Report Number: WUCSE-2006-41

2006-01-01

Multimodal Congestion Control for Low Stable-State Queuing

Maxim Podlesny and Sergey Gorinsky

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... **Read complete abstract on page 2.**

Multimodal Congestion Control for Low Stable-State Queuing

Maxim Podlesny and Sergey Gorinsky

Complete 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. The reported evaluation of MCP confirms achieving its design objectives.

2006-41

Multimodal Congestion Control for Low Stable-State Queuing

Authors: Maxim Podlesny, Sergey Gorinsky

Corresponding Author: podlesny@arl.wustl.edu

Web Page: http://www.arl.wustl.edu/~gorinsky/pubs.html

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. The reported evaluation of MCP confirms achieving its design objectives.

Type of Report: Other

Multimodal Congestion Control for Low Stable-State Queuing

Maxim Podlesny and Sergey Gorinsky

Technical Report WUCSE-2006-41
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*

August 2006

***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. The reported evaluation of MCP confirms achieving its design objectives.**

I. Introduction

Transition 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* such as Weighted Fair Queueing (WFQ) [2], Packet-by-packet Generalized Processor Sharing (PGPS) [3], Deficit Round Robin (DRR) [4], and Worst-case Fair Weighted Fair Queueing (WF2Q) [5] 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* [6]–[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. Evolutional TCP (E-TCP) [11] is a recent loss-driven proposal for such congestion control.
- *Delay-based congestion control* is represented by such protocols as Congestion Avoidance using Round-trip Delay (CARD) [12], TCP Vegas [13], and TCP Africa [14]. In this approach, a flow measures its RTT and curbs transmission when RTT increases. The reaction to raising 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) [15], Rate Control Protocol (RCP) [16], and JetMax [17]. Examples of limited-feedback protocols are Explicit Congestion Notification (ECN) [18] and Variable-structure congestion Control Protocol (VCP) [19].

In this paper, we develop Multimodal Control Protocol (MCP), which belongs to the latter category of explicit limited-feedback congestion control protocols. 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. Section II describes requirements that we impose on MCP. Section III discusses our design principles. In Section IV, we derive MCP design from the above principles. Section V conducts evaluation of the protocol. Finally, Section VI concludes the paper with a summary and discussion.

II. Control requirements

This section presents requirements that we impose on a desired protocol. One of them is the common congestion control objective that packet flows generated by applications should utilize the network efficiently and share it fairly. By efficiency, we mean high utilization of bottleneck links. In our view of fairness, flows that contend for a bottleneck link should acquire equal shares of the link bitrate. Hence, our first design requirement is as follows:

Requirement 1: Controlled flows should converge to utilizing the network efficiently and fairly, where fairness means equal rates on shared bottleneck links.

TCP does not support such maxmin fairness because the sending rate of a TCP flow depends on the packet size and RTT of the flow. Since local flows might compete for a bottleneck link with flows that span countries or even continents, we find a TCP-like dependence of the stable transmission rate on RTT unacceptable. We also consider as undesirable any dependence of the fair share on packet sizes. Therefore, we add the following clarifying requirement:

Requirement 2: Heterogeneity of RTTs or packet sizes should not undermine fairness.

While the propagation component alone can make RTT of a flow high, extra queuing in the network can render the RTT unacceptable for the served application. For example, human perception of an interactive multimedia application is likely to degrade dramatically when RTT exceeds few hundred milliseconds. Another negative aspect of long queuing in routers is the possibility of packet discard when the queue grows too large. Such losses are undesirable because recovery from the losses raises communication overhead (e.g., when the recovery relies on forward error correction) or boosts delays even further (e.g., when the recovery is through retransmission). Thus, our third design requirement is as follows:

Requirement 3: At any network link, queuing should be low, and losses should be avoided.

It is important that the communication overhead of the protocol does not consume a significant fraction of the network capacity. Moreover, to facilitate integration of the design with current protocols, such as the Internet Protocol (IP) [20], we require that the congestion control protocol uses no more than few bits in the header of each packet:

Requirement 4: Congestion control overhead should be limited to few bits per packet.

TCP enables a new flow to acquire the available network capacity promptly. Unfortunately, the reaction of existing TCP flows to increases in bottleneck link capacities or declines in competing traffic is less scalable. When the available capacity is large, TCP in congestion avoidance converges to high utilization of the bottleneck link slowly. We postulate that the congestion control protocol should provide scalable responses to all possible changes in network conditions:

Requirement 5: Response to changes in communication demands and network capabilities should scale well.

III. Design principles

Satisfying all of the above requirements with a single simple algorithm is difficult, if not infeasible. To provide a new flow with scalable acquisition of the available capacity, TCP uses the slow-start mode which unfortunately does not assure convergence to fairness and might cause significant losses by overrunning the buffer of the bottleneck link. For fairness convergence and smoother transmission increases, TCP also employs the congestion-avoidance mode. The design of TCP highlights the promise of a *multimodal* approach where each mode of operation pursues only a subset of all the design requirements. This approach constitutes the basis for our first design principle:

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

Requirement 3 of low queuing in the network necessitates that flows react to incipient congestion of a link even before the router queue starts to build up. Hence, we opt for a design where routers inform hosts explicitly about the bottleneck link utilization. In accordance with Requirement 4, such explicit feedback should not consume more than few bits in the header of any packet. This leads us to:

Principle 2: To prevent queue buildups in routers, use the bottleneck link utilization as explicit few-bit feedback.

