This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2007 proceedings.

Multimodal Congestion Control for Low Stable-State Queuing

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*— To discover an efficient fair sending rate for a flow, Transmission Control Protocol (TCP) saturates the bottleneck link and its buffer until the router discards a packet. Such TCP-caused queuing is detrimental for interactive and other delay-sensitive applications. In this paper, we present Multimodal Control Protocol (MCP) which strives to maintain low queues and avoid congestion losses at network links. The multimodal MCP engages routers and hosts in limited explicit communication. A distinguishing property of MCP is stable transmission after converging to efficient fair states. To ensure convergence to fairness, MCP incorporates an innovative mechanism that enables a flow to urge all flows sharing its bottleneck links to operate in a fairing mode, dedicated to fairness improvement. To make the stable fair rates independent of round-trip times and packet sizes, MCP employs rate-based control and uniform timing of adjustments.**

I. Introduction

Transmission Control Protocol (TCP) [1] is the most prominent representative of a popular congestion control paradigm where a flow increases its transmission 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 some applications. For example, human perception of an interactive multimedia application might degrade dramatically after round-trip time (RTT) exceeds few hundred milliseconds.

In this paper, we explore how a congestion control protocol can accommodate delay-sensitive applications by keeping link queues short. This problem has a lot of extensive related work. Below, we just briefly discuss some of the investigated approaches to keeping the queuing low.

Fair queuing algorithms [2]–[4] maintain a separate queue for each flow sharing the link and service the queues in a fair manner. The flow 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 [5], [6] 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 [7].

In delay-based congestion control [8]–[10], 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 rich-feedback [11]–[13] and limited-feedback [14], [15].

In this paper, we develop Multimodal Control Protocol (MCP) which belongs to the latter category of limited-feedback explicit congestion control protocols. We derive MCP design from the following requirements:

1) Controlled flows should converge to utilizing the network efficiently and fairly, where fairness means equal rates on shared bottleneck links.
2) Heterogeneity of RTTs or packet sizes should not undermine fairness.
3) At any network link, queuing should be low, and losses should be avoided.
4) Congestion control overhead should be limited to few bits per packet.
5) Response to changes in communication demands and network capabilities should scale well.

Due to the space constraints, we relegate a rationale for the requirements to an extended version of this work [16].

A distinguishing property of MCP is stable transmission after converging to efficient fair states. To ensure convergence to fairness, MCP incorporates an innovative mechanism that enables a flow to urge all flows sharing its bottleneck links to operate in a fairing mode, dedicated to fairness improvement. To make the stable fair rates independent of round-trip times and packet sizes, MCP employs rate-based control and uniform timing of adjustments.

The rest of the paper is organized as follows. In Section II, we formulate our main design principles and derive MCP. Section III conducts evaluation of the protocol. Section IV concludes the paper with a summary.

II. MCP DESIGN

A. Design principles

Before presenting the full MCP design, we highlight its key principles. Since satisfying all of the above requirements with a single simple algorithm is difficult, if not infeasible, we adhere to a multimodal approach:

Principle 1: Incorporate multiple modes of operation where each mode pursues a subset of the design objectives.

To support low queuing in the steady state, we keep steady-state transmission as smooth as possible, i.e., constant:

Principle 2: Keep the transmission constant after achieving high efficiency and fairness.

The ability of MCP to reach such a stable mode rests on an insight that a relatively short sequence of transmission adjustments suffices to provide high fairness:

Principle 3: Incorporate a fairing mode and operate in it only as long as needed for convergence to sufficient fairness.

B. Link utilization as explicit feedback

In MCP, the bottleneck link utilization serves as explicit feedback to the sender. The feedback involves computing the link utilization at every router on the data path of the flow.

1) Timing: The router measures utilization for each of its output links. In particular, all routers adhere to uniform timing by measuring the link utilization periodically with the same period of duration T. According to statistics [17], [18], between 75% and 90% of all Internet flows have RTT less than 200 ms. Therefore, to alleviate unnecessary oscillations of the end-to-end control, we use 200 ms as the value of T.

2) Calculation: The router employs the following equation to compute utilization U of a link:

$$U = \frac{N + Q}{CT} \tag{1}$$

where N is the amount of data that has arrived for the link during the previous period, Q is the minimum size of the link queue during this previous period, and C is the link capacity.

