MCP: Few Bits for Fairing and Small Queues in the Stable State

Maxim Podlesny and Sergey Gorinsky

Applied Research Laboratory
Department of Computer Science and Engineering, Washington University in St. Louis
One Brookings Drive, St. Louis, MO 63130-4899, USA
{*podlesny,gorinsky*}*@arl.wustl.edu*

***Abstract*— Interactive and other delay-sensitive applications are interested in keeping end-to-end delays of their packets minimal. Unfortunately, congestion control offered by Transmission Control Protocol (TCP) and other existing protocols inflates the end-to-end delays by building up queues at bottleneck links. In this paper, we investigate Multimodal Control Protocol (MCP) designed to maintain low queues after converging to the stable state where MCP flows utilize shared bottleneck links efficiently and fairly. To achieve this goal, MCP incorporates multiple modes of operation and allocates few bits in each packet header for explicit communication between hosts and routers. An innovative aspect of the explicit communication mechanism is an ability of a flow to urge all flows on its bottleneck links to switch temporarily into a fairing mode and thereby improve fairness of the bottleneck sharing. To make the fair sharing independent of round-trip times and packet sizes, MCP uses the sending bitrate as a control parameter and employs uniform adjustment timing for all flows. Our evaluation of MCP demonstrates its efficient fair operation and significantly shorter stable-state queues than under existing congestion control protocols.**

I. Introduction

Queuing delay experienced by packets inside the network is of significant importance for some applications. For example, human perception of an interactive multimedia application might degrade dramatically after round-trip time (RTT) exceeds few hundred milliseconds. In traditional Internet congestion control exemplified by Transmission Control Protocol (TCP) [1], transmission increases until the bottleneck link buffer saturates, causing the router to discard a packet. Since conventional routers employ First-In First-Out (FIFO) discipline for their link scheduling, such TCP-like probing for the available network capacity builds up long queues at shared bottleneck link buffers and hampers performance of delay-sensitive applications. In this paper, we explore how congestion control can keep link queues short. While this problem has a lot of related work, our investigation centers around innovative features of Multimodal Control Protocol (MCP) [2].

Support of low queuing together with other congestion control objectives (e.g., high utilization of bottleneck links) is a challenging problem. A common solution is to operate in multiple modes, with different modes dedicated to fulfilling different goals. MCP – as its name manifests – also follows this paradigm. In particular, MCP employs a fairing mode dedicated to improving fairness of bottleneck link sharing. Other modes enable convergence to efficient utilization of the network capacity. After MCP achieves efficient fair sharing of a bottleneck link, all competing flows switch to a stable mode where they transmit at constant rates. The constant-rate transmission in the stable mode is a noteworthy MCP innovation. Since transmission at constant rates is the smoothest, MCP excels in keeping link queues short in the steady state. To make the stable transmission rates independent of RTT and packet sizes, MCP uses the transmission rate as a control parameter and prescribes uniform timing for rate adjustments in all flows.

To treat a newly started flow fairly, MCP incorporates another innovative mechanism which relies on explicit communication between hosts and routers. Whenever the sender of a flow f sets special bits in sent packet headers, the explicit communication mechanism notifies all flows on bottleneck links of f to switch temporarily into the fairing mode, allowing flow f to acquire its fair rate.

MCP is targeted for networks where time between flow arrivals to a bottleneck link is longer than transition to the steady state. Whereas we do not expect MCP to improve performance in networks with congestion in the core, Odlyzko indicates that Internet bottlenecks lie at the network edge, within the ”first mile” or ”last mile” of transmission [3]. MCP exhibits a greater promise in such environments with lower levels of flow multiplexing on bottleneck links.

This paper is organized as follows. Section II reviews MCP. Section III presents related designs. Section IV evaluates MCP. Section V concludes the paper with a summary and discussion of future work.

