This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2011 proceedings

Rate based feedback: some experimental evaluation with NetFPGA

Nizar Malangadan and Gaurav Raina
India-UK Advanced Technology Center of Excellence in Next Generation Networks
Department of Electrical Engineering
Indian Institute of Technology Madras, Chennai 600 036, India
Email: nizar.mpn@gmail.com, gaurav@ee.iitm.ac.in

***Abstract*—Transport protocols that use explicit rate feedback are a promising alternative to the protocols which use implicit, packet loss based, feedback. A key issue in the design of rate controlled communication networks is how new flows may be admitted at a fair and high starting rate.**

In this paper, for the management of new flows we present a modified NetFPGA Rate Control Protocol (RCP) implementation for a recently proposed admission management process at the routers. Using this implementation, we investigate the admission process of new flows in rate controlled networks. We conduct experimental tests across a range of bandwidth-delay product environments, and also deal with the arrival of a large number of flows.

Our work serves to exhibit that the proposed process appears to be attractive for dealing with the admission of new flows while maintaining small queues. Additionally, the routers are able to maintain a specified target link utilisation. A consequence of being able to maintain small queues and a desired link utilisation is that buffers, in routers, may be dimensioned to be much smaller than the current, bandwidth-delay product, buffer sizing rule.

I. Introduction

There is growing interest in the design of new transport protocols as it is widely accepted that the currently deployed protocols are beginning to show signs of strain. The family of protocols that use explicit rate feedback provide a promising direction in the quest for new, faster and fairer protocols; for example, see [1], [4], [7], [8], [15] and references therein.

An issue that is central to all transport protocols is the choice of buffer size in routers. In particular, the currently used bandwidth-delay product rule has been questioned, and shown not to be scalable for next generation, high-speed, communication networks; for example see [5], [14], [16], and references therein for work associated with TCP like networks.

For some early work on the issue of sizing router buffers in rate controlled networks see [8], [9], [15]. In such networks, a key design issue is how to admit new flows at a fair and high starting rate. To address this issue, [8] has proposed an admission management procedure.

The NetFPGA is an open platform that enables researchers to build high-speed, hardware-accelerated networking systems [10], [12]. NetFPGA is appealing due to its flexibility in implementing hardware at line speed, and its ability to closely emulate a real networking environment. NetFPGA is beginning to play a role in networking research; for example see [2], [3], [6].

The focus, and contribution, in this paper is an implementation and evaluation of an admission management procedure on a modified NetFPGA RCP platform. We first modified the existing Linux implementation of RCP end-systems. Then we extended the current NetFPGA RCP router implementation with the addition of a module required for the admission management of new flows in rate controlled networks. We use the aforementioned implementations to conduct experiments using a proportionally fair rate control protocol.

The rest of the paper is organised as follows. In section II, we briefly describe the proportionally fair rate control protocol. In section III, we outline an admission management process. In section IV, we explain the modified RCP implementation in end-systems, and the modified NetFPGA RCP router implementation. In section V, we discuss the NetFPGA experimental results. In section VI, we provide an outline of our contributions and in section VII, we provide avenues that merit further research.

II. Proportionally fair rate control protocol

In this section, we recapitulate the proportionally fair RCP introduced in [8]. Consider a network with a set J of resources and a set R of routes. A route r is identified with a non-empty subset of J, and we write $j \in r$ to indicate that route r passes through resource j. For each j, r such that $j \in r$, let T_{rj} be the propagation delay from the source of flow on route r to the resource j, and let T_{jr} be the return delay from resource j to the source. Then

$$T_{rj} + T_{jr} = T_r \qquad j \in r, r \in R, \tag{1}$$

where T_r is the round-trip propagation delay on route r: the identity (1) is a direct consequence of the end-to-end nature of the signalling mechanism, whereby congestion on a route is conveyed via a field in the packets to the destination, which then informs the source. We assume that the queueing delays form a negligible component of the end-to-end delay; this is consistent with the assumption of a network operating with small queues.