3) Communication: Due to Requirement 4, the router encodes the calculated link utilization into few bits. If E_1 and E_2 denote respectively encodings of utilizations U_1 and U_2, then $E_1 > E_2$ is equivalent to $U_1 > U_2$. The sender transmits each data packet with the lowest encoding. When the router receives a packet, the router compares encoding E of the local link utilization and encoding P contained in the packet header. If $E > P$, then the router resets the encoding in the packet header to E. After the packet reaches the receiver, the encoding in the packet header represents the bottleneck link utilization for the data path of the flow. The receiver echoes this encoding to the sender via an acknowledgment (ACK) packet.

C. Scaling mode

Now, we discuss how the sender benefits from received encoding E of bottleneck link utilization U. Whenever E indicates that U is below 48%, the sender operates in a *scaling mode* designed for scalable increase of U into the area of relatively light underutilization where U becomes between between 48% and 96%. The scaling mode achieves this by using MI(2), multiplicative increase with factor 2.

D. Overloaded mode

Another extreme on the bottleneck utilization spectrum is represented by the encoding that corresponds to $U \geq 0.98$. When U is at least 98%, the sender is in an *overloaded mode* and uses MD(0.5), i.e., multiplicative decrease with factor 0.5. The only exception to this rule is discussed later in the context of another mode. The overloaded mode provides MCP with scalable response to severe overloads of the bottleneck link. The choice of factor 0.5 ensures that a decrease in the overloaded mode does not lower the bottleneck link utilization below 49%.

E. Fairing mode

MCP concerns itself with fairness improvement and uses a *fairing mode* for these purposes only when $U \in [0.48; 0.98)$. The fairing mode improves fairness by using AI(80 kbps), i.e., additive increase with coefficient 80 kbps. The measurement of the increase step in bits per second ensures the independence of a flow throughput from RTT and packet sizes. Whereas TCP, VCP (Variable-structure congestion Control Protocol) [15], and other protocols continue adjusting the transmission even when the bottleneck link is utilized efficiently and shared fairly, these further oscillations yield no meaningful improvement in fairness but cause such undesirable effects as long queuing at the bottleneck link or low utilization of the link capacity. Thus, MCP operates in the fairing mode only as long as needed to achieve sufficient fairness.

1) Time to stay in the fairing mode: To determine an appropriate longevity for the fairing-mode operation, we conduct analysis in the classical Chiu-Jain model. Detailed descriptions of the model and our analysis are relegated to the extended version of this work [16]. Our main conclusion is that a small number of increase-decrease cycles is sufficient for AIMD(x; 0.5) to provide high fairness, where x is an additive increase step, and 0.5 is a multiplicative decrease factor. In particular, MCP operates in the fairing mode for *seven increase-decrease cycles*.

2) Mechanism for switching into the fairing mode: When a flow terminates, or the capacity of the bottleneck link changes, the fairness index does not decrease. Hence, the mechanism for switching into the fairing mode is important primarily when new flows start. A less common but plausible scenario occurs when an application that has willingly generated a data flow at an unfairly low rate decides to increase the flow rate to the fair share. Handling such scenarios is not straightforward. Without assistance from the network, it is extremely difficult for existing flows to detect that their bottleneck links have started to serve a new flow. Even the router of the bottleneck link has no effective implicit means to infer the desire of an existing slow flow to reclaim its fair share of the link capacity.

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

MCP incorporates an explicit communication mechanism that enables a flow to urge all flows sharing its bottleneck links to operate in the fairing mode. More specifically, whenever a flow wants to improve fairness of the bottleneck link sharing, the sender of the flow sets a *fairing bit* and *this-path bit* in the headers of all data packets transmitted during the next seven increase-decrease cycles. If a packet with the set fairing and this-path bits arrives for being forwarded to a link, then the router does the following for all data packets of *all* flows forwarded into the link before the end of the next period of the link-utilization measurement: before forwarding the packet into the link, the router checks the link-utilization encoding in the packet header; if the encoding corresponds to utilization range $[0.48; 0.98)$, the router sets the fairing bit in the header of the packet. Whenever the receiver of a flow receives a data packet with the set fairing bit, the receiver echoes the set fairing bit to the sender of the flow via an ACK packet. After the sender receives an ACK packet with the set fairing bit and link-utilization encoding that indicates the bottleneck link utilization between 48% and 98%, the sender switches into the fairing mode. Introduction of the this-path bit into the mechanism ensures that a request for operating in the fairing mode affects only those flows that share bottleneck links with the requesting flow [16].