As per our Requirement 2, the stable transmission rate of a flow should be independent from RTT. To compensate the natural dependence of self-clocked protocols on RTT, the algorithm for adjusting the transmission window should explicitly account for RTT. VCP is an example of such solution. However, the RTT-aware transmission adjustment faces instability problems due to changes in RTT [19]. To satisfy Requirement 1, we select a different approach where the control parameter is not the congestion window but the sending rate, and all flows adhere to uniform rules and timing for transmission adjustment. Furthermore, to make the transmission rate independent of packet sizes, the protocol should view the rate in terms of bits, not packets. Hence, our third design principle is as follows:

Principle 3: Use the sending bitrate as the control parameter and employ uniform adjustment timing for all flows.

TCP, VCP, 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. This leads us to the following design principle that distinguishes our approach significantly from the existing protocols:

Principle 4: Incorporate a fairing mode and operate in it only as long as needed for convergence to sufficient fairness.

After the bottleneck link is utilized efficiently, and the fairing mode ensures high fairness, the protocol supports Requirement 3 the best if the transmission is kept the most smooth, i.e., constant. Hence, our last design principle is as follows:

Principle 5: Keep the transmission steady after achieving high efficiency and fairness.

IV. MCP DESIGN

Based on the principles formulated in Section III, we now design Multimodal Control Protocol (MCP). First, we describe MCP features in the order of their derivation from the design principles. Then, we summarize the protocol with respect to the following four general categories of its operation: explicit communication format, router, sender, and receiver algorithms.

A. Link utilization as explicit feedback

In accordance with Principle 2, MCP uses the bottleneck link utilization as explicit few-bit feedback. Computing the feedback involves computing the link utilization at every router on the data path of the flow. Below, we describe when and how this information is computed and communicated to the sender.

1) Timing: The router measures utilization for each of its output links. In support of our Principle 3, all routers adhere to uniform timing by measuring the link utilization periodically with the same period of duration T. According to statistics [21], [22], 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. Including Q into the equation helps MCP to avoid persistent long queuing.

3) Communication: Because Principle 2 limits the amount of communication overhead, 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.

B. Scaling mode

Now, we discuss how the sender benefits from received encoding E of bottleneck link utilization U. Whenever E indicates that the bottleneck link utilization is below 48%, the sender operates in a *scaling mode* designed for scalable increase of U into an area of relatively light underutilization. The scaling mode achieves this by using MI(2), multiplicative increase with factor 2. The choice of the factor ensures that upon leaving the scaling mode, MCP does not raise the bottleneck link utilization above 96%.

C. 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 the decrease 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 from the overloaded mode lowers the bottleneck link utilization to at most 49% and does not overswing MCP into the scaling mode.

D. 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, rather than packets per RTT, is in conformance with our Principle 3. The increase by 80 kbps corresponds to one 1000-byte packet per RTT of 100 ms, a common setting for TCP flows in congestion avoidance. In general, the duo of the fairing and overloaded modes provides fairness convergence via AIMD(80 kbps; 0.5) control similar to TCP congestion avoidance. According to Principle 4, flows should operate in the fairing mode only as long as needed to achieve sufficient fairness. Below, we specify how long the flows stay in the fairing mode, and how they decide when to switch into the fairing mode.

1) Time to stay in the fairing mode: To determine an appropriate longevity for operating in the fairing mode, we conduct analysis in the classical Chiu-Jain model where n distributed users adjust their loads on a shared resource in response to uniform binary feedback that indicates whether the total load exceeds the target load [23]. Each user i uses AIMD(x; 0.5) to adjust its load $l_i(t)$ at time t, where $i = 1, \ldots, n$. The particular value of x is not essential for our analysis, and the analytical results also apply to the AIMD(80 kbps; 0.5) control employed by MCP. We reason about fairness improvement under AIMD(x; 0.5) in terms of *increase-decrease cycles* where an increase-decrease cycle consists of adjustments between two peaks of the oscillating total load. We quantify fairness of the resource sharing at time t with the following fairness index:

$$F(t) = \min_{i,j=1}^{n} \frac{l_i(t)}{l_j(t)}. \quad (2)$$

The fairness index of 1 corresponds to the perfect fairness when all individual loads are equal.

To improve readability, we relegate detailed descriptions of our model and analysis to the Appendix. The main conclusion from the analysis is that a small number of increase-decrease cycles is sufficient for AIMD(x; 0.5) to provide high fairness. For scenarios where the increase step is small in comparison to the fair share, and new users do not outnumber existing converged users, we derive a lower bound for fairness improvement under AIMD(x; 0.5) and show that the fairness index grows after one increase-decrease cycle to at least 0.33, after two cycles to at least 0.60, after three cycles to at least 0.77, and after seven cycles to at least 0.98. Since we view 98% as sufficiently high fairness, MCP operates in the fairing mode for *seven increase-decrease cycles*. If during the process of fairness improvement the contending flows switch temporarily into the scaling mode (because of decline in competing traffic or increase in the bottleneck link capacity), the disruption does not affect the count of remaining 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 rate of the flow to the fair share of the bottleneck link capacity. 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.

Due to above reasons, and because Principle 2 allows explicit limited feedback, 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* in the headers of all data packets transmitted during the next seven increase-decrease cycles. When a packet with the set fairing bit arrives for being forwarded to a link, 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. This ensures that the receivers of all flows behind the shared bottleneck links learn about the need to switch into the fairing mode. 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 the ACK packet.