Mode	Bottleneck link utilization		Fairing bit	Control rule
	Range	Encoding		
Scaling	$[0; 0.48)$	00	0 or 1	MI(2)
Fairing	[0.48;0.98)	01 or 10	1	AI(80 kbps)
Enhancing	[0.48;0.88)	01	0	MI(1.1)
Smoothing	[0.88;0.98)	10	0	AI(80 kbps) until first overload then AD(80 kbps) once
Stable	[0.88;0.98)	10	0	constant
Overloaded	$[0.98; \infty)$	11	0 or 1	MD(0.5)

Fig. 1. Modes of MCP operation

II. Review of MCP

A. Explicit communication format

MCP allocates four bits in the header of each data packet for explicit communication between hosts and routers. Two of the bits are used to notify the sender about the bottleneck link utilization of its data path. The other two bits (fairing bit and this-path bit) enable the sender to urge all flows sharing its bottleneck links to operate in the fairing mode.

B. Router operation

Routers provide explicit feedback to senders through receivers. To form the feedback, each router periodically computes utilizations of its output links. Routers also set fairing bits in forwarded packets to disseminate to appropriate flows a request of operating in the fairing mode.

C. Sender operation

The sender operates in one of the following six modes: scaling, overloaded, fairing, enhancing, smoothing, and stable. The choice of the mode depends on explicit feedback in accordance with Figure 1. While the original MCP proposal describes the modes in detail [2], below we provide only a brief rationale for each of the modes:

- The *fairing mode* is for convergence of MCP to fairness and exercises Additive-Increase Multiplicative-Decrease (AIMD) control [4] for seven increase-decrease cycles. According to our analysis [5], seven AIMD(x; 0.5) oscillations suffice for highly fair sharing. Figure 2 compares our analytical predictions with MCP steady-state fairness in ns-2 [6] simulations for different durations of the fairing mode. The convergence is the slowest when number n of flows on a shared bottleneck link is minimal, i.e., 2. Although the predictions are somewhat off due to simplicity of our model and ignored impacts of other MCP modes, the packet-level simulations confirm that that the seven-cycle longevity of the fairing mode provides high fairness.
- The *scaling mode* is for scalable increase of the bottleneck link utilization to at least 48%.

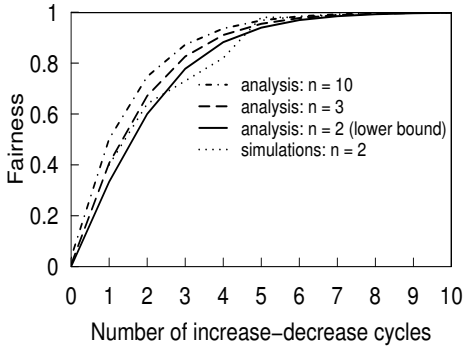


Fig. 2. MCP fairness convergence: analysis of AIMD(x; 0.5) versus ns-2 simulations of MCP steady-state fairness for different durations of the fairing mode.

- The *enhancing mode* is for scalable increase of the bottleneck link utilization to at least 88%.
- The *smoothing mode* raises the bottleneck link utilization further toward 98% with cautious additive steps avoiding a buildup of the link queue.
- The *overloaded mode* provides MCP with scalable response to overload of the bottleneck link.
- The *stable mode* is a steady-state regime of constant-rate transmission that sustains high utilization and low queuing at the bottleneck link.

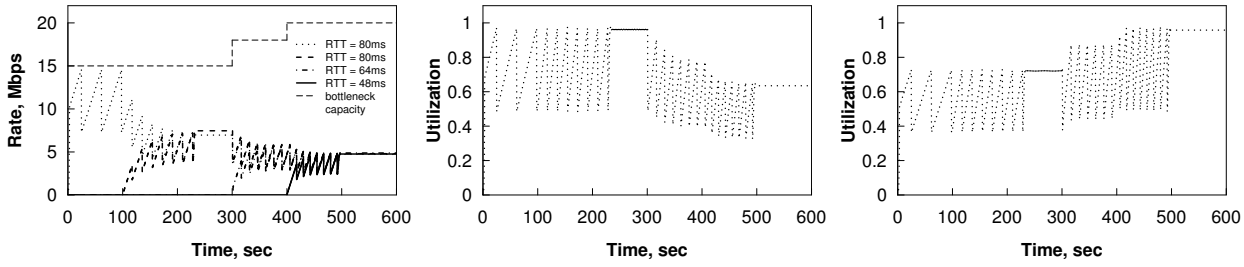
D. Receiver operation

The receiver sends an acknowledgment packet (ACK) for every incoming data packet. ACK echoes the fairing bit and encoding of the the bottleneck link utilization.