A proportionally fair RCP can be modelled by the following system of differential equations

$$\frac{d}{dt} R_j(t) = \frac{a R_j(t)}{C_j \overline{T}_j(t)} \left(C_j - y_j(t) - b_j C_j p_j(y_j(t)) \right) \tag{2}$$

978-1-61284-233-2/11/$26.00 ©2011 IEEE

where

$$y_j(t) = \sum_{r:j\in r} x_r(t - T_{rj}) \tag{3}$$

is the aggregate load at link j, $p_j(y_j)$ is the mean queue size at link j when the load there is y_j, and

$$\overline{T}_j(t) = \frac{\sum_{r:j\in r} x_r(t)T_r}{\sum_{r:j\in r} x_r(t)} \tag{4}$$

is the average round-trip time of *packets* passing through resource j. The parameter a controls the speed of convergence at each resource, the parameter b_j controls the utilisation at resource j at equilibrium, and C_j is the link capacity. We suppose that the flow rate $x_r(t)$ leaving the source of route r at time t is given by

$$x_r(t) = w_r \left(\sum_{j\in r} R_j(t - T_{jr})^{-1} \right)^{-1}. \tag{5}$$

We interpret these equations as follows. Resource j updates $R_j(t)$, the nominal rate of a flow which passes through resource j alone, according to equation (2). In this equation the term $C_j - y_j(t)$ represents a measure of the rate mismatch, at time t, at resource j, while the term $b_j C_j p_j(y_j(t))$ is proportional to the mean queue size at resource j. Equation (5) gives the flow rate on route r as the product of the weight w_r and reciprocal of the sum of the reciprocals of the nominal rates at each of the resources on route r. Note that equations (3) and (5) make proper allowance for the propagation delays, and the average round-trip time (4) of packets passing through resource j scales the rate of adaptation (2) at resource j.

The computation (5) can be performed as follows. If a packet is served by link j at time t, $R_j(t)^{-1}$ is added to the field in the packet containing the indication of congestion. When an acknowledgement is returned to its source, the acknowledgement feedbacks the sum, and the source sets its flow rate equal to the returning feedback to the power of -1.

A simple approximation for the mean queue size is as follows. Suppose that the workload arriving at resource j over a time period τ is Gaussian, with mean $y_j\tau$ and variance $y_j\tau\sigma_j^2$. Then the workload present at the queue is a reflected Brownian motion, with mean under its stationary distribution of

$$p_j(y_j) = \frac{y_j\sigma_j^2}{2(C_j - y_j)}. \tag{6}$$

The parameter σ_j^2 represents the variability of resource j's traffic at a packet level. Its units depend on how the queue size is measured: for example, packets if packets are of constant size, or Kilobits otherwise.

At the equilibrium point $y = (y_j, j \in J)$ for the dynamical system (2-6) we have

$$C_j - y_j = b_j C_j p_j(y_j). \tag{7}$$

From equations (6-7) it follows that at the equilibrium point

$$p_j'(y_j) = \frac{1}{b_j y_j}. \tag{8}$$

Observe that in the above model formulation there are two forms of feedback: rate mismatch and queue size. In this paper, we focus on the case where there is feedback based only on rate mismatch.

A. Local stability with feedback based only on rate mismatch.

If the parameters b_j are all set to zero, then the protocol uses a target, or virtual, capacity of say 90% of the actual capacity, then this will achieve an equilibrium utilisation of 90%. In this case the sufficient condition for local stability is

$$a < \frac{\pi}{2}. \tag{9}$$

For the requisite derivations and associated analysis see [8].

III. Admission management

In rate controlled networks when a new flow arrives, it expects to learn, after one round-trip time, of its starting rate. An important aspect in the design of such networks is the management of new flows; in particular, a key question is the scale of the step-change in rate that is necessary at a resource to accommodate a new flow.

A. Step-change algorithm

In equilibrium, the aggregate flow through resource j is y_j, the unique value such that the right hand side of (2) is zero. When a new flow, r, begins transmitting, if $j \in r$, this will disrupt the equilibrium by increasing y_j to $y_j + x_r$. Thus, in order to maintain equilibrium, whenever a flow, r, begins R_j needs to be decreased, for all j with $j \in r$.

