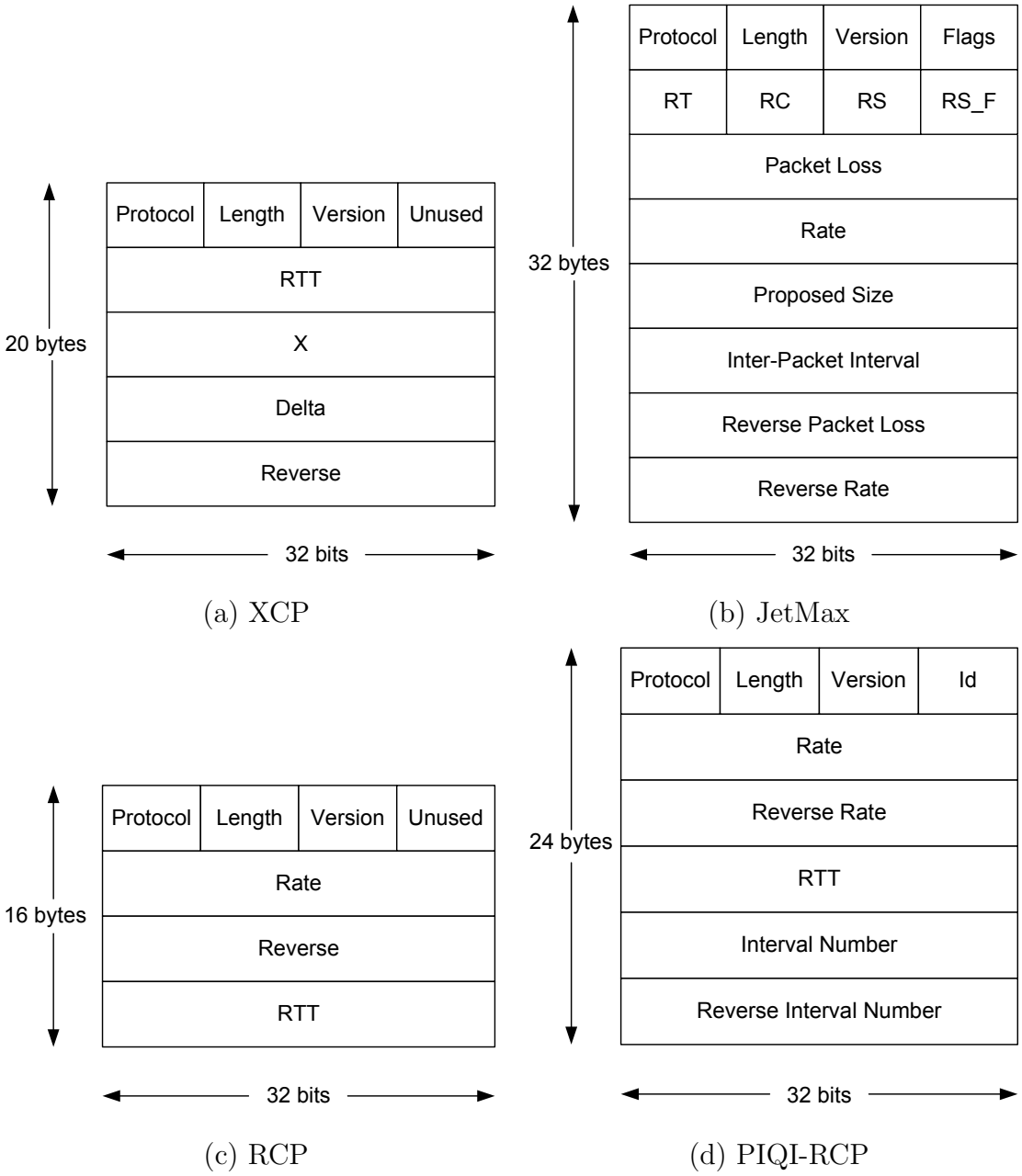
packets during the subsequent interval. For XCP, this interval is the average round-trip time of flows passing through the router, for RCP and PIQI-RCP it is 10 msec, and for JetMax it is 100 msec.

C. Congestion Header Format

The congestion header for XCP, JetMax, RCP, and PIQI-RCP used in our implementation is shown in Fig. 28. The common fields in all the headers include: 1) **Protocol** is the protocol number of the Transport layer above the AQM-layer. For example, for TCP this value is 6; 2) **Length** is the size in bytes of the congestion header between the TCP and the IP header; 3) **Version** is the protocol version of the AQM algorithm; and 4) **Unused** may be required later for possible protocol extensions. Its value should be set to zero.

For XCP, the congestion header is identical to the one suggested in [10]. **RTT** represents the end-host's current estimate of round-trip time measured in msec. **X** stores the inter-packet transmission delay of a flow measured in msec. **Delta** represents the desired throughput of a flow expressed in bytes per msec. Routers modify this field to represent the allocated change in throughput expressed in bytes per msec. Data receivers copy the value stored in the **Delta** field of the received packets into the **Reverse** field and send it in acknowledgement packets. The total size of XCP's congestion header is 20 bytes.

JetMax's congestion header is elaborately discussed in [56]. **RT** stores the router Id of the current bottleneck router. The value of **RC** is incremented at every router encountered in a flow's path. It helps in calculating the value of RT and RS. **Packet Loss** is used to store the virtual packet loss rate at routers as the packet passes through them. Router, which has the highest packet loss rate is considered as the



(a) XCP (b) JetMax

(c) RCP (d) PIQI-RCP

Fig. 28. Congestion header format.

bottleneck router of a flow. Bottleneck switching takes place whenever a router has a packet loss rate that is higher than the current bottleneck router of a flow. **RS** stores the router Id of the bottleneck router when a bottleneck switch is detected. Data receivers copy the value of RS in the received packets into the **RS_F** field and send it in acknowledgement packets. **Rate** is modified by the bottleneck routers and is used to assign the allocated sending rate. **Proposed Size** is used by flows to propose a new sending rate and request routers in the path for approval. **Inter-Packet Interval** field stores inter-packet transmission delay for a flow expressed in msec. It assists router's estimation of number of flows in the system. Data receivers copy the value of **Packet Loss** and **Rate** in the received packets to **Reverse Packet Loss** and **Reverse Rate** field respectively and send it in acknowledgement packets. The total size of JetMax's congestion header is 32 bytes.

In RCP's congestion header, **Rate** field is modified by routers to assign the control rate to a flow as feedback. **RTT** represents the end-host's current estimate of round-trip time measured in msec. If the end-host is a data receiver, it copies the value stored in the **Rate** field of the received packets into the **Reverse** field and sends it in acknowledgement packets. The total size of RCP's congestion header is 16 bytes.

PIQI-RCP's congestion header is similar to that of RCP except for addition of three new fields such as **Interval Number**, **Reverse Interval Number**, and **ID**. **Interval Number** is assigned by the router to indicate the control interval number corresponding to the assigned **Rate** feedback value. This field is copied into the **Reverse Interval Number** field while sending the acknowledgement packets. **Interval Number** helps the end-host to identify the uniqueness of the received feedback and invoke the control algorithm once per the router control interval. **ID** is the id of the bottleneck router. The total size of PIQI-RCP's congestion header is 24

bytes.

D. Kernel Tuning

The default Linux kernel's network stack has many parameters that require tuning in order to support gigabit throughput for wide-range of RTTs. This is required to fairly evaluate the limitations of the protocol. We increase the maximum size of both socket read and write buffers (**rmem_max, wmem_max**) from a default value of 131071 bytes to 107374182 bytes, per-connection memory space defaults (**tcp_rmem, tcp_wmem, tcp_mem**) from (4096, 87380, 174760) bytes to (4096, 107374182, 107374182) bytes, size of backlog queue (**netdev_max_backlog**) in the receive path from 300 to 10000 slots, and transmit queue in the forward path (**txqueuelen**) from 1000 to 10000 slots. The size of transmit and receive ring buffers of the NIC was also increased from 256 slots to the maximum possible value of 4096 slots. This is mainly required to absorb bursts of incoming packets during gigabit transfers in the case of XCP. We also disable the TCP segmentation offload and checksum verification feature of NICs to support a new congestion header for our implementation. More information about the Linux network stack can be found in [17].

Using the implementation described in this chapter, we next provide our experimental results in the following chapter.