III. Related work

The problem of providing delay-sensitive applications with low queuing of their packets at network links has a lot of extensive and diverse related work. Below, we just briefly discuss some of investigated approaches to keeping link queues short.

Fair queuing algorithms such as Weighted Fair Queueing (WFQ) [7] and Deficit Round Robin (DRR) [8] maintain a separate queue for each flow sharing the link and service the queues in a fair manner. The flow



(a) (b) (c)

Fig. 3. Reaction of MCP to bottleneck link migrations: (a) transmission rates of the four flows, (b) utilization of the 15-Mbps link AB, and (c) utilization of the 20-Mbps link CD.

isolation protects a delay-sensitive flow from queuing delays of other traffic. However, the solution requires costly per-flow state, and the delay-sensitive application still needs an end-to-end congestion control protocol to keep the size of its own queue small.

Small link buffers [9], [10] assure that a shared queue stays short. This approach also requires a complementary end-to-end congestion control to ensure that buffer overflow and link underutilization do not disrupt application performance. E-TCP [11] is a recently proposed loss-driven protocol for such congestion control.

Delay-based congestion control is represented by such protocols as Congestion Avoidance using Round-trip Delay (CARD) [12] and TCP Vegas [13]. In this approach, a flow measures its RTT and curbs transmission when RTT increases. The reaction to rising delays is helpful for avoiding buffer overflows but unfortunately comes only after the link queue has started to grow.

Explicit congestion feedback from routers enables a congestion control protocol to prevent queuing. Depending on whether the explicit feedback consumes few bits per packet or more, explicit congestion control protocol can be classified as limited-feedback and rich-feedback. Rich-feedback designs include eXplicit Control Protocol (XCP) [14], Rate Control Protocol (RCP) [15], and JetMax [16]. Examples of limited-feedback protocols are Explicit Congestion Notification (ECN) [17] and Variable-structure congestion Control Protocol (VCP) [18]. MCP belongs to the latter category of limited-feedback designs.

Smooth adjustment algorithms bring a promise of combining short link queues with high utilization of bottleneck links in the steady state. Both linear and nonlinear adjustments have been proposed for smooth congestion control [19], [20]. Since transmission at constant rates is the smoothest, the stable mode of MCP is an extreme representative of this algorithmic family.

IV. Experimental evaluation

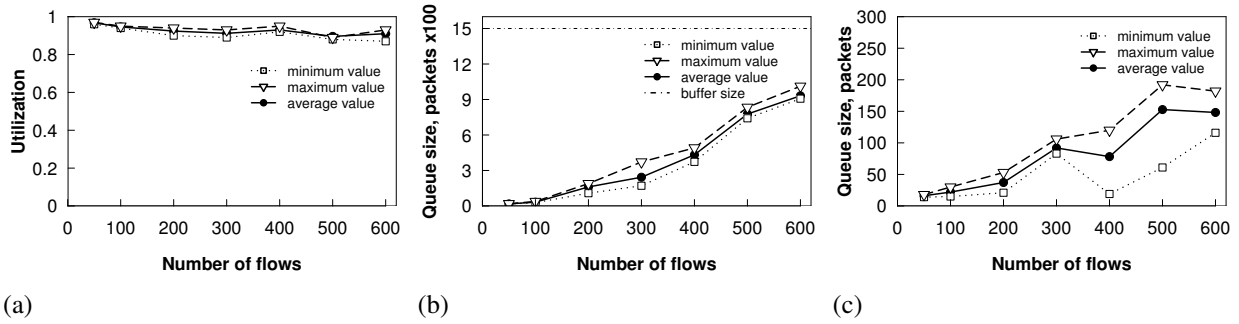
In this section, we report our simulations conducted in version 2.29 of ns-2 [6] and discuss the results. General settings in our experiments are as follows: a packet size equals 1000 bytes; propagation delay of a bottleneck link is 8 ms; buffer size of a link is equal to the product of the link capacity and the minimum RTT among the flows in the simulation; link queuing discipline is FIFO. Unless stated otherwise, the capacity of a bottleneck link is 20 Mbps, the capacities of non-bottleneck access links are 40 Mbps, and the network topology is a single-bottleneck dumbbell. To trace changes of a queue size in time, we sample the queue size every 10 ms. To plot the dependency of a queue size on a parameter, we measure the instantaneous value of the queue size. In the experiments, when we vary a single parameter while keeping all the other parameters fixed, we conduct 5 simulations for each value and report the minimum, average, and maximum values of each performance metric in the simulations.