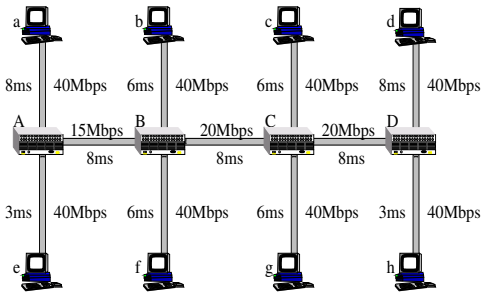


Fig. 1. Network topology.

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.

The above mechanism conceals a danger that a request to switch into the fairing mode might affect an unnecessarily large portion of the network. Consider the following example in the topology shown in Figure 1, where a through h denote hosts while A, B, C, and D are routers. Flows f_1 and f_2 traverse paths $gCDh$ and $aABCDd$ respectively. Then, flow f_3 starts and traverses path $eABf$. After the new flow requests the fairing mode, router A sets the fairing bit in packets of flow f_2, and this causes router C to set the fairing bit in packets of flow f_1. Consequently, all three flows start operating in the fairing mode, even though f_1 does not share any link with f_3.

To ensure that a request of the fairing mode affects only those flows that share bottleneck links with the requesting flow, MCP relies once again on explicit communication between hosts and routers and incorporates a *this-path bit*. Whenever the sender of a flow sets the fairing bit in a data packet, the sender also sets the this-path bit in the packet. Routers never change this-path bits of forwarded packets. Only if both the fairing and this-path bits are set in the header of an incoming packet, the router propagates the set fairing bit to other flows. This prevents chain reactions that multiply set fairing bits needlessly.

E. 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 $[0.88; 0.98)$. Hence, MCP distinguishes between only the following four ranges of link utilization: $[0; 0.48)$, $[0.48; 0.88)$, $[0.88; 0.98)$, and $[0.98; \infty)$. Their respective two-bit encodings are 00, 01, 10, and 11. 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.

F. 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

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. 2. Modes of MCP operation

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. As we mention in Section IV-C, applying AD(80 kbps) is the only exception to using MD(0.5) for decrease. After the backtracking, MCP switches into a *stable mode*.

G. Stable mode

The stable mode is the ultimate regime of constant transmission prescribed by Principle 5. A flow stays in the stable state as long as $U \in [0.88; 0.98)$ and no flow requests a switch into the fairing mode.

H. Timing of transmission adjustment

As per Principle 3, MCP prescribes uniform timing for transmission adjustment by all flows. According to the control theory, the sender of a flow should adjust its transmission rate only after the feedback reflects the impact of the previous adjustment on the network [24]. In our design, the sender increases its transmission once per $2T$, where T is the period of link-utilization measurement. Since the default value of T is 200 ms, each flow regardless of its RTT raises its transmission rate once per 400 ms. As soon as the feedback indicates nascent overload, the sender immediately curbs the transmission and restarts its transmission-adjustment timer. The reaction to sustained overload is at the standard pace of one decrease per $2T$.

The initial transmission rate for every flow is 80 kbps. To translate the current transmission rate into a specific schedule of packet transmission, MCP distributes the transmitted packets uniformly during any inter-adjustment interval. After starting transmission of a packet of size S bytes, the sender schedules transmission of the next packet for $\frac{8S}{R}$ seconds later, where R denotes the current transmission rate in bps. Whenever the sender adjusts R (i.e., when the transmission-adjustment timer expires, or when the feedback indicates nascent overload), the sender also recalculates $\frac{8S}{R}$ and adjusts accordingly the remaining wait for the next packet.

I. Summary of MCP design

In this section, we summarize MCP design in terms of its explicit communication format, router, sender, and receiver operation.

1) 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.

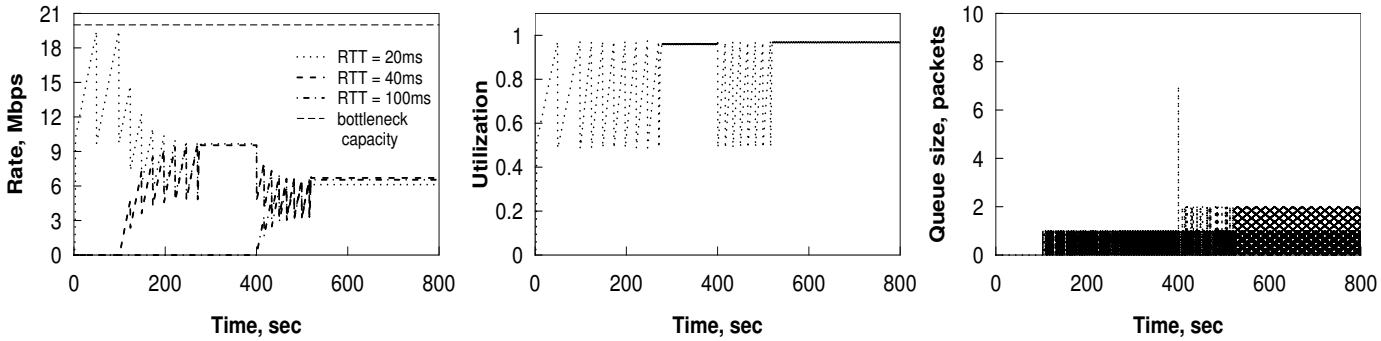
2) Router operation: Routers provide explicit feedback to senders through receivers. To form the feedback, each router periodically computes utilizations of its output links. Routers also set fairing bits in forwarded packets to disseminate to appropriate flows a request of operating in the fairing mode.

3) 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 2.

4) Receiver operation: The receiver sends an ACK packet for every incoming data packet. The ACK packet echoes the fairing bit and encoding of the the bottleneck link utilization.

