On the Design of Load Factor based Congestion Control Protocols for Next-Generation Networks[☆,☆☆]

Ihsan Ayyub Qazi[a,*], Taieb Znati[a]

[a]*Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260, USA*

Abstract

Load factor based congestion control schemes have shown to enhance network performance, in terms of utilization, packet loss and delay. In these schemes, using more accurate representation of network load levels is likely to lead to a more efficient way of communicating congestion information to hosts. Increasing the amount of congestion information, however, may end up adversely affecting the performance of the network. This paper focuses on this trade-off and addresses two important and challenging questions: (i) How many congestion levels should be represented by the feedback signal to provide near-optimal performance? and (ii) What window adjustment policies must be in place to ensure robustness in the face of congestion and achieve efficient and fair bandwidth allocations in high Bandwidth-Delay Product (BDP) networks, while keeping low queues and negligible packet drop rates?

Based on theoretical analysis and simulations, our results show that 3-bit feedback is sufficient for achieving near-optimal rate convergence to an efficient bandwidth allocation. While the performance gap between 2-bit and 3-bit schemes is large, gains follow the *law of diminishing returns* when more than 3 bits are used. Further, we show that using multiple back-off factors enables the protocol to adjust its fairness convergence rate, rate variations and responsiveness to congestion based on the degree of congestion at the bottleneck. Based on these insights, we design Multi-Level feedback Congestion control Protocol (MLCP). In addition to being efficient, MLCP converges to a fair bandwidth allocation in the presence of diverse RTT flows while maintaining near-zero packet drop rate and low persistent queue length. Using extensive packet-level simulations we show that the protocol is stable across a range of network scenarios. A fluid model for the protocol reinforces the stability properties that we observe in our simulations and provides a good theoretical grounding for MLCP.

[☆]An earlier version of the paper was presented at IEEE INFOCOM 2008, April 2008, Phoenix, AZ.

[☆☆]This work was supported by NSF under grant 05-010684.

[*]Corresponding author. Tel.: +1 412 624 8442; Fax: +1 412 624 8854

Email addresses: ihsan@cs.pitt.edu (Ihsan Ayyub Qazi), tznati@nsf.gov (Taieb Znati)

Keywords: Congestion Control, TCP, AQM, ECN, Load factor

1. Introduction

The congestion control algorithm in the Transmission Control Protocol (TCP) has been widely credited for the stability of the Internet. However, future trends in technology (*e.g.*, increases in link capacities [1] and incorporation of wireless WANs into the Internet), coupled with the need to support diverse QoS requirements, bring about challenges that are likely to become problematic for TCP. This is because (1) TCP reacts adversely to increases in bandwidth and delay and (2) TCP's throughput and delay variations makes it unsuitable for many real-time applications. These limitations may lead to the undesirable situation where most Internet traffic is not congestion-controlled; a condition that is likely to impact the stability of the Internet.

TCP was designed to suit an environment where the BDP was typically less than ten packets and any packet loss inside the network was assumed to be due to overflow of router buffers at the bottleneck [2]. These assumptions are no longer true today. BDP of many Internet paths is orders of magnitude larger and in networks such as wireless LANs and WANs, congestion is no longer the only source of packet loss; instead bit errors, hand-offs, multi-path fading etc account for a significant proportion of lost packets. A more fundamental problem with TCP (*e.g.*, Reno, NewReno, SACK) and its other variants (*e.g.*, Vegas, Fast) is the usage of packet loss and queueing delay as signals of congestion, respectively [3, 4, 5, 6]. Packet loss is a binary signal and so provides little information about the level of congestion at the bottleneck, while (forward path) queuing delay is hard to measure reliably. Moreover, loss and delay are important performance metrics; using them as signals of congestion implies that action can only be taken after performance has degraded.