CHAPTER VI

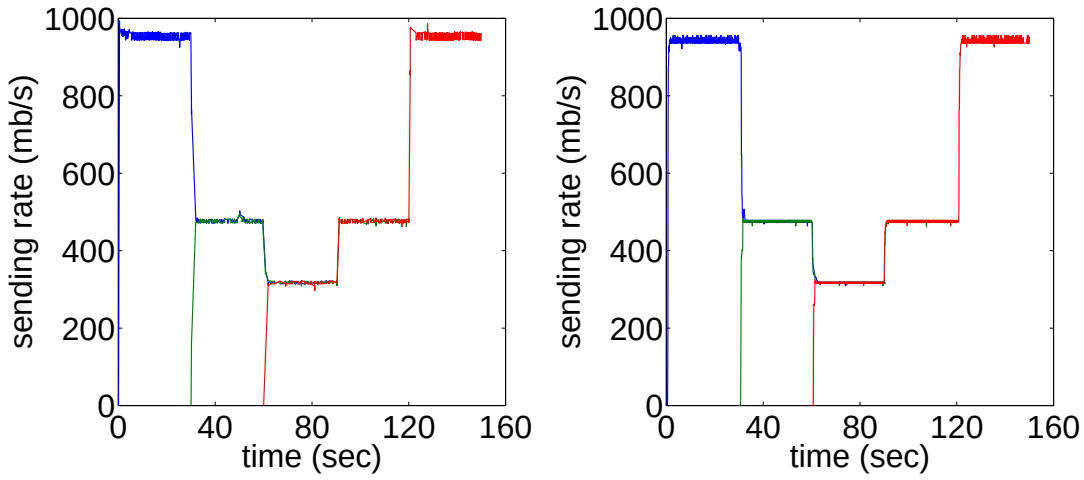
LINUX EXPERIMENTS

In this chapter, we describe the results of experiments we conducted in different setups using our implementation of XCP, JetMax, RCP, and PIQI-RCP. Our goal is to demonstrate that efficient implementation of explicit congestion control algorithms can be realized in high-speed networks and verify several key properties of these protocols. All experiments were performed in Emulab [9] using Dell Poweredge 2850 servers with 3.0 GHz 64-bit Xeon processors, 2 GB of RAM, and multiple gigabit network cards. Throughout this chapter, we set parameters $\alpha = 0.4$ and $\beta = 0.226$ for XCP, $\tau = 0.6$ for JetMax, $\alpha = 0.1$ and $\beta = 1$ for RCP, and $\alpha = 0.5$, $\tau_1 = 0.01$, $\tau_2 = 0.1$, $\gamma = 0.95$ for PIQI-RCP. In all the plots, the sending rate is obtained by averaging the IP layer throughput every two round-trip times (RTTs) for JetMax, RCP, and PIQI-RCP and is approximated by cwnd/srtt for XCP using packet size of 1500 bytes. For link capacity of 1 gb/s, the achievable IP layer throughput is 970 mb/s.

A. Experiments

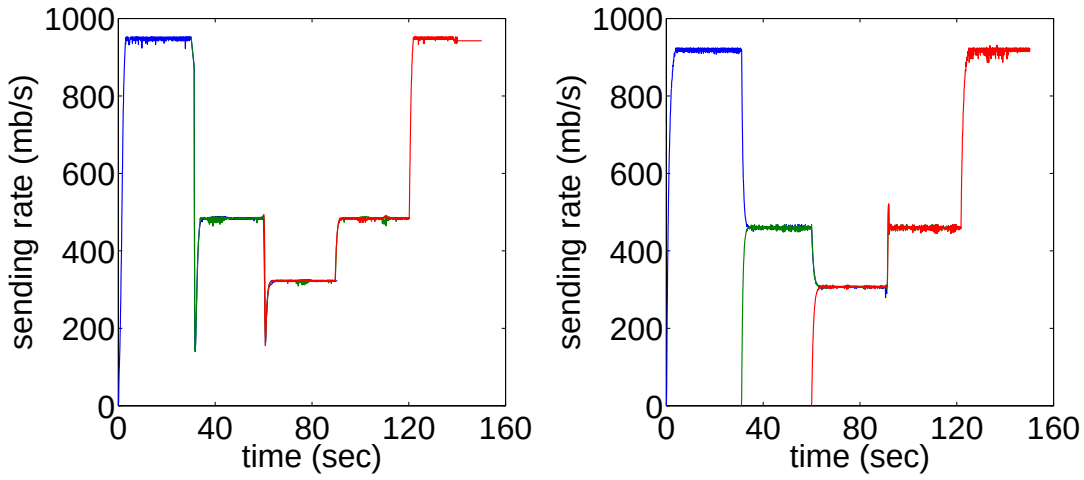
1. Single-Bottleneck Topology

We first examine the performance of these protocols in high-speed networks with a single bottleneck link. Consider a dumb-bell topology 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 through a different access link of capacity 1 gb/s and negligible delay. Flows start at $t = 0$, 30, 60 and each lasts for 90 seconds.



(a) XCP

(b) JetMax



(c) RCP

(d) PIQI-RCP

Fig. 29. Performance in single-bottleneck topology with link capacity 1 gb/s and RTT 50 ms. Flows start at $t = 0$, 30, and 60. Each flow lasts for 90 seconds.

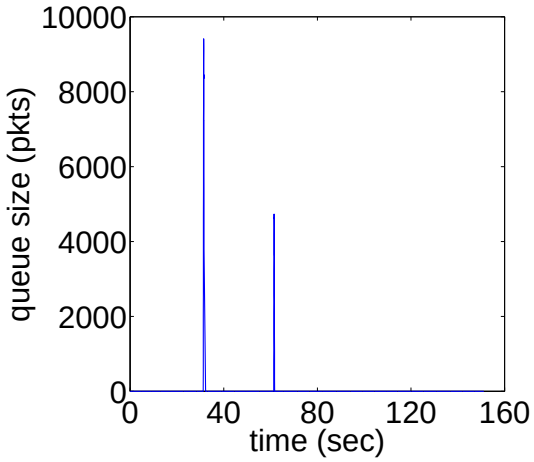


Fig. 30. Queuing dynamics of RCP in single-bottleneck topology.

Dynamics of the actual sending rates of these protocols are illustrated in Fig. 29. All the methods are able to maintain high sending rates and the router can easily process the traffic. During the experiment, we also monitor the IP layer queue inside the bottleneck router and find that XCP and JetMax are successful in controlling their queue length at very low levels. However, as shown in Fig. 30, RCP experiences significant queue buildup (up to 9415 packets) when new flows join the system. This phenomenon is attributable to the fact that when a flow starts, the bottleneck router cannot tell whether it is a new flow and will assign the old (i.e., before the flow joins) fair rate to this flow, which immediately increases its sending rate to this value upon receiving the feedback and overshoots the bottleneck link capacity. This problem is especially serious in the real Internet, where sessions are regularly joining and leaving the system and the transient queue buildups can potentially overflow any router buffer. RCP-AC [8] has been recently proposed to mitigate this at the expense of increased header size, more per-packet computations inside the router, and compromise on average flow completion time.