A. Convergent behavior with migrating bottlenecks

We illustrate MCP convergent behavior in a parking-lot topology [5] with three potential bottleneck links AB, BC, and CD that have capacity 15 Mbps, 18 Mbps, and 20 Mbps respectively. The other links have capacity 40 Mbps. All links have propagation delay 8 ms. Propagation RTTs for four examined flows are 80 ms, 80 ms, 64 ms, and 48 ms. After the third flow arrives at time 300 seconds, the bottleneck migrates from AB to BC. Upon arrival of the fourth flow at time 400 seconds, the bottleneck migrates again, now to link CD. Figure 3 shows that MCP reacts to the bottleneck link migrations by converging the flows to new fair efficient rates and low in-network queuing.

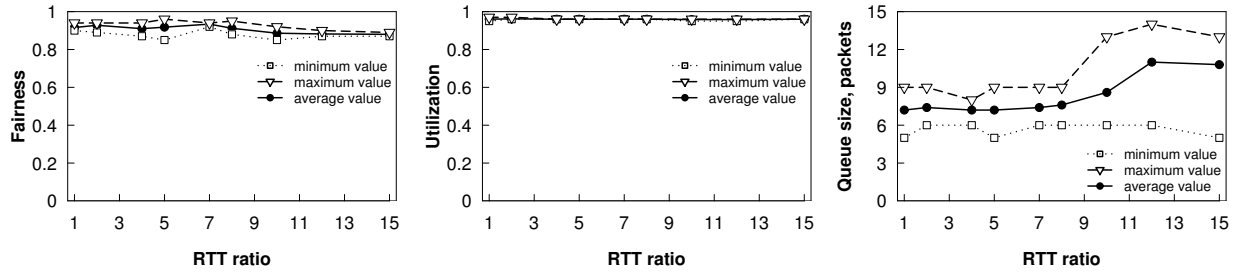
B. Dependence on the number of flows

Figure 4 depicts MCP bottleneck link utilization and queuing with different numbers of flows in the single-bottleneck dumbbell topology where the core bottleneck



(a) (b) (c)

Fig. 4. Dependence of MCP performance on the number of competing flows: (a) utilization of the bottleneck link, (b) peak queue size at the bottleneck link, and (c) peak queue size at the bottleneck link in the stable state.



(a) (b) (c)

Fig. 5. Independence of MCP steady-state operation from RTT when the bottleneck link capacity is 50 Mbps: (a) fairness index, (b) utilization of the bottleneck link, and (c) peak queue size at the bottleneck link.

link has capacity 200 Mbps and propagation delay 24 ms while every access link has capacity 400 Mbps and propagation delay 3 ms. The average link utilization remains relatively stable (it varies between 0.896 and 0.968) but the queue size is significantly more sensitive to the flow count. Although Figure 4b shows that MCP avoids buffer overflow in all the conducted simulations, per-flow buffer consumption under MCP is about 1.6 packets in general and about 0.26 packets in the steady state.

The above experiments demonstrate that queuing under MCP has imperfect population scalability because no constant buffer size precludes buffer overflow with arbitrarily many flows. Our ongoing study shows that the imperfection is fundamental for asynchronous congestion control. However, we also observe that the rate-based MCP has significant headroom for improving its population scalability. To realize this potential, we will investigate window-based control for at least some of the MCP modes, e.g., stable mode.