According to (3)

$$y_j = \sum_{r:j\in r} w_r \left(\sum_{k\in r} R_k^{-1} \right)^{-1}$$

and so the sensitivity of y_j to changes in the rate R_j is readily deduced to be

$$\frac{\partial y_j}{\partial R_j} = \frac{y_j \overline{x}_j}{R_j^2} \tag{10}$$

where

$$\overline{x}_j = \frac{\sum_{r:j\in r} x_r \left(\sum_{k\in r} R_k^{-1}\right)^{-1}}{\sum_{r:j\in r} x_r}.$$

This $\overline{x}_j$ is the average, over all packets passing through resource j, of the unweighted fair share on the route of a packet.

Suppose now that when a new flow r, of weight w_r, arrives, it sends a request packet through each resource j on its route, and suppose each resource j, on observation of this packet, immediately makes a step-change in R_j to a new value

$$R_j^{new} = R_j \cdot \frac{y_j}{y_j + w_r R_j}. \tag{11}$$

The purpose of the reduction is to make room at the resource for the new flow. Although a step-change in R_j will take time to work through the network, the scale of the change

anticipated in traffic from existing flows can be estimated from (10) and (11) as

$$(R_j - R_j^{new}) \cdot \frac{\partial y_j}{\partial R_j} = w_r \overline{x}_j \cdot \frac{y_j}{y_j + w_r R_j}.$$

Thus the reduction aimed for from existing flows is of the right scale to allow one extra flow at the average of the w_r-weighted fair share through resource j. Note that this is achieved without knowledge at the resource of the individual flow rates through it, $(x_r, r : j \in r)$: only knowledge of their equilibrium aggregate y_j is used in expression (11), and y_j may be determined from the parameters C_j and b_j as in (7).

IV. NetFPGA Implementation

In this section, we outline the NetFPGA implementation details for both the RCP end-systems as well as the RCP router.

A. Implementation of RCP in the end-systems

In the previous implementation of RCP [3] the congestion window size is set to the flow rate times the round-trip time of the flow, which acts to limits the rate of the flow. Controlling the rate in this manner sends packets in bursts, instead of being paced. Pacing packets do have desirable network properties; see [8]. To that end, a key aspect of our implementation is that we provide a mechanism for pacing packets. Thus we need to control the number of packets that are sent over a small time interval, called a jiffy (few milliseconds) [11]. The choice of the time period is limited by the implementation, and a more accurate timer would allow more fine grained pacing to be done.

We enumerate the steps required to control the number of packets sent per jiffy, according to the feedback rate.

1) At the start of the timer, the value of the rate limiting variable is set to the maximum number of packets that can be sent per jiffy. Since floating point operations are not normally allowed in the Linux kernel, we set the value of this variable to 1000 times the number of packets sent.
2) Each time a packet is sent out, the value in the variable is reduced by 1000. When the value becomes less than 1000, packets are not sent.
3) The timer is again restarted at the end of one jiffy, and the process described in 1 and 2 above is repeated. At this time, the fractional number of packets that remains from the earlier time period is carried over to the next time interval.
4) When the number of packets to be sent per jiffy is less than one, the period of the timer is increased so that at least one packet is sent within the timer interval. The value in the variable is multiplied by the same amount as in the increased time interval. Then the process as described in steps $1-3$ is repeated, except that the time interval is now more than one jiffy, and the variable stores the number of packets sent within this time period.

A requirement for the step-change algorithm is that the new flow request packet be indicated by a field in the RCP header. The RCP router, on seeing the field set with a request, will execute the step-change algorithm. RCP headers have place for SYN which occupies 1 bit. In our current RCP implementation, we set the SYN flag to 1 to indicate the start of a RCP flow.

B. RCP router implementation in NetFPGA

Currently, there exists a NetFPGA RCP router implementation [3]. This RCP router implementation was extended by adding the step-change algorithm. A new module called step-change was created, which upon the arrival of a new RCP flow computes the new equilibrium fair rate. It takes as input the equilibrium aggregate traffic (y_r), the weight of the flow (ω_r), and current fair rate (R_j), and provides as output the new equilibrium fair rate (R_j^{new}). Then the fair rate of the outgoing link is updated to the new equilibrium rate, and this value is stamped onto the new flow request packet before it is sent out to the corresponding outgoing link. The rest of the RCP functionality is as in the previous RCP implementation [3]. The rate update equation of the proportionally fair RCP is run on the software control layer of NetFPGA.