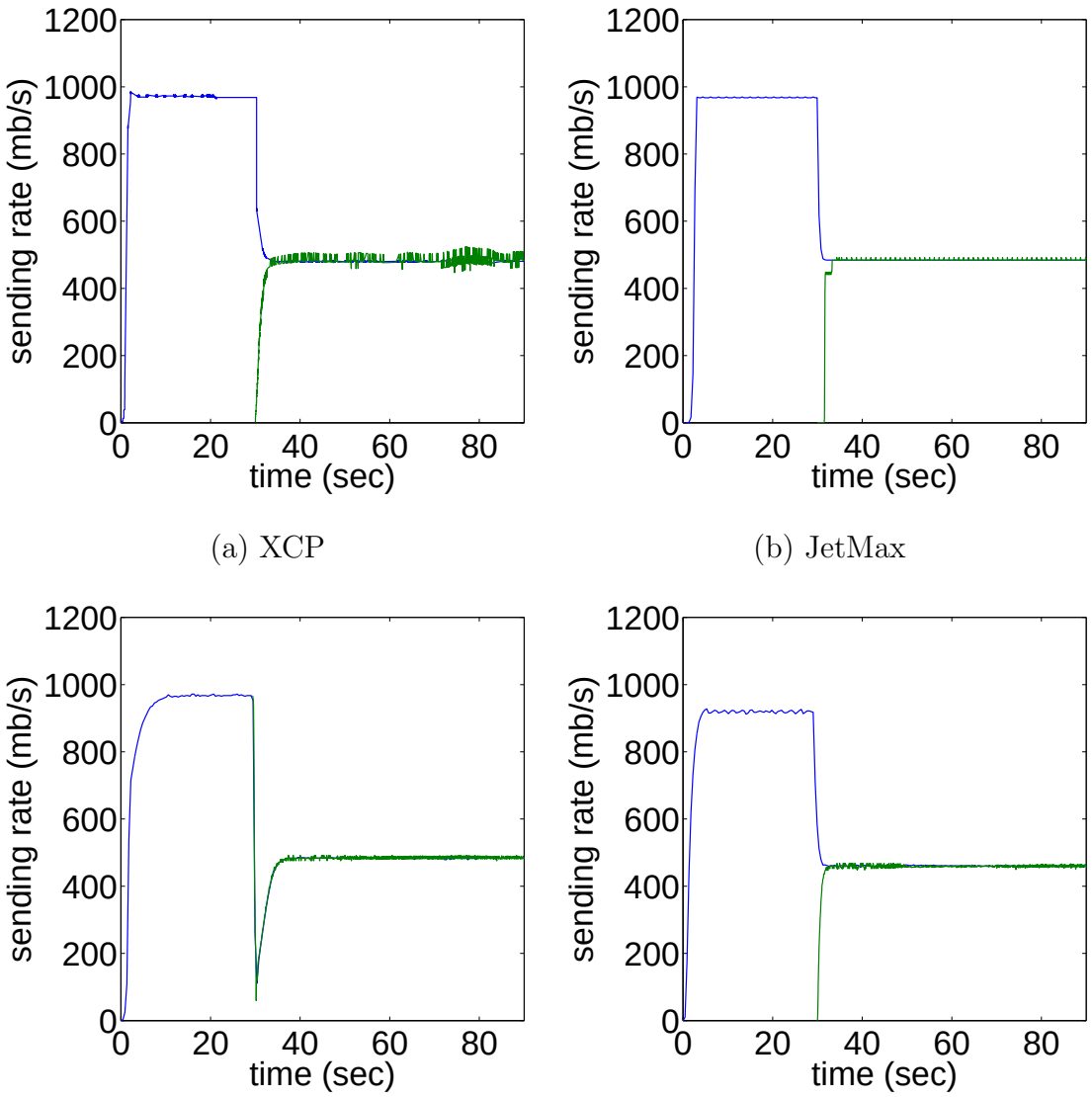
Unlike RCP, PIQI-RCP is able to maintain almost zero queue since new flows

entering the system start with a small sending rate that increases gradually. This allows the router enough time to converge to a new steady-state feedback value without significantly overshooting the queue. As shown in the figure, PIQI-RCP has a slightly smaller throughput as compared to other methods since the router control algorithm operates on the virtual link capacity γC with $\gamma = 0.95$. This is required to drain any queue buildup over a period of time.

2. RTT Unfairness

TCP Reno and many other end-to-end high-speed TCP variants suffer from severe RTT unfairness [53]. Two flows with different RTTs passing through a common bottleneck may share bandwidth in a very unfair fashion. Explicit congestion control algorithms based on max-min fairness are very robust in this scenario. Flows bottlenecked at a common router share equal rates irrespective of their RTTs. Although this has already been shown extensively in ns-2 simulations, we confirm this in a practical scenario. In this experiment, we use a dumb-bell topology where the bottleneck link has a capacity of 1 gb/s and propagation delay of 15 ms. The access link for flow x_1 connecting to the bottleneck router has a capacity of 1 gb/s and delay of 95 ms, while the access link for flow x_2 connecting to the bottleneck router has a capacity of 1 gb/s and negligible delay. Hence the round-trip propagation delay of flow x_1 is 220 ms and that of flow x_2 is 30 ms, i.e., they differ by a factor of seven.

Fig. 31 shows the dynamics of the two flows. Flow x_1 starts at $t = 0$. In the case of XCP and JetMax, it achieves the bandwidth almost instantaneously. However, for RCP flow x_1 takes nearly 10 seconds to saturate the link capacity. This is because when flow x_1 initially joins the system, the computed rate at the router is very low. As the flow starts sending data, the control algorithm computes a new rate. It takes a number of iterations or control cycles before the router's rate computation gives a



(a) XCP

(b) JetMax

(c) RCP

(d) PIQI-RCP

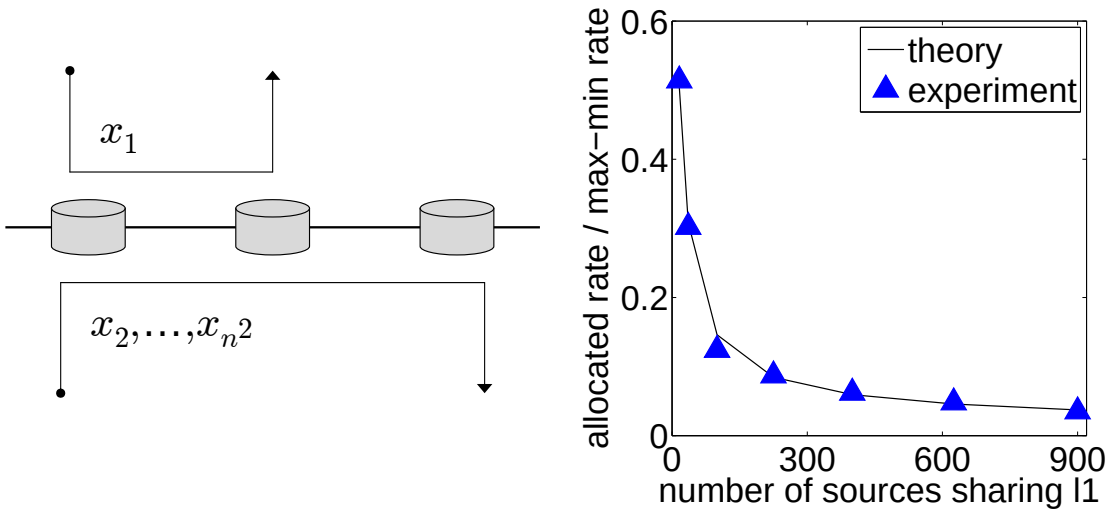
Fig. 31. Performance in the case of flows with heterogeneous RTTs. The bottleneck link capacity is 1 gb/s. Flow x_1 with RTT 220 ms starts at $t = 0$ while flow x_2 with RTT 30 ms joins the system at $t = 30$.

rate close to the link capacity. This can be improved by choosing a higher value of α in the control equation, but doing so would have its own side effects in other scenarios. When flow x_2 joins the system at $t = 30$, the two flows converge to their fair share with equal rates. Again, XCP and JetMax converge almost instantly while RCP takes some time to clear the buildup queue and give both flows their fair share. PIQI-RCP has better convergence time than RCP, where not only does x_1 saturates the virtual link capacity faster, the system converges to its steady-state almost instantaneously when x_2 joins the system.

XCP, being a window-based protocol, emits packets into the network in bursts. To support high throughput, flows with larger RTT have to maintain a large congestion window. Because of these two reasons, flows having small RTT experience high variance in queuing delay. This can be easily seen in Fig. 31(a). Flow x_2, with small round-trip propagation delay entering the system at $t = 30$, experiences small oscillations in its sending rate when co-existing with flow x_1.

3. Scalability

With extensive ns-2 and testbed experiments, we have confirmed that the explicit congestion control algorithms such as XCP, JetMax, RCP, and PIQI-RCP are highly scalable with increase in link capacity and round-trip propagation delay. Even a single flow can easily saturate the link without requiring multiple additional flows. This is in sharp contrast to TCP Reno and many other end-to-end high-speed TCP variants [12], [28], [31], [53], where throughput of a flow is still a function of its RTT. For the sake of brevity, the corresponding plots have not been shown.



(a) topology (b) result

Fig. 32. Experimental verification of XCP's fairness issue identified in [34].

4. Max-min Fairness in XCP

Recall that it is demonstrated in [34] that XCP does not achieve max-min fairness in general and its stationary resource allocation can be arbitrarily unfair in certain topologies. We next verify this by considering the topology shown in Fig. 32(a), which is composed of two links l_1 and l_2 and n^2 flows where n is a given constant. One flow passes only through link l_1 and the other n^2-1 flows traverse both links. In addition, we set link capacities, to be $C_1 = 155$ and $C_2 = C_1(n-1)/n$ mb/s. Using the definition of max-min fairness, the $n^2 - 1$ long flows should be congested at link l_2 with stationary sending rate $x_2^* = 155/n(n+1)$ mb/s and the other flow should converge its sending rate to $x_1^* = 155/n$ mb/s.