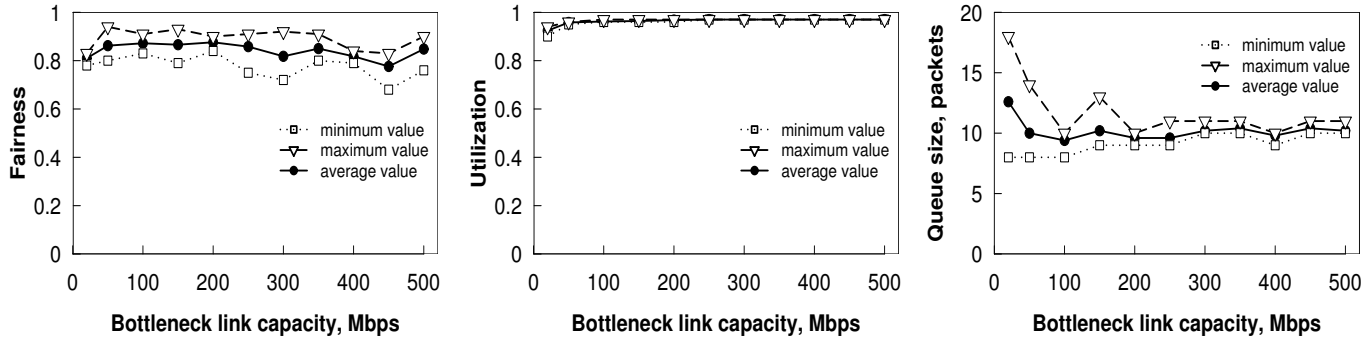
V. Experimental evaluation

In this section, we report our simulations conducted in version 2.29 of ns-2 [25] 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, and the capacities of non-bottleneck access links are 40 Mbps. 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. We conduct three types of simulations. Simulations of the first kind illustrate how MCP behaves, e.g., with respect to fairness convergence and bottleneck link utilization. The second class of simulations studies characteristics of MCP as functions of different network parameters. In these experiments, we vary a single parameter while keeping all the other parameters fixed. For every value of the parameter, we conduct 5 simulations and report the minimum, average, and maximum values of each performance metric in the simulations. The third type of our experiments compares MCP with the existed protocols TCP and VCP.



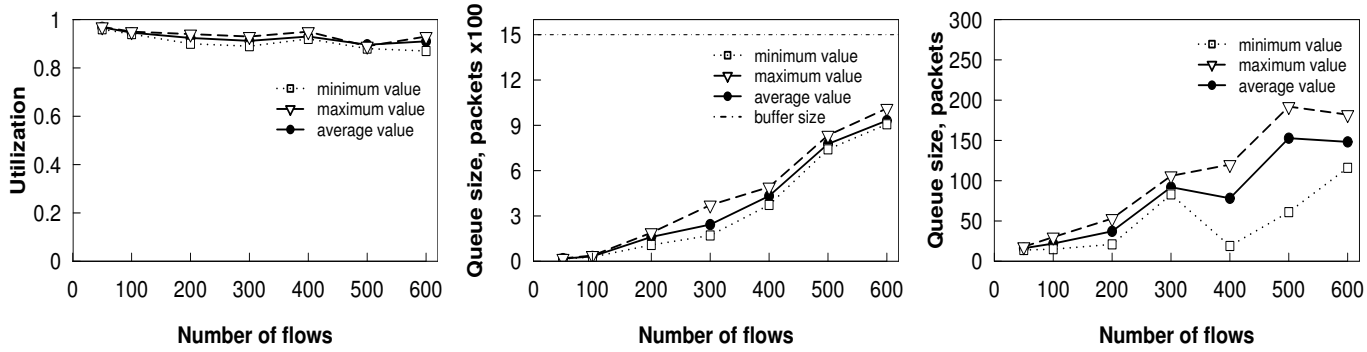
(a) (b) (c)

Fig. 3. MCP convergence to fair high utilization and low queue size at the bottleneck link: (a) transmission rates of the three flows, (b) utilization of the bottleneck link, and (c) queue size at the bottleneck link.



(a) (b) (c) link

Fig. 4. Low sensitivity to the bottleneck link capacity: (a) fairness index, (b) utilization of the bottleneck link, and (c) peak queue size at the bottleneck link.



(a) (b) (c)

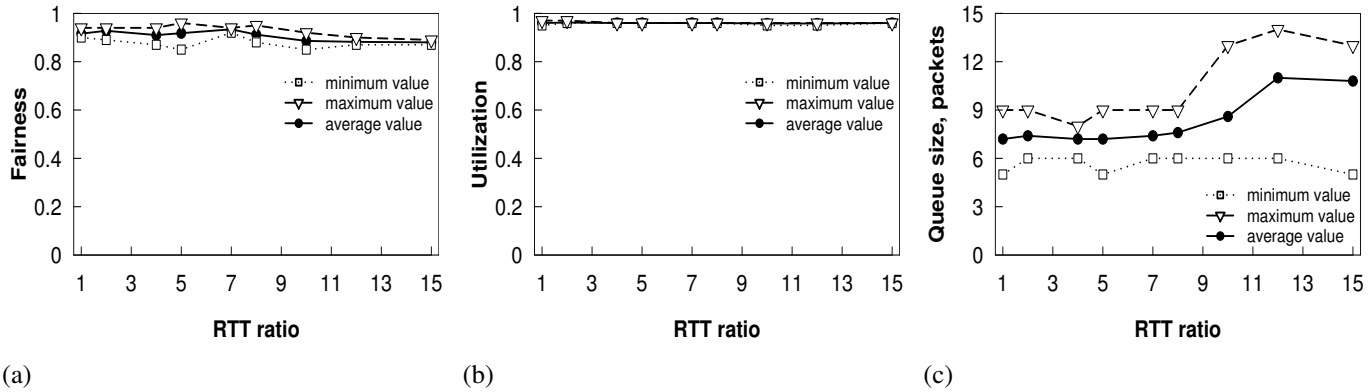
Fig. 5. Dependence on the number of flows: (a) utilization of the bottleneck link, (b) peak queue size at the bottleneck link, and (c) peak queue size at the bottleneck link in the stable state.