To address these issues, researchers have proposed transport protocols that can be placed into three broad categories, (a) end-to-end (e2e) schemes with implicit feedback, (b) e2e schemes with explicit feedback and (c) network-based solutions. e2e schemes with implicit feedback treat the network as a black box and infer congestion via implicit signals such as loss and delay. Research studies have shown that using only packet loss and/or delay as a signal of congestion poses fundamental limitations in achieving high utilization and fairness while maintaining low bottleneck queue and near-zero packet drop rate on high BDP paths [4, 7]. The benefit of using such schemes is in the ease of their deployment because they require modifications only at the end-hosts. e2e schemes with explicit feedback (such as TCP+AQM/ECN proposals [8, 9, 10, 11, 12] and VCP [13]) use one or few bits of explicit feedback from the network, however, the bulk of their functionality still resides at the end-hosts. They typically require changes at the end-hosts with incremental support from the network. Such schemes have been shown to perform better than their counterparts with implicit feedback. However, it is unclear how the amount of congestion feedback information affects performance; a question we make an attempt to answer in

this work. In network-based schemes (*e.g.*, XCP [14], RCP [15]), fairness and congestion control are enforced inside the network, therefore, these schemes are likely to induce more overhead on routers. Moreover, such schemes require significant changes in the routers and end-hosts which makes their deployment difficult.

VCP is an e2e scheme that uses two bits of explicit feedback from the network. It is a generalization of the one-bit ECN that uses load factor (ratio of demand to capacity) as a signal of congestion [13]. However, VCP's rate of convergence to an efficient bandwidth allocation is far from optimal, which considerably increases the AFCT of short flows (see Section 2). VCP's usage of a single, fixed Multiplicative Decrease (MD) parameter reduces responsiveness to congestion in high load and causes slow convergence to fairness. Further, in the presence of diverse RTT flows, VCP becomes considerably unfair as shown by simulation results in Section 4. A closer look at the VCP analysis reveals that (1) more refined spectrum of congestion levels is necessary to avoid inefficiencies on high BDP paths, (2) The window adjustment policies in high load regions should adapt to the degree of congestion, to provide smooth rate variations and to ensure robustness in the face of congestion and (3) mechanisms should be in place to achieve good fairness while maintaining low queues in the presence of RTT heterogeneity. This, however, raises few fundamental questions about load factor based congestion control schemes: (i) *What representation of the network load provides the best trade-off between performance gains and the adverse effects due to the larger amount of feedback?* (ii) *What window increase/decrease policies must be in place to ensure efficient and fair bandwidth allocations in high BDP networks while keeping low queues and near-zero packet drop rate?* This paper addresses these issues and uses the insights gained by the analysis to design **M**ulti-**L**evel Feedback **C**ongestion Control **P**rotocol.

The theoretical analysis and simulations carried out as part of this work show that using 3-bit representation of the network load levels is sufficient for achieving near-optimal rate of convergence to an efficient bandwidth allocation. While the performance improvement of 3-bit over 2-bit schemes is large, the improvement follows the "*law of diminishing returns*" when more than three bits are used. Our results also show that using multiple levels of MD enables the protocol to adjust its rate of convergence to fairness, rate variations and responsiveness to congestion according to the degree of congestion at the bottleneck. Guided by these fundamental insights, we design MLCP, in which each router classifies the level of congestion in the network using four bits while employing *load factor* as a signal of congestion [16]. In addition, each router also computes the *mean RTT* of flows passing through it, to dynamically adjust its load factor measurement interval. These two pieces of information are tagged onto each outgoing packet using *only* seven bits. The receiver then echoes this information back to the sources via acknowledgment packets. Based on this feedback, each source applies one of the following window adjustment policies: Multiplicative Increase (MI), Additive Increase (AI), Inversely-proportional Increase (II) and Multiplicative Decrease (MD). MLCP like XCP decouples efficiency control and fairness control by applying MI to converge exponentially to an efficient

bandwidth allocation and then employing AI-II-MD control law for providing fairness among competing flows [14]. MLCP adjusts its aggressiveness according to the spare bandwidth and the feedback delay which prevents oscillations, provides stability in the face of high bandwidth or large delay, and ensures efficient utilization of network resources. Dynamic adaptation of the load factor measurement interval allows MLCP to achieve high fairness in the presence diverse RTT flows. In addition, MLCP decouples loss recovery from congestion control which facilitates distinguishing error losses from congestion-related losses; an important consideration in wireless environments. MLCP has low algorithmic complexity, similar to that of TCP and routers maintain no per-flow state.