To examine whether XCP achieves max-min fairness in this topology, we plot in Fig. 32(b) ratios $\tilde{x}_1/x_1^*$ and $\hat{x}_1/x_1^*$ for different values of n, where $\tilde{x}_1$ is the sending rate of the short flow predicted by the model developed in [34] and $\hat{x}_1$ is the actual sending rate measured in our Linux experiment. Clearly, these ratios indicate how

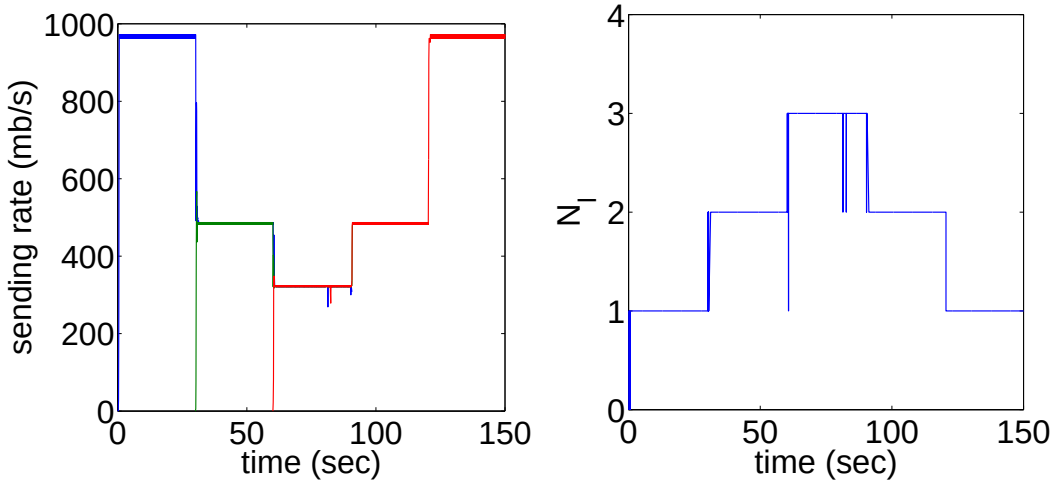
close the system is to max-min fairness, i.e., the closer the ratios are to 1, the more max-min fair the system is. As shown in the figure, the system departs from the max-min fair state when the number of flows increases. Our experimental measurements match the results predicted by the model developed in [34] and the corresponding ns-2 simulations.

5. Effect of Router Control Interval

The length of the router control interval is a tradeoff between the response time and how accurately the control algorithm can capture network dynamics. In the case of XCP, the router control interval is set to be the minimum of average round-trip time of all flows passing though the router and 500 ms, but greater than 10 ms, while for RCP, PIQI-RCP, and JetMax the suggested value is 10 ms, 10 ms, and 100 ms respectively. Most of the experiments in the paper involving JetMax have been done using the suggested value for the router control interval of 100 ms. However, we have confirmed using ns-2 simulations and testbed experiments that JetMax can work well with a control interval as low as 10 ms. There may be small fluctuations in rate but the system quickly stabilizes. Consider the experiment corresponding to the setup shown in Fig. 29 for JetMax but with a router control interval of 10 ms rather than 100 ms. The corresponding rate dynamics of flows and estimation of N_l is shown in Fig. 33. The plots indicate the stability of rate that end-hosts can maintain even with a small router control interval.

6. CPU Usage at Routers

Due to per-packet processing for examining the congestion header of every incoming packet, computation of feedback for every control interval, and stamping feedback on every outgoing packet at the routers, it may be believed that explicit congestion



(a) sending rate (b) estimation of N_l at router

Fig. 33. Sending rates of three JetMax flows sharing a single bottleneck with link capacity 1 gb/s and RTT 50 ms. Flows start at $t = 0$, 30, and 60. Each flow lasts for 90 seconds. The control interval inside the router is 10 ms.

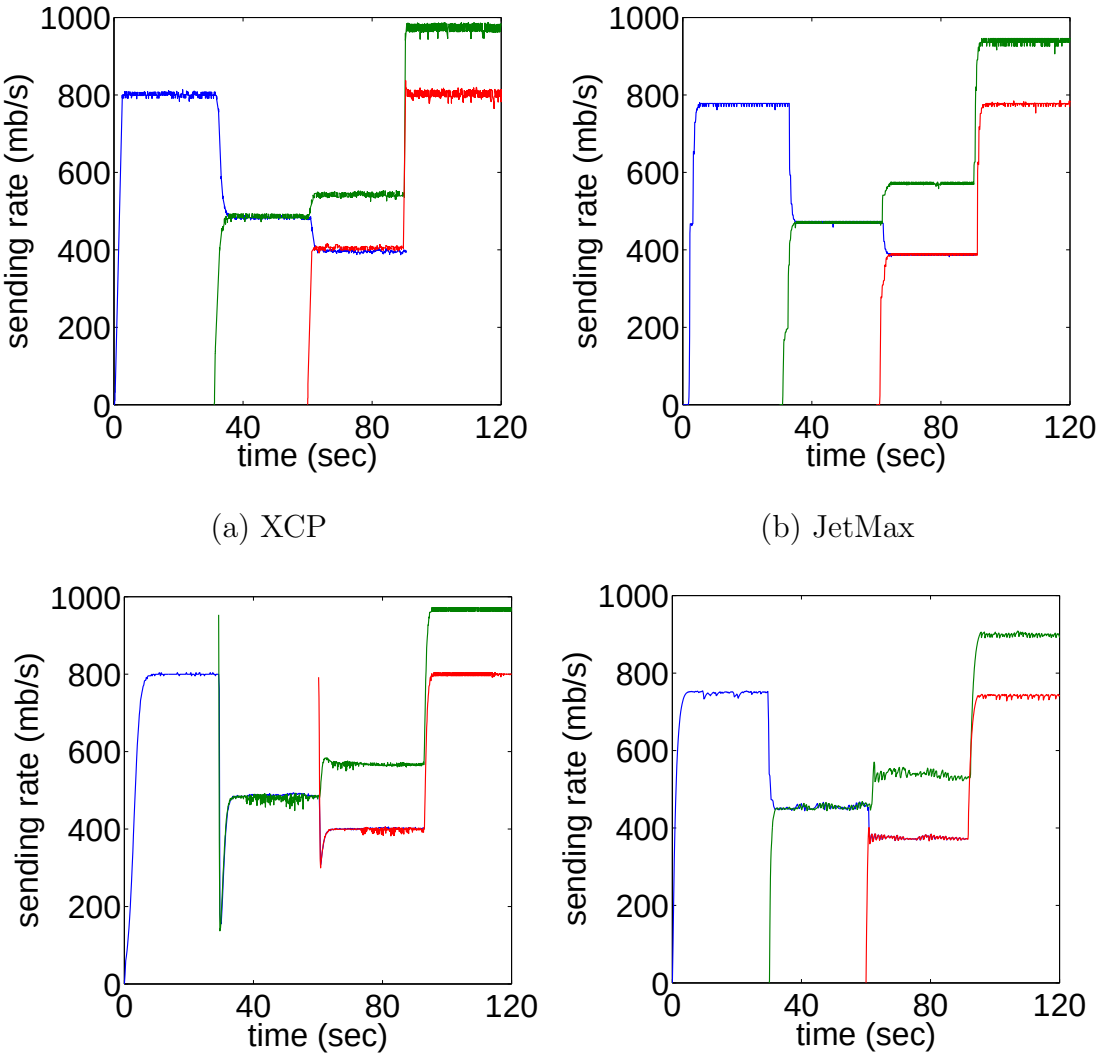
control would involve significant computational overhead, especially in high-capacity links, that can undermine their deployment. However, our experiments show that the overhead involved in these computations is not very significant. Specifically, for a gigabit transfer from one sender to a receiver through a router, the average load on the router is around 30% for all the protocols such as TCP, XCP, JetMax, RCP, and PIQI-RCP. Hence, for 70% of the time, the router is idle. Out of the 30% load, most was utilized in the handling of IRQs (Hardware Interrupt Requests) and Software Interrupts. This indicates that the computational overhead involved is relatively small as compared to processing the interrupts invoked to enqueue and dequeue packets. The experiment was repeated a number of times and in all cases we observed similar behavior.

7. Multiple-Bottleneck Topology

We next examine the performance of these protocols in a parking-lot topology, which is composed of two bottleneck links (l_1, l_2) and three flows (x_1, x_2, x_3). Capacity of these two links are $C_1 = 970$ and $C_2 = 800$ mb/s, and the round-trip propagation delay of each link is 50 ms. Flow x_1 passes through both the links but flows x_2 and x_3 respectively utilize l_1 and l_2. Flow x_1 starts first and converges its rate to the capacity of l_2, i.e., 800 mb/s. When x_2 joins 30 seconds later, x_1 switches its bottleneck to l_1 and both the flows converge to an even share of $C_1/2 = 485$ mb/s. As x_3 starts at time $t = 60$ s, x_1 changes its bottleneck back to l_2 and converges its sending rate together with x_3 to $C_2/2 = 400$ mb/s. Flow x_2 then utilizes the remaining bandwidth on link l_1, i.e., 570 mb/s. Finally, when x_1 terminates at $t = 90$ s, flows x_2 and x_3 change their sending rates to the capacity of each link. As demonstrated in Fig. 34(a)-(d), all methods are stable and max-min fair in this case with their dynamics following the theoretical understanding. In the case of PIQI-RCP, the sending rate of flows $x_1 - x_3$ are scaled by $\gamma = 0.95$ since the router controller operates on the virtual link capacity γC with $\gamma = 0.95$.