V. Experiments

For our experiments, we created a setup of two ingress links, one router and one egress link, each link of bandwidth 1 Gbps; see Figure 1 for a representation of the experimental set up. The sender and receiver hosts run the modified RCP enabled Linux 2.6.18 distribution. The modified NetFPGA RCP router implementation was used as the bottleneck link. We used the traffic simulator *Iperf* to generate network traffic, and to introduce time delays we used the Linux network emulation tool *netem*. The RCP parameter a was set to 0.1.

We conducted experiments with two sets of target utilisation values: 90% and 70%. Our focus was on the admission of new flows, over a range of different round-trip times. The round-trip times used were $10ms$ and $100ms$. The scenarios considered were the following: with a 100 flows, in equilibrium, we exhibit the step-change algorithm, when there is a sudden request for the admission of another $10, 50$, and 100 flows. In each of these cases, we find that the queue sizes remain small and the target utilisation is maintained. For the experimental results with a target utilisation of 90%, see Figure 2, and for the results with a target utilisation of 70%, see Figure 3.

Our experiments highlight the effectiveness of the step-change algorithm, as an integral part of the admission management procedure.

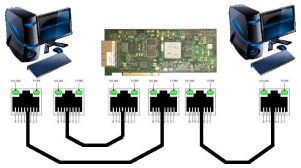
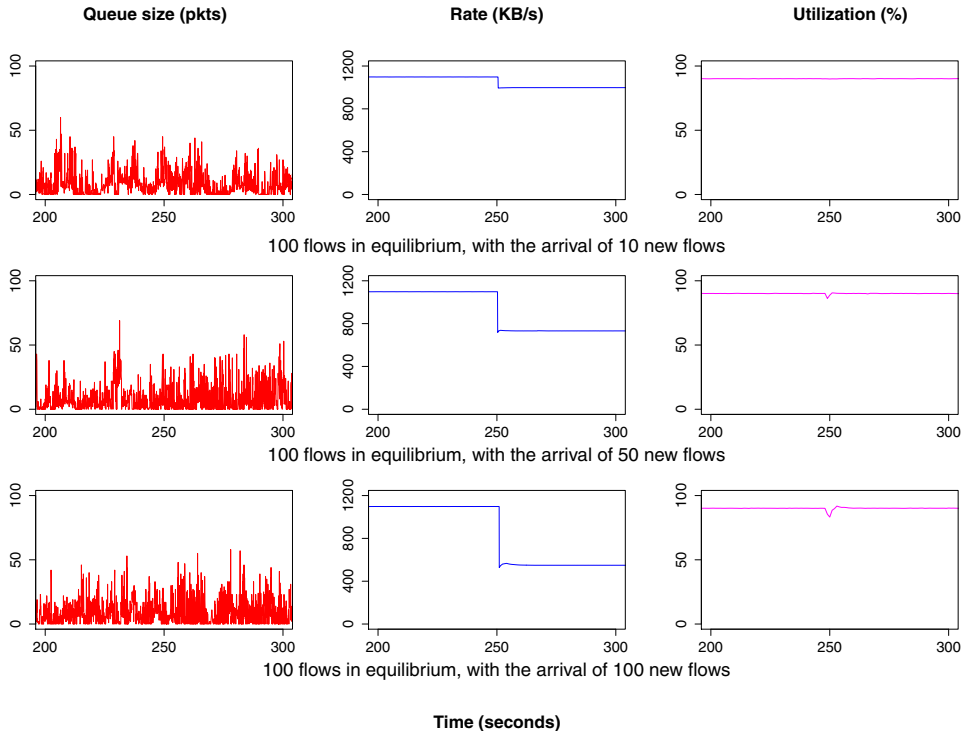
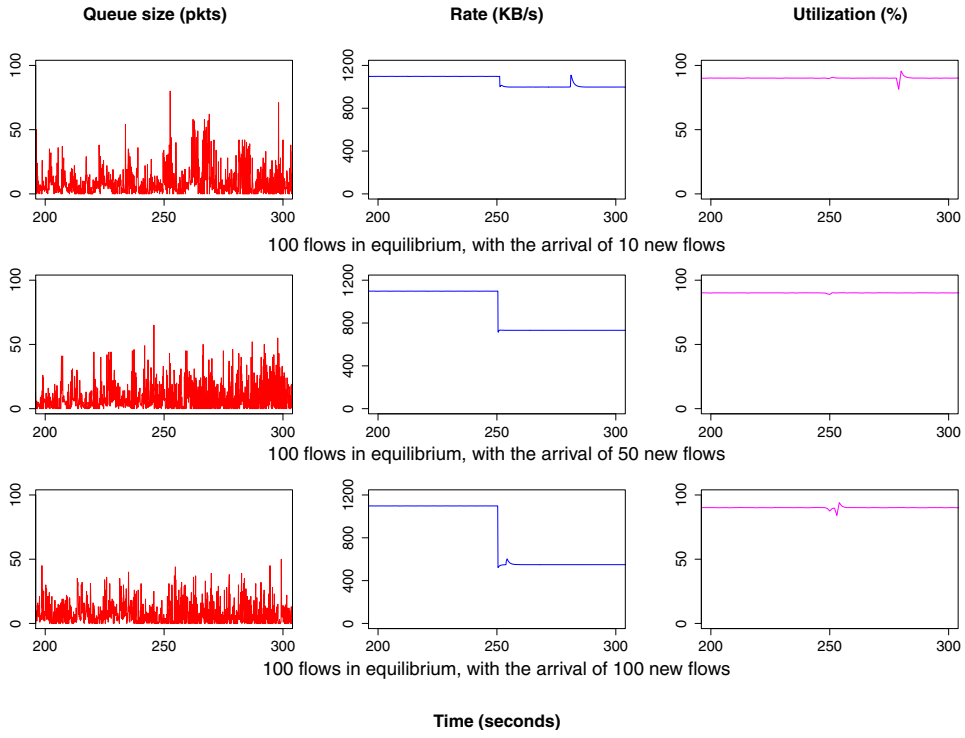