Using extensive packet-level ns2 [17] simulations, we show that MLCP achieves high utilization, low persistent queue length, negligible packet drop rate and good fairness. We use an approximate fluid model to show that the proposed protocol is globally stable for any link capacity, feedback delay or number of sources for the case of a single bottleneck link shared by identical RTT flows. The model reinforces the stability properties that we observe in our simulations and provides a good theoretical grounding for MLCP.

The rest of the paper is organized as follows. In Section 2, we present the feedback analysis for determining the number of congestion levels. We describe the components of the protocol in Section 3. In Section 4, we evaluate the performance of MLCP using extensive packet-level simulations. Section 5 discusses the stability of MLCP using a fluid model. Section 6 discusses related work and Section 7 offers concluding thoughts and future work.

2. Feedback Analysis

Every protocol that uses load factor as a signal of congestion must consider three important issues (1) *How many bits to use for carrying load-factor information?* (2) *What transition points to choose for each symbol?* (3) *What actions should end-hosts take based on the received signal?* In this section, we address these issues in detail.

The number of bits used in representing the feedback signal impacts the preciseness of congestion information. This, in turn, determines how conservative a source may need to be in order to compensate for the loss of information. However, having large number of bits in representing load-factor information is not necessarily desirable. On one hand, increasing the number of bits is likely to increase the overhead caused by the need to process and respond to different levels of congestion. On the other hand, it leads to a more precise estimation of the level of congestion and, therefore, a more accurate response from the sources. Hence, the goal is to determine the number of congestion levels that provide the best trade-off between performance improvements and the number of bits used in the feedback.

The performance metrics likely to be affected by the preciseness of the feedback signal include (1) rate of convergence to high utilization and (2) rate of convergence to fairness. The analysis of these metrics is used to derive the *optimal* number of congestion levels.

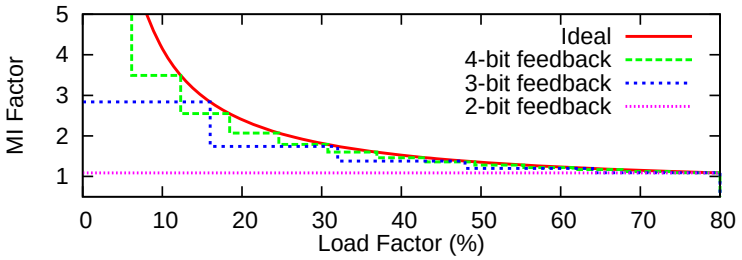

Figure 1: Comparison of MI factors of the ideal protocol with 2-bit, 3-bit and 4-bit feedback schemes

2.1. ***Rate of Convergence to High Utilization***

Window-based congestion control protocols often use MI to converge exponentially to high utilization. However, *stable* protocols often require the magnitude of the MI factor to be proportional to the available bandwidth at the bottleneck [14, 13]. In the context of load-factor based congestion control protocols, this translates into requiring the MI factor to be proportional to $1 - \sigma$, where σ is the load factor at the bottleneck. We, therefore, define the MI gain function of the ideal, stable, load factor based congestion control protocol as follows.

$$\xi(\sigma) = \kappa \cdot \frac{1 - \sigma}{\sigma} \tag{1}$$

where $\kappa = 0.35$ is a stability constant. The stability result presented in Section 5 shows that congestion control protocols whose MI gains are upper-bounded by the above function, are indeed stable. It should be noted that the actual MI factor is given by $1 + \xi(\sigma)$ [13].

Figure 1 shows the MI factors used by the ideal protocol along with 2-bit, 3-bit and 4-bit feedback schemes. The goal of the protocol designer is to closely match the MI gain curve of the ideal protocol using as few bits as possible. The more congestion levels the feedback signal represents, the more aggressive the sources can be due to higher MI factors. If the number of congestion levels is small, sources would have to make a conservative assumption about the actual load factor value at the bottleneck, forcing them to use small MI gains. To compare the performance of schemes using different representations of the network load levels, we examine their speed of convergence for achieving efficient bandwidth allocations.