8. Performance with Mice Traffic

Studies have shown that the majority of flows in the Internet are short lived flows, called *mice*, that transfer only few packets. Coupled with these short flows, there are a few long-lived flows, called *elephants*, that remain longer in the system as they transfer a large number of packets. In such a scenario, it becomes necessary to understand how the system will perform when the traffic is a combination of both short and long lived flows. The test setup in this case is a dumb-bell topology with 2 senders and 1 receiver. The long flow starts from one of the two sender machines at $t = 0$, while



(a) XCP

(b) JetMax

(c) RCP

(d) PIQI-RCP

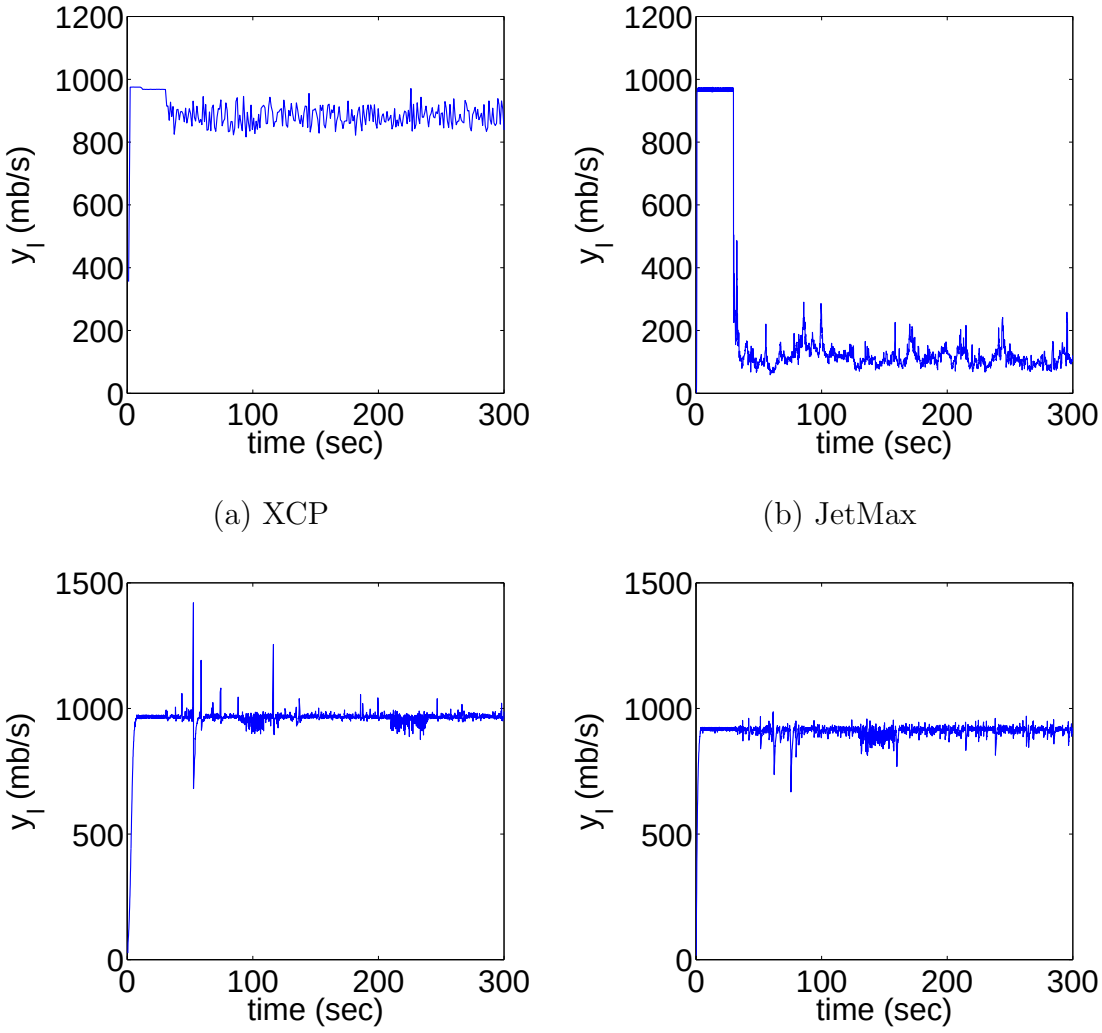
Fig. 34. Comparison in a multiple-bottleneck link topology. Flows start at $t = 0$, 30, and 60. Each flow lasts for 90 seconds. The RTT of flow x_1 is 100 ms while RTT of flow x_2 and x_3 is 50 ms. The capacity of link l_1 and l_2 is 970 and 800 mb/s respectively.

mice traffic is generated from the other at $t = 30$. Both sets of traffic pass through a common bottleneck link with capacity 1 gb/s and round-trip propagation delay of 50 ms. The pattern of mice traffic follows Poisson arrivals with mean inter-arrival time of 0.2 seconds and Pareto distributed traffic size with shape parameter 1.4 and mean of 100 packets. The results are shown in Fig. 35.

In the case of XCP, the link utilization for first 30 seconds is close to 100% when only the long flow is present in the system. But when mice traffic is started, the link utilization drops slightly. The input traffic rate at the router fluctuates between $800 - 970$ mb/s. The loss in link utilization is because a part of the feedback sent by the router to short flows is not utilized as they exit from the system after transferring a small number of packets. The system still operates at high utilization since XCP is conservative in giving bandwidth to new flows entering the system.

In the case of RCP, the link utilization remains very high with occasionally overshooting the link capacity by huge margins for small period of time. This is evident from Fig. 35. For the first 30 seconds, the input traffic rate at the router is very stable around the link capacity. Fluctuations arise in the system after mice traffic comes into play at $t = 30$. New flows entering the system are given the control rate computed in the last control interval. As a result, when many flows enter the system simultaneously, the input traffic rate exceeds the link capacity with high queue levels. This makes the router reduce the rate to be fed back in the next control interval in order to drain the queue, which would temporarily reduce link utilization. This behavior can be seen at $t = 55$. Hence, in the case of RCP, high queue size at the router is recommended in order to absorb the sudden rise in traffic and prevent high packet losses.

PIQI-RCP also shows high link utilization in this scenario. Throughout the experiment, the input traffic rate is very close to the set virtual link capacity. Unlike



(a) XCP

(b) JetMax

(c) RCP

(d) PIQI-RCP

Fig. 35. Performance in the presence of background mice traffic in a dumbbell topology. Mice traffic is generated in the system at $t = 30$.

RCP, the input traffic rate does not significantly overshoot the link capacity. This is primarily because new flows entering the system start with a small sending rate that increases gradually upon receiving feedback from the router.

In contrast to other methods, JetMax behaves very differently in the presence of mice traffic. JetMax was primarily designed considering long flows that remain in the system for a longer period of time and have much data to transfer. They primarily govern the stability of the system. Another objective of JetMax is to provide fairness among all flows in the system while maintaining nearly zero queue level and packet loss rate. Link utilization may drop when there is a combination of long and short flows. The primary reasons are:

- In order to provide fairness to all flows in the system, they are all given equal fair rate as feedback. Short flows will not completely utilize the rate given to them by the router as they do not have much data to send.

- In order to provide almost zero queue level and packet loss, new flows entering the system propose a rate that gets approved by the routers only when existing flows have reduced their sending rate correspondingly. This approach works extremely well when the system has only long flows but has side effects in the presence of short flows. By the time existing flows have reduced their sending rate, new flows may exit the system or not completely utilize the approved rate due to lack of data to send.