A. Convergent behavior

To illustrate the convergent behavior of MCP, we conduct an experiment with 3 flows that have propagation RTT of 20 ms, 40 ms, and 100 ms respectively. The network topology is a single-bottleneck dumbbell with the bottleneck link capacity of 20 Mbps. The duration of the experiment is 800 sec. Figure 3 reports the experimental results. The queue size at the bottleneck link never exceeds 7 packets. All three flows converge to stable transmission. In the stable state, the fairness index is 0.91, and the bottleneck link utilization reaches the high 0.97.

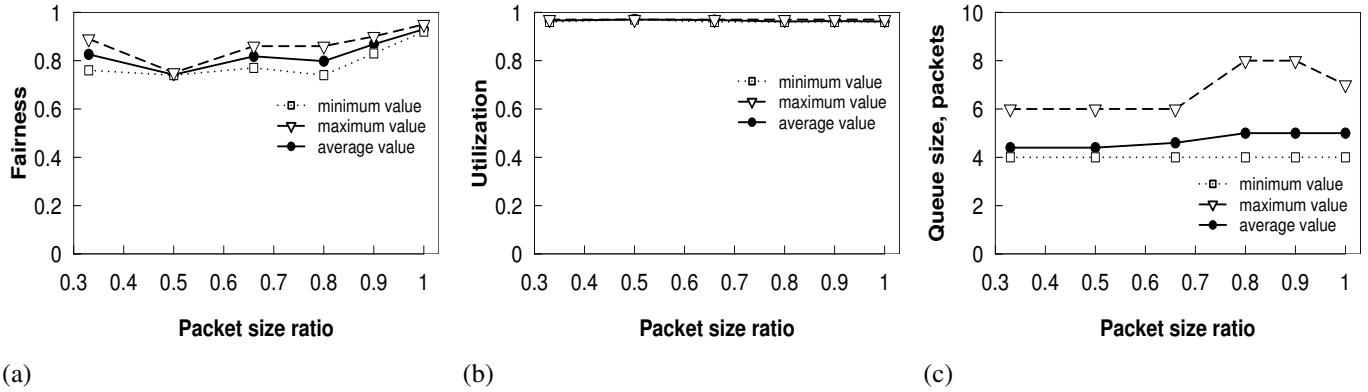
B. Influence of the bottleneck link capacity

To study the impact of the bottleneck link capacity on MCP operation, we perform simulations with 20 flows. Propagation RTTs of the flows are uniformly distributed over the range from 20 to 100 ms. Each simulation lasts for 480 sec. The network topology is a single-bottleneck dumbbell where the



(a) (b) (c)

Fig. 6. Independence of MCP stable-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.



(a) (b) (c)

Fig. 7. Independence of MCP stable-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.

bottleneck link capacity varies from 20 to 500 Mbps, and the capacities of non-bottleneck access links are scaled up proportionally. Arrival times of the flows are selected randomly from the interval between 0 and 10 sec. Figure 4 shows that the stable-state fairness, bottleneck link utilization, and peak queue size are relatively insensitive to the bottleneck link capacity: the average fairness index is between 0.776 and 0.886; the average link utilization lies is the range from 0.924 to 0.970; the peak queue size falls between 9.4 and 12.6 packets.

C. Dependence on the number of flows

We investigate MCP performance with different numbers of flows in the single-bottleneck dumbbell topology where the bottleneck link has capacity 200 Mbps and propagation delay 24 ms while every access link has capacity 400 Mbps and propagation delay 3 ms. Hence, propagation RTT for each flow is 60 ms. We vary the number of flows from 50 to 600. Arrival times of the flows are randomly chosen from the interval between 0 and 10 sec. Figure 5 depicts how the number of flows affects the bottleneck link utilization and queue size. While the average link utilization remains relatively stable (it varies between 0.896 and 0.968), the queue size at the bottleneck link is significantly more sensitive to the flow count. As the number of flows increases, the average peak queue size rises from 17.8 to 931.4 packets, and the average peak queue size in the stable state grows from 6.2 to 152.8 packets. Although the buffer consumption per flow is less than 1.6 packets in general and less than 0.26 packets in the stable state, MCP design needs further improvements in order to maintain a low overall queue size regardless of the number of flows. Figure 5b also shows that MCP prevents packet losses in all the conducted simulations.