C. RTT heterogeneity

We simulate ten competing MCP flows with different RTTs and use $\frac{P_{max}}{P_{min}}$ as a control parameter where P_{max} and P_{min} respectively refer to the maximum and minimum propagation RTT of all the flows. P_{min} is always set to 20 ms. Propagation RTTs of other flows are uniformly distributed between P_{min} and P_{max}. Figure 5 confirms that MCP steady-state operation is relatively independent of RTT heterogeneity.

D. Packet-size heterogeneity

We also examine the impact of packet-size heterogeneity. Propagation RTTs of the flows are in the range between 20 and 100 ms. We employ $\frac{S_{min}}{S_{max}}$ as a control parameter where S_{max} and S_{min} denote respectively the maximum and minimum packet size across all the flows. Packets within each particular flow are of the same size. S_{max} is always fixed to 1500 bytes. Figure 6 shows relative immunity of MCP steady-state performance to heterogeneity in packet sizes.

E. Comparison of MCP with VCP

Since VCP is the closest to MCP in terms of its design aspirations and features, we compare MCP with VCP in the topology where the bottleneck link capacity is 500 Mbps, and the capacities of the access links are 1 Gbps. In both experiments, 1000 flows with propagation RTT 60 ms arrive at random times between 0 and 1 second. Figure 7 traces the bottleneck link utilization and queue size under each protocol. The peak queue size is close to 39% under VCP and about 31% under MCP. However, while the peak queue size under VCP stays at the same level in the steady state, MCP reduces its peak

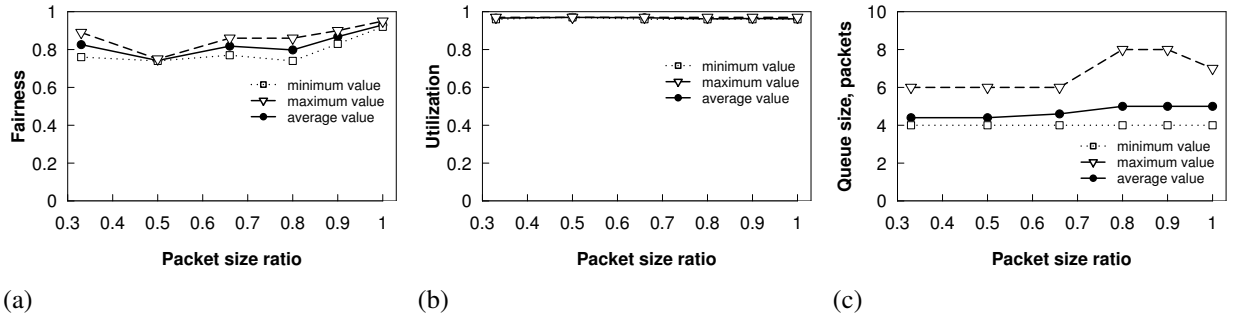


Fig. 6. Independence of MCP steady-state operation from packet sizes when the bottleneck link capacity is 50 Mbps: (a) fairness index, (b) utilization of the bottleneck link, and (c) peak queue size at the bottleneck link.