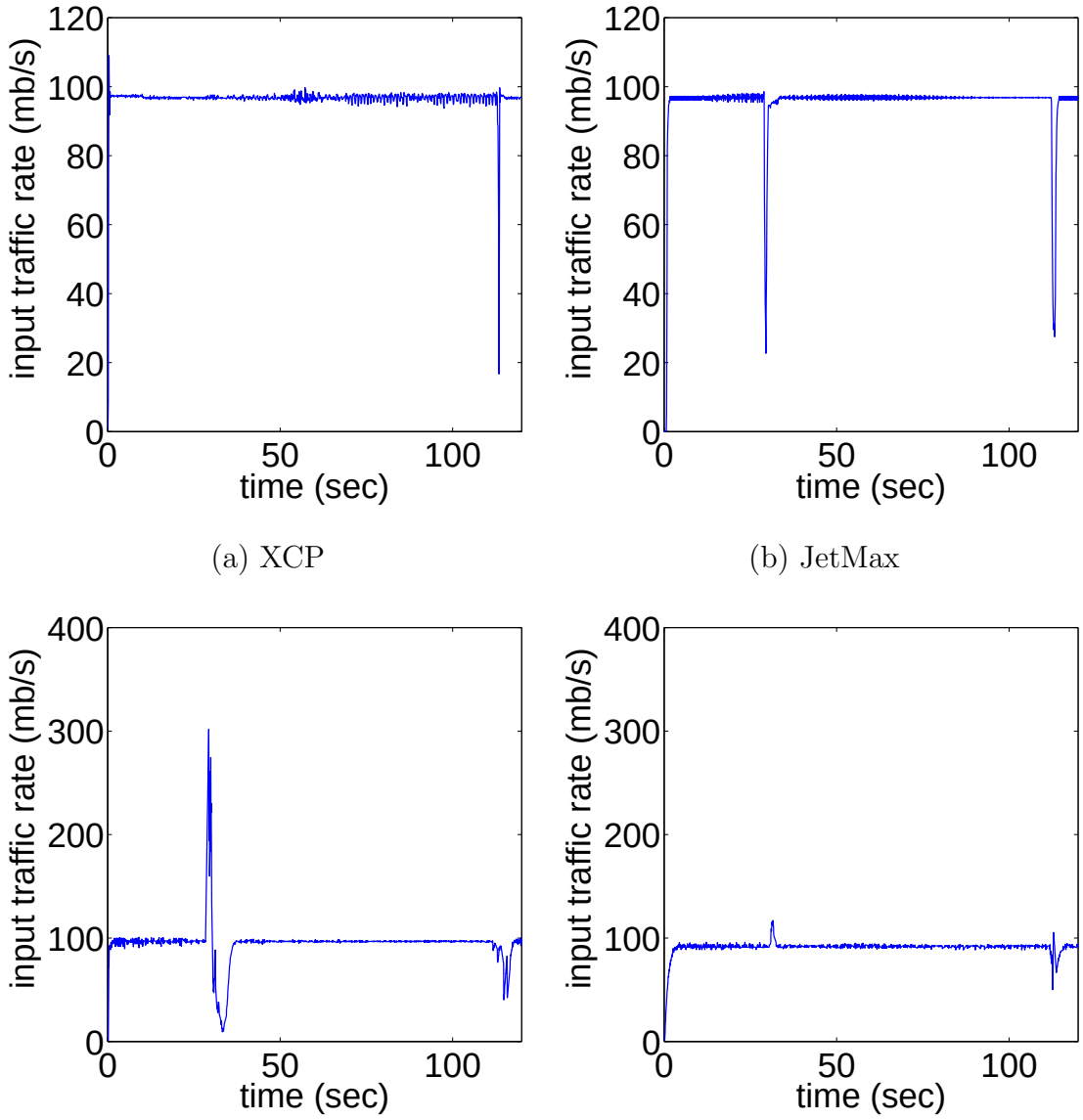
As seen in Fig. 35, until $t = 30$, JetMax operates at high utilization levels when there is only one long flow in the system. However, when mice traffic is started the router perceives that a lot of flows have entered the system, the reason being that the estimation of the number of flows N_l in the system is independent of the input traffic rate and depends only upon the sum of inter-packet transmission delay set in the

congestion header by the flows. In the presence of mice traffic, the system is unable to differentiate between short and long flows and so feeds back an equal rate to all of them. This causes the long flow to significantly decrease its sending rate, i.e., the long flow gets penalized in the presence of short flows. For the current setup, the link utilization fluctuates between $10 - 20\%$. The experiment is also repeated for $\tau = 1.0$ rather than the suggested value of $\tau = 0.6$ to observe similar behavior. However, the buffer size at the router remains zero with no packet drop throughout the experiment.

9. Abrupt Change in Traffic Demand

In this experiment, we examine the performance of the system with abrupt increase or decrease in traffic demand. Dumb-bell topology is used with two machines on one side of the bottleneck acting as sender and one machine on the other side acting as receiver. All the access links are 1 gb/s while the bottleneck link has capacity 100 mb/s and round-trip propagation delay 50 ms. At $t = 0$, one long flow is started for a duration of 120 seconds. At $t = 30$ another 10 flows abruptly enter the system. These 10 flows continue to remain in the system until $t = 113$, when they all suddenly exit. The performance in this scenario is shown in Fig. 36. The dynamics of the first flow, which lasts for 120 seconds, in the case of RCP and PIQI-RCP is shown in Fig. 37.

From the figures, it can be clearly inferred that XCP and JetMax are robust in the face of sudden increase or decrease in traffic demand. The link utilization may drop momentarily when a large number of flows join or leave the system simultaneously, but the system converges very fast to its steady-state. RCP performs the worst in this case. At time $t = 30$, the average input traffic rate overshoots to around 300 mb/s and the queue size jumps to around 11000 packets as shown in Fig. 38. This is because at this time, the router's control algorithm has the per flow rate computed to be 100 mb/s. All the incoming flows are given this rate. When flows start sending



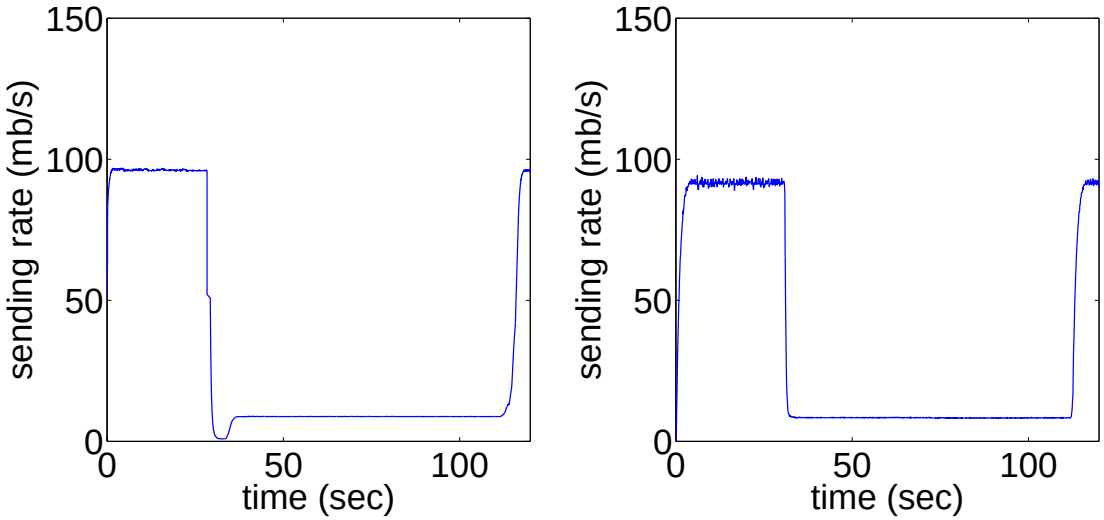
(a) XCP

(b) JetMax

(c) RCP

(d) PIQI-RCP

Fig. 36. Performance with abrupt change in traffic demand. One long flow starts at $t = 0$ and ends at $t = 120$. At $t = 30$, 10 flows join the system and leave at $t = 113$. The bottleneck link capacity is 100 mb/s. All flows have round-trip propagation delay of 50 ms.



(a) RCP (b) PIQI-RCP

Fig. 37. Dynamics of the first flow in the case of RCP and PIQI-RCP with sudden increase and decrease in traffic demand. One long flow starts at $t = 0$ and ends at $t = 120$. At $t = 30$, 10 flows join the system and leave at $t = 113$. The bottleneck link capacity is 100 mb/s. All flows have round-trip propagation delay of 50 ms.