D. Heterogeneous RTTs

To evaluate MCP under different RTTs, we conduct simulations in the single-bottleneck dumbbell topology with 10 competing flows. We 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}. Arrival times for the flows are randomly picked from the range between 0 and 10 sec. The complete duration of the simulation is 120 sec. Figure 6 confirms that MCP stable-state operation is relatively independent of RTT heterogeneity: the fairness index varies from 0.880 to 0.934; the bottleneck link utilization lies in the range between 0.958 and 0.962; the average peak queue size

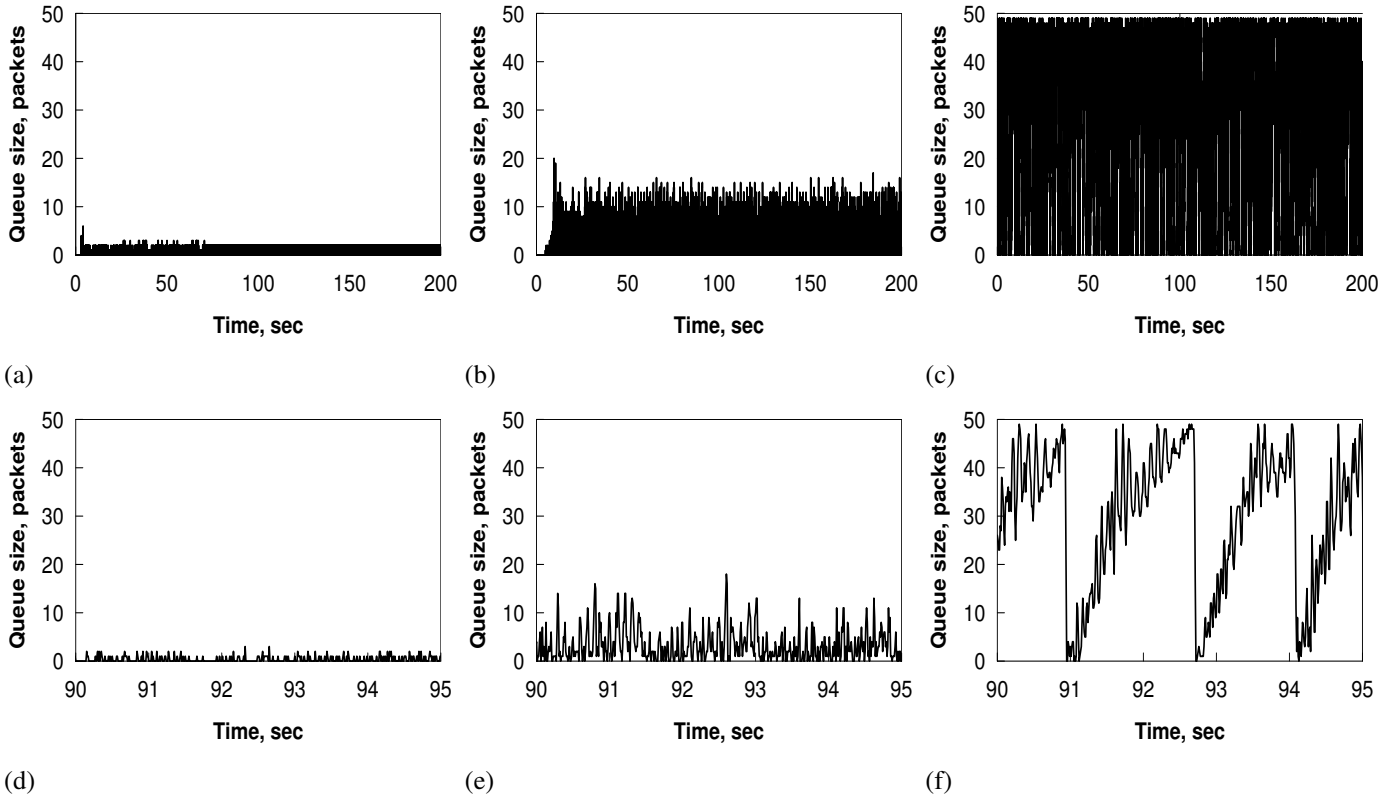


Fig. 8. Queuing at the bottleneck link: (a) under MCP, (b) under VCP, (c) under TCP, (d) zoomed look at MCP queuing, (e) zoomed look at VCP queuing, and (f) zoomed look at TCP queuing.

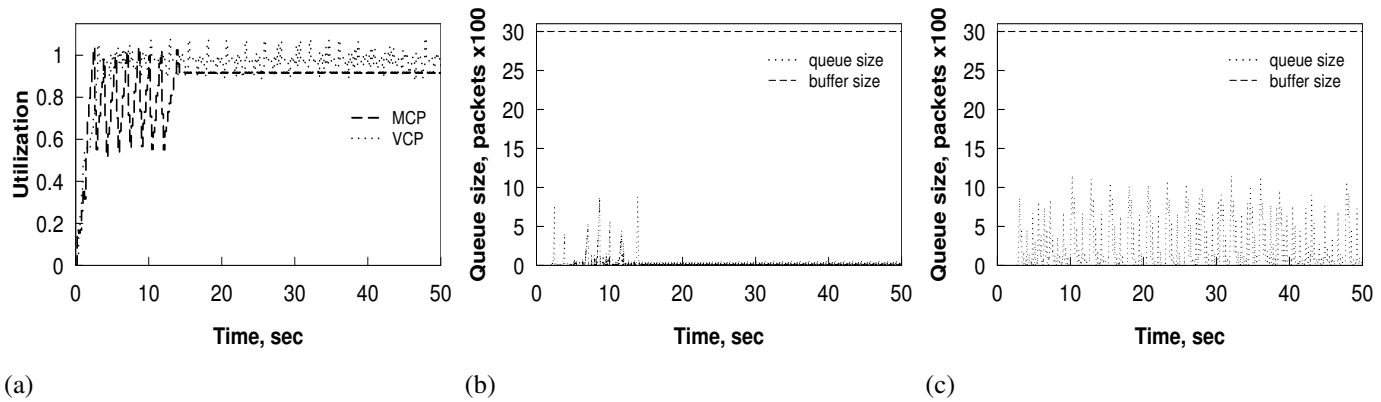


Fig. 9. MCP versus VCP: (a) utilization of the bottleneck link, (b) queue size at the bottleneck link under MCP, and (c) queue size at the bottleneck link under VCP.

at the bottleneck link changes only slightly from 7.2 to 11 packets.