F. Enhancing mode

From now on, we discuss MCP operation when no flow demands the fairing mode but $U \in [0.48; 0.98)$. Although high fairness is achieved by this point, the bottleneck link utilization can be as low as 48%. To improve the utilization, we split $[0.48; 0.98)$ into ranges $[0.48; 0.88)$ and $[0.88; 0.98)$. When $U \in [0.48; 0.88)$, MCP is in an *enhancing mode* and uses MI(1.1), which ensures a scalable increase into the $[0.88; 0.98)$ range without overshooting into the overloaded mode.

G. Smoothing mode

After rising from the enhancing mode, MCP switches into a *smoothing mode* where $U \in [0.88; 0.98)$. The goal is to push the bottleneck link utilization further up, while being cautious not to cause a buildup of the link queue. The smoothing mode employs AI(80 kbps) until the first overload (i.e., until U becomes at least 98%) and then applies AD(80 kbps) once to negate the previous overloading increase. After the backtracking, MCP switches into a *stable mode*.

H. Stable mode

The stable mode is the ultimate regime of constant transmission prescribed by Principle 2. A flow stays in the stable state as long as $U \in [0.88; 0.98)$ and no other flow requests a switch into the fairing mode. Figure 1 summarizes all modes of MCP operation.

I. Transmission adjustment timing

The initial transmission rate of a flow equals 80 kbps. The sender adjusts the rate once per $2T$. The only exception occurs when feedback indicates nascent overload: in such circumstances, the sender decreases the rate immediately. After starting transmission of a packet of size S bytes, the sender schedules the next packet for $\frac{8S}{R}$ seconds later, where R denotes the current transmission rate in bps. Whenever the sender adjusts R, the sender also recalculates $\frac{8S}{R}$ and adjusts accordingly the remaining wait for the next packet.

III. Experimental evaluation

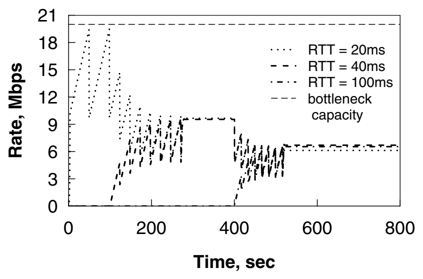
We simulate MCP in ns-2 version 2.29 [19]. Unless specified otherwise, our experiments involve a single-bottleneck dumbbell topology, packets sized to 1000 bytes, bottleneck link with propagation delay 8 ms, and FIFO link buffers sized to the product of the link capacity and maximum RTT among simulated flows. To plot the dependency of a queue size on time, we sample the queue size every 10 ms. We also measure the instantaneous queue size for plotting its dependency on a network parameter. Reported maximum, minimum, and average values are for five experiments conducted for each setting of the parameter.

A. Convergence

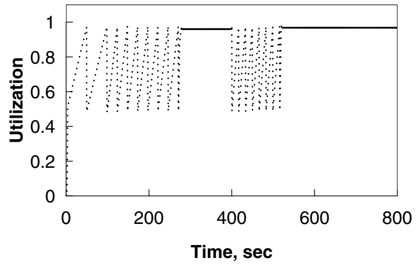
To illustrate the convergent behavior of MCP, we simulate 3 flows with RTTs 20 ms, 40 ms, and 100 ms over the bottleneck link with capacity 20 Mbps. The simulation lasts 800 sec. Figure 2 shows stabilization of the flow rates. In the stable state, the bottleneck link utilization is 0.97, fairness index [16] is 0.91, and peak queue size is 7 packets.

B. Capacity scalability

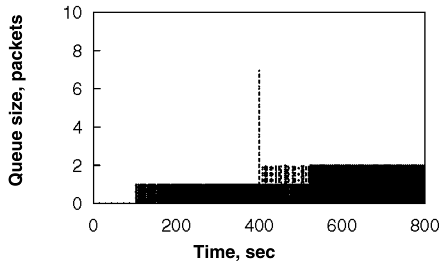
To study scalability of the protocol with respect to the bottleneck link capacity, we use 20 flows with RTTs uniformly distributed in the range from 20 to 100 ms. Each simulation lasts 480 sec. The bottleneck link capacity varies from 20 to 500 Mbps. The flows arrive randomly between 0 and 10 sec.