Fig. 1. Experimental setup for NetFPGA tests

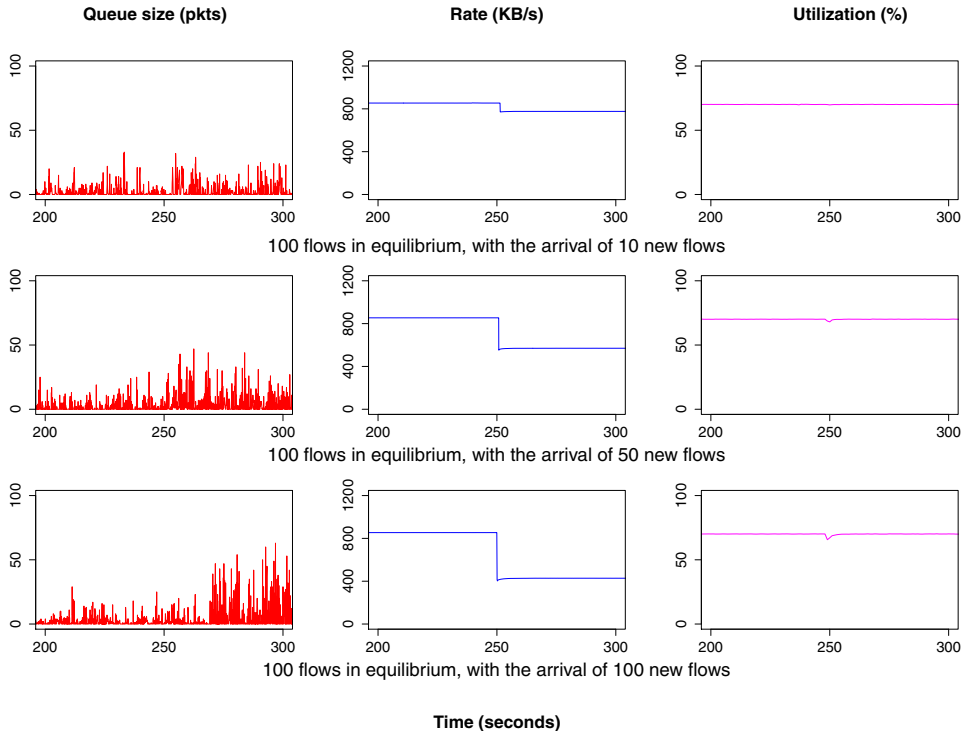


a) Round-trip time = 10 milliseconds

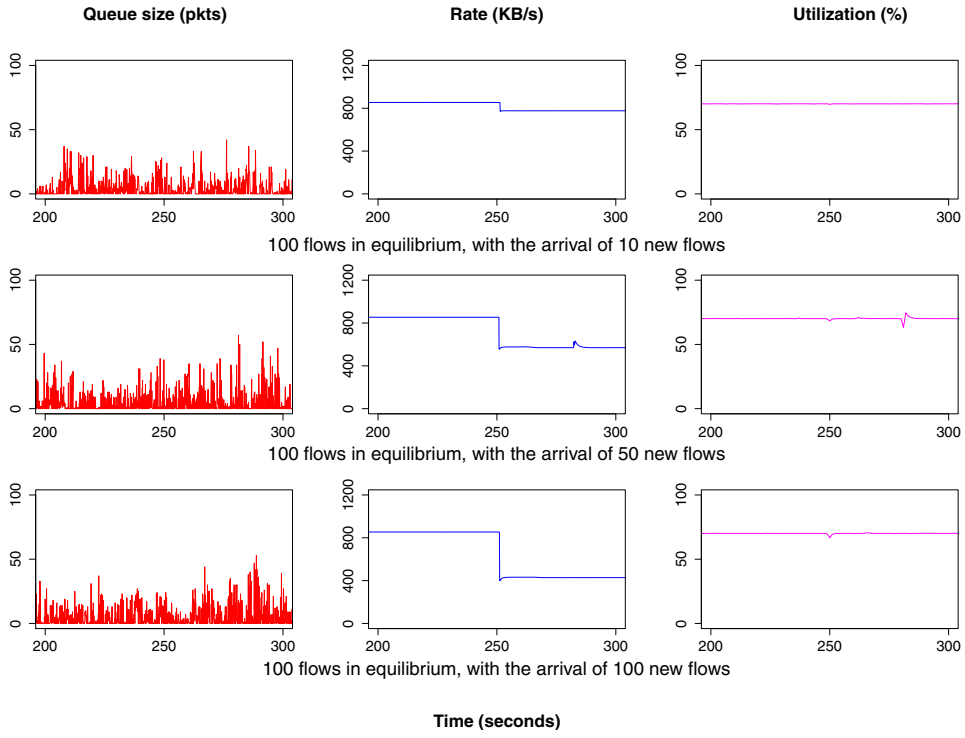


b) Round-trip time = 100 milliseconds

Fig. 2. Illustration of the step-change algorithm with the following parameters: $C = 1$Gbps, $a = 0.1$, and a target utilisation of 90%.



a) Round-trip time = 10 milliseconds



b) Round-trip time = 100 milliseconds

Fig. 3. Illustration of the step-change algorithm with the following parameters: $C = 1$Gbps, $a = 0.1$, and a target utilisation of 70%.

A. Discussion

Recall that the current buffer dimensioning rules have a bandwidth-delay product scaling [16]; this rule of thumb has a direct impact on end-to-end performance [13], [14] and on the energy consumption of routers [16].

Our experiments highlight the ability of rate controlled networks to maintain very small queues, with no packet loss, at a desired link utilisation. This clearly reduces latency at the routers, and hence contributes directly towards end-to-end performance. Thus, we could be in a position to vastly reduce the buffer requirements without compromising end-to-end performance which in turn would also help reduce the energy consumption in routers.

The experiments were conducted with only rate mismatch as the feedback mechanism; however, the admission management process is also applicable with both forms of feedback, i.e. the queue size and rate mismatch.

VI. Contributions

In the design and evaluation of new transport protocols, it is important to study new proposals both in theory and using experiments. Previous theoretical contributions have highlighted that explicit rate controlled protocols are an appealing alternative to the current implicit, loss based protocols. One key design issue in such rate controlled networks is the management of new flows; in particular, how to admit new flows at a fair and high starting rate.