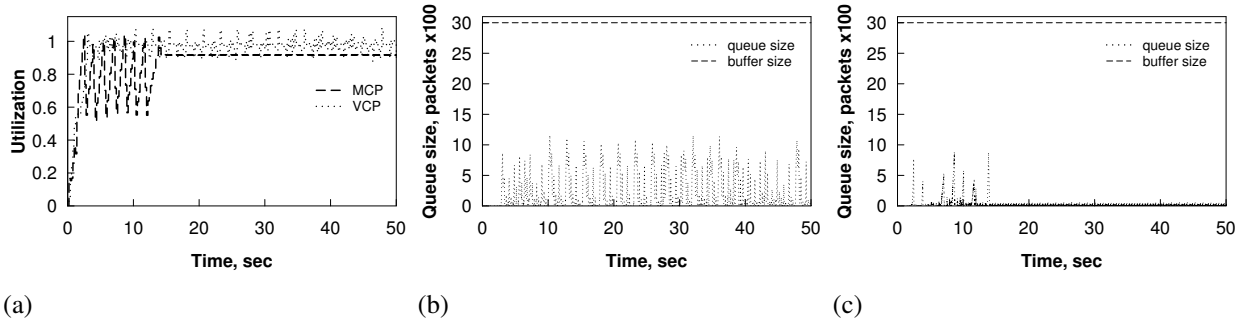


Fig. 7. MCP versus VCP: (a) bottleneck link utilization, (b) bottleneck link queuing under VCP, and (c) bottleneck link queuing under MCP.

queue size in the stable state dramatically to 1.7% of the buffer size.

F. Validation of MCP parameter values

MCP is a protocol with multiple parameterized modes. We experimentally examined whether chosen parameter values are reasonable. Section II already reported our validation for the 7-cycle longevity of the fairing mode. Figure 8 evaluates the choice of 1.1 as the MI factor in the enhancing mode. The plotted peak queue sizes are from experiments where 10 flows with RTTs between 20 and 200 ms share a 50-Mbps bottleneck link. MI factors below 1.1 do not reduce bottleneck queuing but undermine capacity scalability of the enhancing mode. On the other hand, raising the MI factor beyond 1.15 produces much longer queues and eventual buffer overflow.

V. Summary and discussion

In this paper, we presented and evaluated MCP, a congestion control protocol for low stable-state queuing at bottleneck links. To achieve its design objectives, the protocol engages hosts and routers in limited explicit communication and exploits the insight that the transmission should be kept constant after converging to fair efficient rates. For convergence to fair transmission rates, MCP incorporates a novel explicit-communication mechanism that allocates fairing and this-path bits in the header of each data packet. These two bits enable the sender of a flow (e.g., of a new flow) to urge all flows sharing its bottleneck links to operate in the fairing mode for seven increase-decrease cycles of AIMD(80 kbps; 0.5) control. Our analysis and simulations confirm that the 7-cycle longevity of the fairing mode is sufficient for highly fair sharing.

In addition to the fairing mode, MCP employs five more control modes. The choice of the current mode depends on the bottleneck link utilization communicated to the sender explicitly via two additional bits in data packet headers. The multimodal approach serves to equip MCP with all its desired properties, which include high bottleneck-link utilization, high fairness, and low queuing in the steady state. To make the stable transmission rates independent of RTT and packet sizes, MCP uses the transmission rate as a control parameter and prescribes uniform timing for rate adjustments in all flows.

Our evaluation of MCP and its comparison with VCP show that, by and large, MCP meets its design objectives. One major deviation is the undesirable growth of the bottleneck link queue as the number of competing flows rises. In our future work, we will investigate how to improve the population scalability of MCP operation. Below, we discuss our other three concerns about MCP design:

1) *Service for short flows* is not an intended application of MCP, which strives to provide small link queues for long delay-sensitive flows in the steady

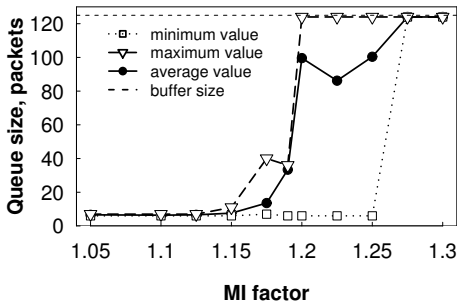


Fig. 8. Validation of 1.1 as the MI factor in MCP enhancing mode.

state. Since short flows care more about transient behavior, MCP innovations are orthogonal to needs of short flows and hence do not improve their performance. A more serious implication of short flows is a high frequency of their arrivals that might prevent long flows from reaping the benefits of the MCP stable mode. To address this problem, we plan to enhance MCP with a mechanism that distinguishes flows based on their size and provides them with size-specific services.