To quantify the speed of convergence, we compute the time required to achieve a given target utilization $U_t \in [0, 1]$ (80% in our case). When the system utilization, U_s, is less than U_t, each flow applies MI with a factor that depends on (1) l, the number of congestion levels used by the scheme and (2) the load factor interval (or utilization region) in which the system is operating. Suppose that a given scheme divides the target utilization region (*i.e.*, $[0, U_t]$) into $l_0, l_1, l_2, .., l_l$ levels, where $l_0 = 0$, the size of each interval $[l_{i-1}, l_i]$ (referred

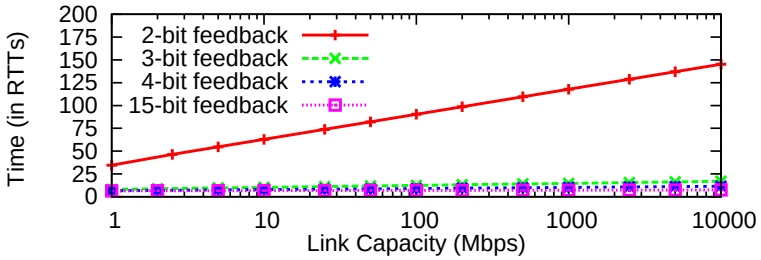

Figure 2: Comparison of the time required to achieve 80% utilization for 2-bit, 3-bit, 4-bit and 15-bit feedback schemes.

to as interval i) is $s = U_t/l$ and $l_i = l_{i-1} + s$. The MI factor applied during interval i is given by $m_i = 1 + \xi(l_i)$. Note that the upper limit of an interval determines the MI factor. The reason is when $U_s \in [l_{i-1}, l_i]$, l_i is an upper-bound on system utilization and since U_s can lie anywhere in the interval, a flow must assume it to be l_i to avoid using a larger MI factor than allowed by Eq. 1.

Consider a single flow with an initial congestion window size of x_0 KB. Suppose that the BDP of the path of the flow is $k = C \cdot RTT$ and the system utilization is l_{i-1}. When the system utilization becomes l_i, the congestion window of a flow must be equal to $x_i = k \cdot l_i$, $\forall i \geq 1$. Therefore,

$$x_{i-1} \cdot (m_i)^{r_i} = x_i \tag{2}$$

where r_i is the number of RTTs required to achieve utilization l_i given that the system started at l_{i-1}. This implies that the amount of time required to complete interval i is

$$r_i = log_{m_i}(x_i/x_{i-1}). \tag{3}$$

Thus, for a flow with an initial congestion window size of x_0 KB, it would take

$$r(l) = \sum_{i=1}^{l} r_i = \sum_{i=1}^{l} log_{m_i}(x_i/x_{i-1}) \tag{4}$$

RTTs to attain a system utilization equal to U_t, where $r(l)$ is the total time required to achieve the target utilization by a scheme that uses l congestion levels. We assume that a protocol using n bits for achieving efficient bandwidth allocations employs $l = 2^n - 3$ levels for representing the target utilization region.[1] The rest of the symbols are used for representing load factor values in $(U_t, 100)$. The analysis for determining the number of levels to represent the overload region (*i.e.*, $\sigma \in [100\%, \infty)$) is presented in Section 2.2.

[1]One symbol (*i.e.*, code $(00)_2$) is reserved for ECN-unaware source hosts to signal "not-ECN-capable-transport" to ECN-capable routers, which is needed for incremental deployment [10].

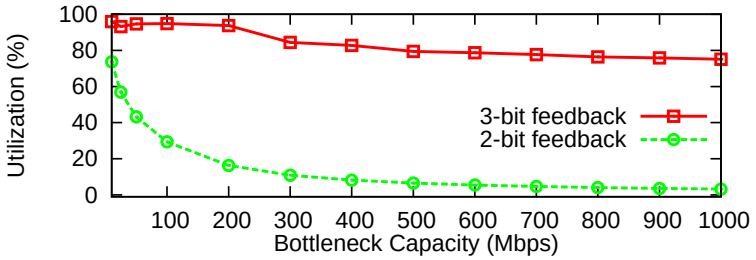

Figure 3: The figure shows the bottleneck utilization at t=10s as a function of link capacity for the 2-bit and 3-bit feedback schemes.