(a)

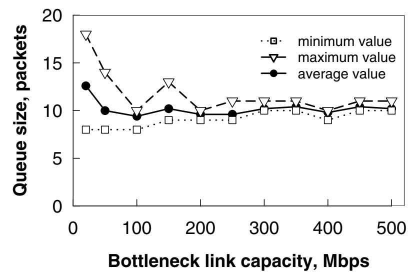


(b)

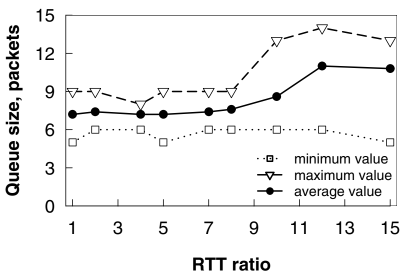


(c)

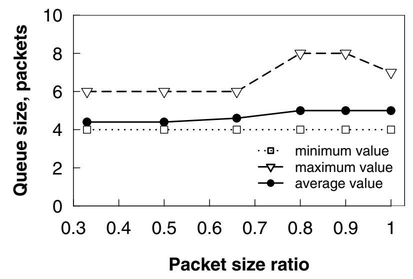
Fig. 2. Convergence to fairness, high utilization, and low queue size: (a) sending rates of the three flows; (b) utilization of the bottleneck link; (c) queue size at the bottleneck link.



(a)



(b)



(c)

Fig. 3. Influence of network parameters on the peak queue size at the bottleneck link: (a) capacity scalability; (b) heterogeneous RTTs; (c) heterogeneous packet sizes.

Figure 3a shows that the peak queue size at the bottleneck link stays about the same, with the average between 9.4 and 12.6 packets. The average fairness index is in the range of 0.776 - 0.886, and average link utilization is 0.924 - 0.970.

C. Heterogeneous RTTs

To evaluate MCP under different RTTs, we conduct simulations with 10 flows. Each flow arrives randomly between 0 and 10 sec. All simulations last 120 sec. To represent the RTT heterogeneity, we use $\frac{RTT_{max}}{RTT_{min}}$ where $RTT_{min} = 20$ ms and RTT_{max} are respectively the minimum and maximum propagation RTT. Propagation RTTs of other flows are uniformly distributed between RTT_{min} and RTT_{max}. Figure 3b plots dependence of the peak queue size at the bottleneck link on the RTT ratio. The average peak queue size is in the range of 7.2 - 11.0 packets, fairness index is 0.880 - 0.934, and utilization is 0.958 - 0.962.

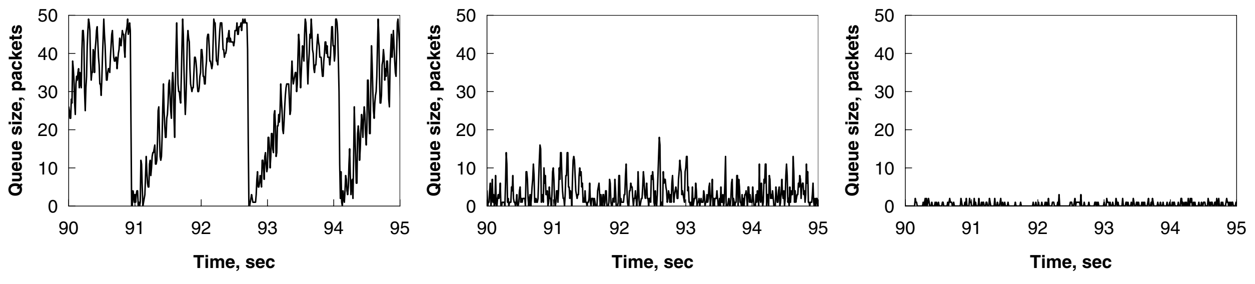
D. Heterogeneous packet sizes

In these simulations, we use 5 flows. Propagation RTTs of the flows are uniformly distributed between 20 and 100 ms. Flow arrival times are random between 0 and 5 sec. Each simulation lasts 200 sec. Packets within a flow are of the same size. To capture packet size heterogeneity among flows, we employ $\frac{S_{min}}{S_{max}}$ where $S_{max} = 1500$ bytes and S_{min} denote respectively the maximum and minimum packet sizes. Other packet sizes are uniformly distributed between S_{min} and S_{max}. Figure 3c reveals that the packet size heterogeneity has little impact on the peak queue size, the average of which stays between 4.4 and 5.0 packets. Average fairness is in the range of 0.742 - 0.930, and utilization is 0.962 - 0.970.