E. Heterogeneous packet sizes

We also examine the impact of packet-size heterogeneity. The topology is a single-bottleneck dumbbell with 5 flows. Arrival times of the flows are randomly chosen from the range between 0 and 5 sec. Each experiment lasts 200 sec. Propagation RTTs of the flows are uniformly distributed 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. Other packet sizes are uniformly distributed between S_{min} and S_{max}. Figure 7 shows relative immunity of MCP stable-state operation to the heterogeneous packet sizes: the fairness index stays in the range from 0.742 to 0.930; the bottleneck link utilization varies from 0.962 to 0.970; the average peak queue size is almost constant, between 4.4 and 5.0 packets.

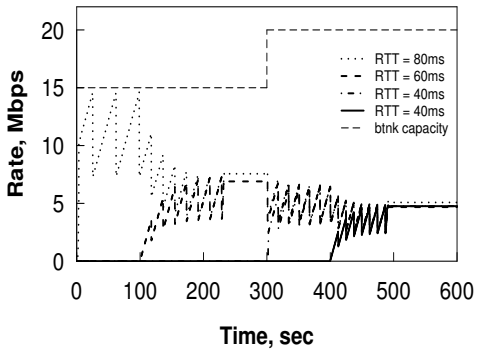


Fig. 10. Transmission rates of MCP flows converge to fair rates after the bottleneck link migrates.

F. Comparison of MCP with TCP and VCP

First, we compare MCP, TCP, and VCP in terms of queuing at the bottleneck link. In each simulation of this series, the single-bottleneck dumbbell serves 5 flows that have propagation RTT of 20, 40, 60, 80, and 100 ms. The bottleneck link has capacity 20 Mbps and buffer for 50 packets of size 1000 bytes. The flows arrive 1 sec after each other. The simulation duration is 200 sec. We repeat the experiment for each of the three protocols. Figure 8 plots the queue size at the bottleneck link for two time scales: the whole experiment duration and zoomed interval between 90 and 95 sec. The graphs show that MCP significantly subdues the queuing: the peak queue size under TCP is 50 packets, under VCP is 20 packets, and under MCP is 6 packets. In the stable state of the above experiments, the bottleneck link utilization is 0.92 under MCP and oscillates between 0.89 and 1 under VCP.

We conduct an additional comparative evaluation of MCP and VCP in the single-bottleneck dumbbell topology where the bottleneck link capacity is 500 Mbps, and the capacities of the access links are 1 Gbps. In both experiments which last 50 sec, 1000 flows with propagation RTT 60 ms arrive at randomized moments between 0 and 1 sec. Figure 9 traces the bottleneck link utilization and queue size under each protocol. The peak queue size under MCP is 31% of the buffer size and close to the 39% under VCP. However, while the peak queue size under VCP stays at the same level in the stable state, MCP reduces its peak queue size in the stable state dramatically to 1.7% of the buffer size.

G. Migrating bottleneck links

To investigate MCP operation in scenarios with multiple and migrating bottleneck links, we conduct an experiment in the parking-lot topology shown in Figure 1. The simulation lasts 600 sec and involves 4 flows with propagation RTT of 80, 60, 40, and 40 ms: flow f_1 arrives at time 0 and traverses path $aABCDd$, flow f_2 arrives at time 100 sec and traverses path $eABCDh$, flow f_3 arrives at time 300 sec and traverses path $bBCc$, and flow f_4 arrives at time 400 sec and traverses path $fBCg$. Flows f_1 and f_2 are bottlenecked at link AB before time 300 sec but the bottleneck migrates to link BC when flow f_3 arrives. Figure 10 shows that despite the migration of the bottleneck link, MCP always enables all the flows to converge to fair transmission rates.

VI. Summary and discussion

In this paper, we developed 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 shows that the seven-cycle longevity of the fairing mode is sufficient to provide high levels of 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 stable 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 TCP and VCP show that, by and large, MCP meets its design objectives. The only 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. To conclude, we discuss the following two concerns about MCP design:

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

[3] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, June 1993.

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

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

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

[7] G. Raina, D. Towsley, and D. Wischik, "Part II: Control Theory for Buffer Sizing," *ACM Computer Communication Review*, vol. 35, no. 3, pp. 79–82, July 2005.

[8] D. Wischik and N. McKeown, "Part I: Buffer Sizes for Core Routers," *ACM Computer Communication Review*, vol. 35, no. 3, pp. 75–78, July 2005.

[9] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Part III: Routers with Very Small Buffers," *ACM Computer Communication Review*, vol. 35, no. 3, pp. 83–90, July 2005.

[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," University of Massachusetts Amherst, Tech. Rep. CMPSCI 06-14, April 2006.

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

[14] 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.

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

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

[17] 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.

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

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

[20] University of Southern California, "DoD Standard Internet Protocol," RFC 760, January 1980.

[21] 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.

[22] V. Paxson, "End-to-End Internet Packet Dynamics," in *Proceedings ACM SIGCOMM 1997*, September 1997.

[23] 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, July 1989.

[24] K. Ramakrishnan and R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer," in *Proceedings ACM SIGCOMM 1988*, August 1988.