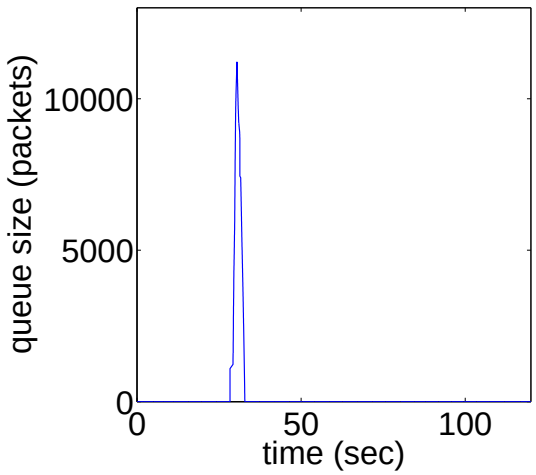


Fig. 38. Queuing dynamics at the router in the case of RCP with sudden increase in traffic demand at $t = 30$.

at this rate, the queue at the router significantly builds up. With the rise in both the input traffic rate and queue size, the router's control equation computes a very low rate. This rate, when assigned to the flows, make them drastically reduce their sending rate and hence cause the drop in link utilization. The system remains in this transient state (i.e., both overshoot and undershoot) for about 7 seconds (i.e., 140×50 ms = 140 RTTs) before reaching its steady-state. Also, when a large number of flows leave the system at $t = 113$, RCP takes nearly 3 seconds (i.e., 60 RTTs) to reach its steady state. In the case of PIQI-RCP, the overshoot in average input traffic rate at $t = 30$, as compared to RCP, is significantly lower. As can be seen from the figure, it increases to around 128 mb/s only momentarily and the system quickly converges to its steady-state. The excessive traffic, for a short period, gets absorbed in the network device queues without overshooting the router queue.

B. Summary of Results

Using the experimental results, the comparison of XCP, JetMax, RCP, and PIQI-RCP has been summarized below considering different performance metrics.

- *Link Utilization:* All the methods can provide high link utilization using long flows. We could not carry out our experiment using link capacities higher than 1 gb/s because of lack of availability. However, because of the improved design of these protocols, we are convinced that these methods can provide high utilization in higher capacity links. Even a single flow can saturate the entire link provided the end-host system does not act as bottleneck in pumping traffic into the network. Unlike other methods, JetMax drops link utilization in the presence of low rate mice traffic. Some suggested remedies to improve the performance of JetMax in such scenarios are listed below:

  - If routers can identify long and short flows through some means then short flows can be treated as unresponsive flows and given a basic rate while long flows are the only ones that are considered as possibly responsive and hence controlled.
  - If short flows can indicate their desired sending rate accurately then the router can feedback a rate to them that is the minimum of desired sending rate and the calculated fair rate. In case the desired sending rate is lower than the calculated fair rate, the flow can be considered unresponsive and existing long flows in the system would then be made to reduce their current sending rate by smaller amounts corresponding to the desired sending rate of the new flow.

  We leave the study of these two approaches as part of future work.

- *Buffering Requirement:* All the methods strive to have zero or low queue size in the steady-state. However, in the transient state XCP, RCP, and PIQI-RCP have higher buffering requirements than JetMax. In the case of JetMax, input traffic rate never exceeds the link capacity. New flows and existing flows in the system always have their sending rate approved by the bottleneck router. This prevents overshooting the queue even in the transient phase. XCP, being a window-based protocol, is bursty in data transmission and so higher queues are required at the router to absorb the traffic burst. RCP has the highest buffering requirement. PIQI-RCP has significantly lower buffering requirements compared to RCP due to a more responsive router controller and an improved controller at the end-hosts.

- *Impact of RTT:* XCP and RCP do not perform well in the case of highly het-

erogeneous RTTs. We confirmed this in ns-2 simulations but could not verify in our Linux experiments due to limitations in emulating large delays. However for small delays, as confirmed by our experiments, all the methods are scalable to round-trip propagation delays of end flows. The throughput of a flow is independent of its RTT. Flows with higher RTT also share the same rate as flows with smaller RTT.

- *Max-Min Fairness:* XCP, unlike the other methods, cannot achieve max-min fairness in all topologies. As shown in [34], in the case of XCP, the bottleneck link utilization is a function of the control parameter α, shuffling factor γ, fraction of unresponsive flows ρ, and fraction of unresponsive traffic σ. Max-min fairness and link utilization can be improved by selecting a very small value of γ but this would degrade the convergence time of the system. However, for a given α and γ it is always possible to tune ρ and σ to prevent flows at certain bottleneck links from attaining max-min fairness.

- *Abrupt Change in Traffic:* As seen in the experiments, XCP and JetMax are robust to abrupt increases or decreases in traffic demand. However, RCP does not fare well considering this metric. There are significant overshoots and undershoots in the input traffic rate and router queue size. The system also takes a considerable amount of time to converge to its steady-state when faced with such scenarios. Unlike RCP, PIQI-RCP limits the rise in input traffic rate and queue size since new flows entering the system start with a smaller sending rate allowing the router to converge to a new steady-state and existing flows in the system to simultaneously decrease their sending rate. Also, a higher integral gain at the router makes it to converge to its steady-state faster. Coupled with both these improvements, PIQI-RCP is more robust to abrupt changes in traffic

demand compared to RCP.

- *Congestion Header Size:* The congestion header size of XCP, JetMax, RCP, and PIQI-RCP in our implementation is 20, 32, 16, and 24 bytes respectively. RCP has the smallest congestion header size while JetMax has the largest. Without the **PROTOCOL**, **LENGTH**, and **VERSION** fields, the congestion header size of XCP, JetMax, RCP, and PIQI-RCP is 16, 29, 12, and 21 bytes. Considering unidirectional data flow, i.e., data packets flow only in one direction and the other direction has only acknowledgement packets, the congestion header size can be further reduced to 12, 20, 8, and 13 bytes. In all of our experiments, we have reported the achievable throughput at the IP layer using packet size 1500 bytes. Hence, the effective data throughput achievable at the application layer needs to be calculated by also considering the size of the congestion header and not just the TCP/IP headers.

- *Per-Packet Computations:* For facilitating congestion control, XCP does 6 additions and 3 multiplications, while RCP and PIQI-RCP do 2 additions and 2 multiplications per-packet. JetMax does 3 additions for a packet from a responsive flow and 2 additions for a packet from an unresponsive flow. Also, JetMax does not require any per-packet multiplication. In all, JetMax requires the least number of per-packet computations while XCP has the highest.

The above comparison shows that all the proposed methods have drawbacks. PIQI-RCP performs better that RCP considering most performance metrics and in almost all experimental setups.

CHAPTER VII

CONCLUSION AND FUTURE WORK

In this chapter, we summarize our work and the results we obtained. We also suggest several open problems that require further study.

A. Conclusion

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 two new controllers called *Queue Independent RCP* (QI-RCP) and *Proportional Integral Queue Independent RCP* (PIQI-RCP). We showed that their heterogeneous stability could be easily established in both continuous and discrete cases 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.

In the second part of the thesis, we experimentally evaluated the performance of XCP, JetMax, RCP, and PIQI-RCP using real systems and gigabit networks. We developed an implementation of these protocols on Linux platform. Our experiments highlight the strengths and weakness of each of these protocols. While XCP can sustain high link utilization and is also scalable with respect to high variance of round-trip times (RTTs), it cannot achieve max-min fairness in all topologies. Also, it does a much higher number of per-packet computations as compared to other methods. Though we could not confirm, we still think there may be scalability issues at routers with higher link capacities. JetMax can also achieve high link utilization in the case of long-lived flows. It requires the least amount of buffer size and per-packet computations inside the routers. However, its congestion header size is the largest

and link utilization suffers when the input traffic contains mice flows. In the case of RCP, the link utilization is high and independent of the RTTs of the end-hosts. The number of per-packet computations are lower as compared to XCP. However, it requires a much higher queue at the router to handle abrupt increases in traffic demand. PIQI-RCP retains most of the strengths of RCP and simultaneously reduces the buffering requirement at the routers while providing better convergence properties. This advantage comes at the cost of slightly higher average flow completion time.

B. Future Work

Explicit congestion control, being a new direction of research, has numbers of open issues. Apart from an excellent discussion in [10] (that is mainly for XCP but also applies to JetMax, RCP, and PIQI-RCP) on deployment related problems, there are other issues that we think are also important to consider.

Stability conditions in the case of XCP and RCP for heterogeneous delays are still not available and require further study. Also, multi-link stability analysis of max-min methods such as XCP, JetMax, RCP, and PIQI-RCP is an important open problem that requires attention.

Most of the existing protocols involve network delays in their control equation. For example, XCP requires computing average RTT in order to decide the instant when the router control algorithm should be invoked. It also requires RTT values while computing feedback information. RCP and PIQI-RCP also involve average RTT in their control equation. However, a number of operating systems have limits on the accuracy of timing and hence RTT estimation. The problem is much worse in the case of local area networks (LANs), where the RTT is so small that it may be difficult to accurately measure it. In such a scenario, it is necessary to study the

effects of incorrect RTT estimation and effect of flows with extremely small RTT on the behavior of the system.