Our contributions, in this paper, are the following. We modified an existing implementation of RCP end-systems. We also implemented a new admission management module on the NetFPGA RCP router. Further, we conducted experiments with the aforementioned implementations to demonstrate the effectiveness of the admission management procedure in handling the arrival of new RCP flows.

Effectively, we demonstrated that it is possible to maintain small queues, and have a target link utilisation, while admitting new flows in rate controlled networks. As a consequence of being able to maintain small queue sizes, we would be able to dimension router buffers to be much smaller than todays widely deployed buffer dimensioning rules.

VII. Avenues for further research

A natural avenue for further investigation would be to consider more complex topologies for our experiments. We would also like to test the protocols over a more dynamic environment with the random arrival and departure of RCP flows. An important question, which still remains open, is the requirement, and impact, of two forms of feedback, i.e. the queue size and rate mismatch, on network performance.

Acknowledgments

This work was carried out under the IU-ATC project funded by Department of Science and Technology (DST), Government of India and the UK EPSRC Digital Economy Programme. The authors would like to acknowledge Professor V. Kamakoti for extending the facilities of the Reconfigurable Intelligent Systems Engineering Lab (RISE) for conducting the NetFPGA experiments. The authors are grateful for valuable suggestions by Shankar Raman, Praveen Raja, and Ganesh Patil.

References

[1] H. Balakrishnan, N. Dukkipati, N. McKeown, and C. J. Tomlin, "Stability analysis of explicit congestion control protocols," *IEEE Communications Letters*, vol. 11, no. 10, pp. 823–825, 2007.

[2] N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, and G. Salmon, "Experimental study of router buffer sizing," *Proceedings of the eighth ACM SIGCOMM conference on Internet measurement*, 2008.

[3] N. Dukkipati, G. Gibb, N. McKeown, and J. Zhu, "Building a RCP (rate control protocol) test network," *Proceedings of the fifteenth Annual IEEE Symposium on High-Performance Interconnects*, 2007.

[4] N. Dukkipati, N. McKeown, and A. G. Fraser, "RCP-AC: Congestion control to make flows complete quickly in any environment," *Proceedings of the twenty-fifth IEEE International Conference on Computer Communications*, 2006.

[5] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Part III: Routers with very small buffers," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 3, pp. 83–90, 2005.

[6] S. Jain and G. Raina, "An experimental evaluation of CUBIC TCP in a small buffer regime," *Proceedings of the seventeenth National Conference on Communications (NCC)*, Bangalore, 2011.

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

[8] F. Kelly, G. Raina, and T. Voice, "Stability and fairness of explicit congestion control with small buffers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 51–62, 2008.

[9] A. Lakshmikantha, R. Srikant, N. Dukkipati, and N. McKeown, and C. Beck, "Buffer sizing results for RCP congestion control under connection arrivals and departures," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 5–15, 2008.

[10] J. W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo, "NetFPGA–An Open Platform for Gigabit-Rate Network Switching and Routing," *Proceedings of the IEEE International Conference on Microelectronic Systems Education*, 2007.

[11] R. Love, *Linux Kernel Development Second Edition*, Massachusetts: Novel Press, 2005.

[12] J. Naous, G. Gibb, S. Bolouki, and N. McKeown, "NetFPGA: reusable router architecture for experimental research," *Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, 2008.

[13] G. Raina and D. Wischik, "Buffer sizes for large multiplexers: TCP queueing theory and instability analysis," *Proceedings of EuroNGI Conference on Next Generation Internet*, 2005.

[14] G. Raina, D. Towsley, and D. Wischik, "Part II: Control theory for buffer sizing," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 3, pp. 79–82, 2005.

[15] T. Voice and G. Raina, "Stability analysis of a max-min fair Rate Control Protocol (RCP) in a small buffer regime," *IEEE Transactions on Automatic Control*, vol. 54, no. 8, pp. 1908–1913, 2009.

[16] D. Wischik and N. Mckeown, "Part I: buffer sizes for core routers," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 3, pp. 75–78, 2005.