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

APPENDIX

In this section, we review the classical Chiu-Jain model and then use it to analyze for how long MCP flows should operate in the fairing mode.

A. Chiu-Jain model

In Chiu-Jain model [23], n distributed users share a single resource that has a target load C. The model is synchronous and employs a discrete timescale. Every instant on the timescale represents a moment when all the users adjust their loads on the resource. At time t, user i imposes a positive real load $l_i(t)$. Vector $\vec{l}(t) = (l_1(t), l_2(t), \ldots, l_n(t))$ captures all individual loads. The total load of the users is $L(t) = \sum_{i=1}^{n} l_i(t)$. By time t, the system provides all users with a uniform binary feedback

$$f(t) = \begin{cases} 0 & \text{if } L(t-1) \leq C, \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

that indicates whether the total load of the users after the previous round of adjustments exceeds the target load.

Chiu and Jain applied their model to analyze behavior of Additive-Increase Multiplicative-Decrease (AIMD) algorithms that change the load of each user i as follows:

$$l_i(t) = \begin{cases} l_i(t-1) + x & \text{if } f(t) = 0, \\ y l_i(t-1) & \text{otherwise} \end{cases} \tag{4}$$

where coefficients x and y are constants such that $x > 0$ and $0 \leq y < 1$. We refer to a specific adjustment algorithm within the AIMD class as AIMD(x; y) where x and y represent respectively the AI and MD coefficients of the algorithm. Chiu and Jain showed that the load of every user under an AIMD algorithm converges from any initial state toward the efficient fair state where the load of each user is $\frac{C}{n}$. The fairness index increases monotonically and converges to 1.

B. Sufficient longevity of the fairing mode

To reason about fairness improvement under AIMD(x; 0.5) after the total load reaches the target load, we define a notion of an *increase-decrease cycle* as a series of adjustments between two peaks of the oscillating load. If the increase step is sizable in comparison to the fair share, each increase-decrease cycle improves fairness significantly. Hence, we focus on more challenging scenarios where x is small with respect to $\frac{C}{n}$ and assume that the extent of overshooting the target load is always

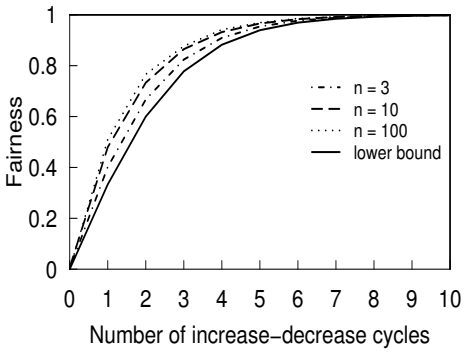


Fig. 11. Speed of fairness convergence under AIMD(x; 0.5).

negligible. Then, since decrease with factor 2 releases one half of the target load, each increase-decrease cycle contains

$$m = \frac{C}{2nx} \tag{5}$$

increase steps. In considered cases, new users arrive to the system when existing users already utilize the resource efficiently and fairly. Also, we limit the analysis to settings where the new users do not outnumber the existing users. Then, the maximal individual load is initially at most twice the fair share:

$$l_{max}(t_0) \leq \frac{2C}{n} \tag{6}$$

where t_k denotes the k-th time the total load reaches the target load, and $l_{max}(t_k)$ represents the maximum individual load at time t_k. The k-th increase-decrease cycle transforms the maximal individual load into

$$l_{max}(t_k) = \frac{l_{max}(t_{k-1})}{2} + \frac{C}{2n}. \tag{7}$$

After $k-1$ increase-decrease cycles, the maximal individual load reduces by time t_{k-1} to

$$l_{max}(t_{k-1}) = \frac{l_{max}(t_0)}{2^{k-1}} + \frac{C}{n}\left(1 - \frac{1}{2^{k-1}}\right). \tag{8}$$

Taking into account Inequality 6, we derive

$$l_{max}(t_{k-1}) \leq \frac{C}{n}\left(1 + \frac{1}{2^{k-1}}\right). \tag{9}$$

From the definition of the fairing index, we derive that fairness after the k-th increase-decrease cycle of one decrease and m increases becomes

$$F(t_k) = F(t_{k-1}) + \frac{1 - F(t_{k-1})}{1 + \frac{l_{max}(t_{k-1})}{2mx}}. \tag{10}$$

Combining the above expression with Equation 5 and Inequality 9, we establish that the fairness index after the k-th increase-decrease cycle is bounded from below as:

$$F(t_k) \geq F(t_{k-1}) + \frac{1 - F(t_{k-1})}{1 + \frac{\frac{C}{n}\left(1+\frac{1}{2^{k-1}}\right)}{2\frac{C}{2nx}x}}, \tag{11}$$

i.e.,

$$F(t_k) \geq F(t_{k-1}) + \frac{1 - F(t_{k-1})}{2 + 0.5^{k-1}}. \tag{12}$$

Since the fairness index is at least zero, combination of $F(t_0) = 0$ and Inequality 12 provides an approximate lower bound for fairness after an arbitrary number of increase-decrease cycles. Figure 11 plots this lower bound together with graphs of the fairness index under AIMD(80 kbps; 0.5) for different values of n in the system where the target load is $C = 100$ Mbps, one (new) user has initial load 80 kbps while the initial load of every other user is $\frac{C}{n-1}$. The line for $n = 2$ is indistinguishable from the lower bound. All the plotted graphs agree that seven increase-decrease cycles are sufficient to provide high fairness of at least 98%.