JetMax is robust to inaccuracy in RTT estimation. However, it has a large congestion header size and a complicated algorithm for handling bottleneck switching. We feel that it may be possible to simplify JetMax by simply considering the fair rate rather than the probability of packet loss in deciding the bottleneck router. In the case of JetMax, the link utilization also suffers in presence of low rate mice traffic. We suggested possible solutions in chapter VI that require further study.

Our experimental work involved link capacities less than or equal to 1 gb/s due to the lack of availability of higher capacity links. Another effort would be to carry out experiments using our implementation in link capacities of tens or hundreds of gigabits per second and study the scalability of these protocols as regard to implementation.

All the methods such as XCP, JetMax, RCP, and PIQI-RCP analyze flow control in their discussion without considering packet loss control/recovery in detail. Most of these methods resort to TCP-based loss detection and recovery, i.e., reducing the congestion window or sending rate by half and restoring the window or rate based on feedback from routers after recovering from losses. This needs to be analyzed further as how it would affect the flow and stability properties of the system.

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] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin, "REM: Active Queue Management," *IEEE Networks*, vol. 15, no. 3, pp. 48–53, Jun. 2001.

[3] S. Bhandarkar, S. Jain, and A. L. N. Reddy, "LTCP: Improving the Performance of TCP in Highspeed Networks." *Computer Communication Review*, vol. 36, no. 1, pp. 41–50, 2006.

[4] D.-M. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," *Computer Networks and ISDN Systems*, vol. 17, no. 1, pp. 1–14, Jun. 1989.

[5] C. A. Desoer and Y. T. Wang, "On the Generalized Nyquist Stability Criterion," *IEEE Trans. Automat. Contr.*, vol. 25, no. 2, pp. 187–196, Apr. 1980.

[6] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown, "Processor Sharing Flows in the Internet," in *Proc. IEEE IWQoS*, Jun. 2005. [Online]. Available: http://yuba.stanford.edu/rcp/RCP-IWQoS.pdf

[7] N. Dukkipati and N. McKeown, "Processor Sharing Flows in the Internet," High Performance Networking Group, Stanford Univ., Tech. Rep. TR04-HPNG-061604, Jun. 2004. [Online]. Available: http://yuba.stanford.edu/rcp/RCP-TR.pdf

[8] N. Dukkipati, N. McKeown, and A. G. Fraser, "RCP-AC: Congestion Control to make flows complete quickly in any environment," in *Proc. High-Speed Networking Workshop: The Terabits Challenge, IEEE INFOCOM*, Apr. 2006. [Online]. Available: http://yuba.stanford.edu/rcp/RCP_AC-dukkipati.pdf

[9] Emulab, (2005). [Online]. Available: http://www.emulab.net/

[10] A. Falk, Y. Pryadkin, and D. Katabi, "Specification for the Explicit Control Protocol (XCP)," Oct. 2005, IETF Internet-draft. [Online]. Available: http://www.isi.edu/isi-xcp/docs/draft-falk-xcp-spec-01.txt

[11] FEDORA CORE, (2005). [Online]. Available: http://fedora.redhat.com/

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

[13] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management," ICIR, Tech. Rep., Aug. 2001. [Online]. Available: http://www.icir.org/floyd/papers/adaptiveRed.pdf

[14] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[15] I. Foster and R. L. Grossman, "Data Integration in a Bandwidth-Rich World," *Commun. ACM*, vol. 46, no. 11, pp. 50–57, 2003.

[16] FreeBSD, (2005). [Online]. Available: http://www.freebsd.org/

[17] T. Herbert, *The Linux TCP/IP Stack: Networking for Embedded Systems*. Boston, MA: Charles River Media, 2004.

[18] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "Analysis and Design of Controllers for AQM Routers Supporting TCP Flows," *IEEE Trans. Automat. Contr.*, vol. 47, no. 6, pp. 945–959, Jun. 2002.

[19] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 1726–1734.

[20] IANA, "Internet Assigned Numbers Authority," (2005). [Online]. Available: http://www.iana.org/

[21] JetMax@TAMU, (2005). [Online]. Available: http://irl.cs.tamu.edu/projects/mkc/

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

[23] C. Jin, D. X. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, and S. Singh, "FAST TCP: From Theory to Experiments," *IEEE Network*, vol. 19, no. 1, pp. 4–11, Jan. 2005.

[24] R. Johari and D. K. H. Tan, "End-to-End Congestion Control for the Internet: Delays and Stability," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 818–832, Dec. 2001.

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

[26] F. P. Kelly, "Charging and Rate Control for Elastic Traffic," *Euro. Trans. on Telecommun.*, vol. 8, no. 1, pp. 33–37, Jan. 1997.

[27] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability," *J. of Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, Mar. 1998.

[28] T. Kelly, "Scalable TCP: Improving Performance in High-speed Wide Area Networks," in *Proc. PFLDnet*, Feb. 2003.

[29] S. Kunniyur and R. Srikant, "Stable, Scalable, Fair Congestion Control and AQM Schemes That Achieve High Utilization in the Internet," *IEEE Trans. Automat. Contr.*, vol. 48, no. 11, pp. 2024–2029, Nov. 2003.

[30] S. Kunniyur and R. Srikant, "Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management," in *Proc. ACM SIGCOMM*, Aug. 2001, pp. 123–134.

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

[32] Y. Li, D. Leith, and R. N. Shorten, "Experimental Evaluation of TCP Protocols for High-Speed Networks," (2007). [Online]. Available: http://www.hamilton.ie/net/eval/results_HI2005.pdf

[33] Linux, (2005). [Online]. Available: http://www.linux.org/

[34] 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.

[35] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. Doyle, "Dynamics of TCP/RED and a Scalable Control," in *Proc. IEEE INFOCOM*, Jun. 2002, pp. 239–248.

[36] P. Marbach, "Priority Service and Max-Min Fairness," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 733–746, Oct. 2003.

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

[38] MATLAB, (2005). [Online]. Available: http://www.mathworks.com/

[39] J. Nagle, "Congestion Control in IP/TCP Internetworks," *IETF RFC 896*, Jan. 1984.

[40] I. J. Nagrath and M. Gopal, *Control Systems Engineering.* New York: John Wiley & Sons, 2004.

[41] Netfilter, (2005). [Online]. Available: http://www.netfilter.org/

[42] Ns-2, "Network Simulator," (2005). [Online]. Available: http://www.isi.edu/nsnam/ns/

[43] NSF FIND, "Future Internet Design," (2006). [Online]. Available: http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=12765&org=CNS

[44] NSF GENI, "Global Environment for Network Innovations," (2006). [Online]. Available: http://www.nsf.gov/cise/cns/geni/

[45] K. K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," *IETF RFC 3168*, Sep. 2001.

[46] H. Sivakumar, R. L. Grossman, M. Mazzucco, Y. Pan, , and Q. Zhang, "Simple Available Bandwidth Utilization Library for High-Speed Wide Area Networks,"

(2003). [Online]. Available: http://www.dataspaceweb.net/papers/sabul-jsc-03.pdf

[47] M. Suchara, R. Witt, and B. Wydrowski, "TCP MaxNet–Implementation and Experiments on the WAN in Lab," in *Proc. IEEE ICON*, Nov. 2005, pp. 901–906.

[48] TeraPaths, "A QoS Enabled Collaborative Data Sharing Infrastructure for Peta-scale Computing Research," (2006). [Online]. Available: http://www.atlasgrid.bnl.gov/terapaths/

[49] Y. Tian and H. Yang, "Stability of the Internet Congestion Control with Diverse Delays," *Automatica*, vol. 40, no. 9, pp. 1533–1541, 2004.

[50] G. Vinnicombe, "On the Stability of End-to-End Congestion Control for the Internet," Cambridge University, Tech. Rep. CUED/F-INFENG/TR.398, Dec. 2000.

[51] 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.

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

[53] 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.

[54] Y. Zhang and T. Henderson, "An Implementation and Experimental Study of the eXplicit Control Protocol (XCP)," in *Proc. IEEE INFOCOM*, Mar. 2005, pp. 1037–1048.

[55] 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.

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

VITA

Saurabh Jain received his Bachelor of Technology (B.Tech.) in electronics and communication engineering from Indian Institute of Technology, Roorkee, India, in May 2003. He then worked for a year at Oracle India Pvt. Ltd, Hyderabad, India. He began pursuing his Master of Science degree in electrical engineering at Texas A&M University in August 2004 and received his degree in May 2007.

He joined Internet Research Lab, Department of Computer Science, Texas A&M University in 2005 and his research interests include congestion control in the Internet, performance analysis of computing and networking systems. He may be contacted at:

Saurabh Jain
30 Mahabir Bhawan
A.T. Road
Guwahati-781001, Assam
India

The typist for this thesis was Saurabh Jain.