2) *Synchronous control* ensures that MCP flows having the same bottleneck link operate in the same mode. However, asynchrony of modes might be beneficial in some scenarios. For example, TCP might provide a new flow with faster convergence to a fair transmission rate than under MCP because the asynchronous TCP allows the new flow to operate in the aggressive slow-start mode whereas existing flows continue to operate in the congestion-avoidance mode, where TCP acquires the available capacity at a slower pace.
3) *Vulnerability to host misbehavior* is another concerning property of MCP. In particular, by setting the fairing and this-path bits persistently beyond the prescribed seven increase-decrease cycles, a malicious host can make all flows sharing its bottleneck link to keep operating in the fairing mode, causing needless oscillations of the bottleneck link utilization and queue size.

The last two concerns are not unique to MCP. For example, VCP also faces the issue of synchronous control. We plan to examine whether randomizing the mode selection under some circumstances is able to realize the potential benefits of asynchrony without undermining the overall performance of MCP. The mechanism for requesting the fairing mode of MCP operation adds a new avenue for attacks to the already wide arsenal available to malicious hosts. Since senders and receivers might collude, effective protection against host attacks necessitates router assistance. We will study lightweight router techniques for making MCP resilient to host misbehavior. Due to the mode synchronization and uniform timing of transmission adjustments, it seems easier for routers to detect misbehaving MCP flows than flows of other protocols where transmission rates depend on non-uniform packet losses, RTT, and packet sizes.

REFERENCES

[1] V. Jacobson, "Congestion Avoidance and Control," in *Proceedings ACM SIGCOMM 1988*, August 1988.

[2] M. Podlesny and S. Gorinsky, "Multimodal Congestion Control for Low Stable-State Queuing," in *Proceedings IEEE INFOCOM Minisymposium 2007*, May 2007.

[3] A. Odlyzko, "The Many Paradoxes of Broadband," *First Monday*, vol. 8, no. 9, September 2003.

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

[5] M. Podlesny and S. Gorinsky, "Multimodal Congestion Control for Low Stable-State Queuing," Department of Computer Science and Engineering, Washington University in St. Louis, Tech. Rep. WUCSE-2006-41, *www.arl.wustl.edu/~ gorinsky/pdf/WUCSE-TR-2006-41.pdf*, August 2006.

[6] S. McCanne and S. Floyd, *ns Network Simulator. http://www.isi.edu/nsnam/ns/.*

[7] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," in *Proceedings ACM SIGCOMM 1989*, September 1989.

[8] M. Shreedhar and G. Varghese, "Efficient Fair Queueing Using Deficit Round Robin," in *Proceedings ACM SIGCOMM 1995*, September 1995.

[9] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing Router Buffers," in *Proceedings ACM SIGCOMM 2004*, September 2004.

[10] S. Gorinsky, A. Kantawala, and J. Turner, "Link Buffer Sizing: A New Look at the Old Problem," in *Proceedings IEEE Symposium on Computers and Communications (ISCC 2005)*, June 2005.

[11] Y. Gu, D. Towsley, C. Hollot, and H. Zhang, "Congestion Control for Small Buffer High Speed Networks," in *Proceedings IEEE INFOCOM 2007*, May 2007.

[12] R. Jain, "A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks," *ACM Computer Communications Review*, vol. 19, no. 5, pp. 56–71, October 1989.

[13] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," in *Proceedings ACM SIGCOMM 1994*, August 1994.

[14] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," in *Proceedings ACM SIGCOMM 2002*, August 2002.

[15] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown, "Processor Sharing Flows in the Internet," in *Proceedings International Workshop on Quality of Service (IWQoS 2005)*, June 2005.

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

[17] K. Ramakrishnan and S. Floyd, "A Proposal to Add Explicit Congestion Notification (ECN) to IP," RFC 2481, January 1999.

[18] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One More Bit Is Enough," in *Proceedings ACM SIGCOMM 2005*, August 2005.

[19] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-Based Congestion Control for Unicast Applications," in *Proceedings ACM SIGCOMM 2000*, August 2000.

[20] D. Bansal and H. Balakrishnan, "Binomial Congestion Control Algorithms," in *Proceedings IEEE INFOCOM 2001*, April 2001.