E. Comparison with TCP and VCP

First, we examine the bottleneck link queuing under TCP, VCP, and MCP when 5 flows with RTT 20 ms, 40 ms, 60 ms, 80 ms, and 100 ms compete for the bottleneck link with capacity 20 Mbps and 50-packet buffer. The flows arrive between 0 and 5 sec. The experiment lasts 200 sec and is repeated for each of the three protocols. Figure 4 tracks the bottleneck queue size during the time interval between 90 and 95 seconds, when all the flows are already in the steady state. The peak queue size is 50 packets under TCP, 20 packets under VCP, and 6 packets under MCP. The bottleneck link utilization is about 0.92 under MCP and varies between 0.89 and 1 under VCP.

Then, we compare MCP and VCP in terms of average queuing when the bottleneck link capacity varies from 50 to 500 Mbps. The capacity of each access link is twice the bottleneck link capacity. Propagation RTTs of all 20 flows are distributed randomly between 22 and 200 ms. The bottleneck link has propagation delay 10 ms. The flows arrive between 0 and 1 sec according to the uniform distribution. Each experiment lasts 480 sec. Figure 5 depicts the bottleneck queue size as an average of queue sizes encountered by all packets. In comparison to VCP, MCP reduces the average queue size by a factor of about 20.



(a) TCP (b) VCP (c) MCP

Fig. 4. Comparison of TCP, VCP, and MCP in terms of queuing at the bottleneck link in the stateady state.

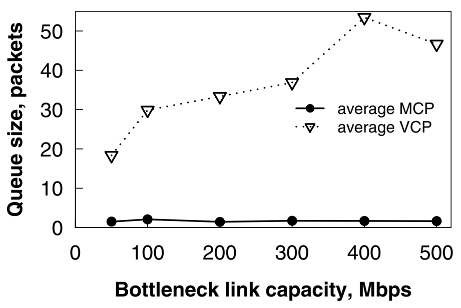


Fig. 5. Average queuing at the bottleneck link under MCP and VCP.

IV. Conclusion

We presented Multimodal Control Protocol (MCP), a congestion control scheme for low queuing in the steady network state. MCP design relies on three key principles: multiple modes of operation, sufficient fairness after a relatively short sequence of transmission adjustments, and constant-rate transmission in the steady state. One of MCP contributions is its mechanism for limited explicit communication between hosts and routers. The mechanism enables the sender of a 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. Simulations confirm that MCP transmission converges to utilize the network efficiently and fairly, is mostly insensitive to RTT and packet sizes, and maintains much lower queues in the steady state than under TCP or VCP.

MCP is geared for long flows in networks where flows arrive to a bottleneck link infrequently. While Internet core traffic is highly dynamic, Odlyzko indicates that congestion typically occurs within the ”first mile” or ”last mile” of transmission [20]. MCP is a better match for such environments with lower levels of flow multiplexing on bottleneck edge links. In our future work, we will adapt MCP for networks with more intense bottleneck dynamics. In particular, we will explore how to accommodate short flows effectively without disrupting the smooth steady-state MCP service for long flows. In addition to further evaluation of MCP, our plans include research on improving MCP efficiency through randomization and making the protocol resilient to host misbehavior.

References

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

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

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

[4] J. Bennet and H. Zhang, “WF2Q: Worst-Case Fair Weighted Fair Queueing,” in *Proceedings ACM INFOCOM 1996*, August 1996.

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

[6] 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.

[7] Y. Gu, D. Towsley, C. Hollot, and H. Zhang, “Congestion Control for Small Buffer High Speed Networks,” University of Massachusetts Amherst, Tech. Rep. CMPSCI 06-14, April 2006.

[8] 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.

[9] L. Brakmo, S. O’Malley, and L. Peterson, “TCP Vegas: New Techniques for Congestion Detection and Avoidance,” in *Proceedings ACM SIGCOMM 1994*, August 1994.

[10] R. King, R. Baraniuk, and R. Riedi, “TCP-Africa: An Adaptive and Fair Rapid Increase Rule for Scalable TCP,” in *Proceedings IEEE INFOCOM 2005*, March 2005.

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

[12] 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.

[13] 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.

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

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

[16] 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.

[17] H. Jiang and C. Dovrolis, “Passive Estimation of TCP Round-Trip Times,” *ACM Computer Communications Review*, vol. 32, no. 3, pp. 75–88, July 2002.

[18] V. Paxson, “End-to-End Internet Packet Dynamics,” in *Proceedings ACM SIGCOMM 1997*, September 1997.

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

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