Consider a single flow traversing a 1 Gbps link with RTT=200 ms and an initial congestion window size of 1 KB. The above analysis implies that in order to achieve a target utilization of 80%, the 2-bit scheme would take roughly $r(1) = 118$ RTTs, the 3-bit scheme would take $r(5) = 15$ RTTs, the 4-bit scheme would take $r(13) = 11$ RTTs and the 15-bit scheme (an approximation to the ideal scheme with infinite congestion levels) would take about $r(32765) = 8$ RTTs. Figure 2 shows the time taken by different schemes to achieve $U_t = 80\%$ as a function of the bottleneck capacity. Observe the dramatic decline in time when n is increased from 2 to 3. However, as n is increased beyond 3, the gain in performance is very little and remains largely unaffected by the bottleneck capacity. Thus for $n \geq 3$, performance improvement follows the *law of diminishing returns*. Intuitively, this happens because increasing the number of bits beyond three only helps a small portion of the target utilization region (<10%, see Figure 1). Since the time taken by a flow to attain 10% utilization is a small component of the total time required by a flow to achieve the target utilization, increasing n has little impact on performance. To validate our results, we ran ns2 simulations. Figure 3 shows the bottleneck utilization at time $t = 10$ s for protocols employing 2-bit and 3-bit feedback signals. The 3-bit protocol is able to achieve 80% utilization within the first 10 seconds across link capacities ranging from 1 Mbps to 10 Gbps, whereas, for the 2-bit protocol, utilization falls significantly as link capacity is increased.

Impact on the AFCT of flows: An important user perceivable metric is the Average Flow Completion Time (AFCT) (or response time). The 3-bit feedback scheme considerably reduces the AFCT due to its higher rate of convergence to efficiency. In particular, it helps short flows to finish much quicker and since most flows on the Internet are short, this improvement impacts the majority of flows [15].

Let r_2 and r_3 be the AFCT corresponding to 2-bit and 3-bit feedback schemes, respectively. The improvement in AFCT is expressed as $(1-r_3/r_2)100\%$. Figure 4 shows the improvement in AFCT that the 3-bit scheme brings over the 2-bit scheme as a function of the average file size on a 10 Mbps and 100 Mbps link with RTT=100 ms. The file sizes obey the Pareto distribution with a shape

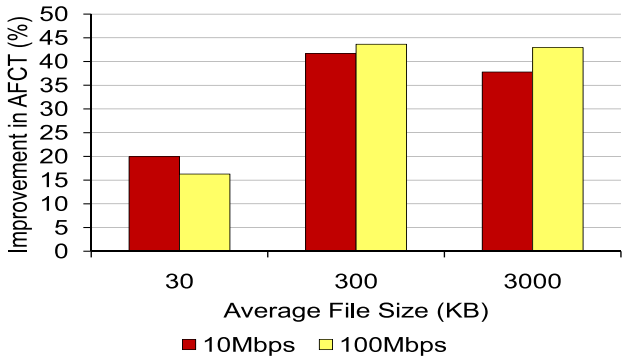

Figure 4: Improvement in AFCT that the 3-bit feedback scheme brings over the 2-bit feedback scheme as a function of the average file size on a 10 Mbps and 100 Mbps link

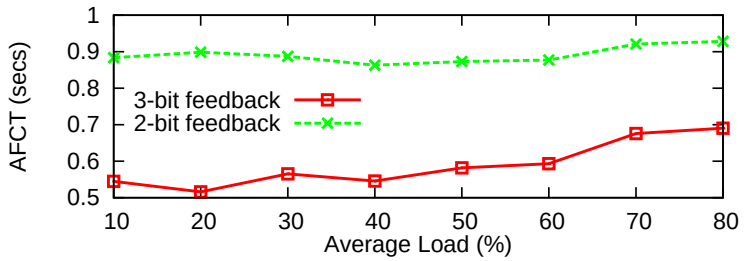

Figure 5: AFCT of flows as a function of load on a 10 Mbps link with RTT=100 ms.

parameter of 1.2 and the offered load was kept at 0.7. *Note that the 3-bit feedback scheme offers a reduction in AFCT of at least 16% and up to 45% over the 2-bit scheme.* Figure 5 shows the AFCT as a function of the average load at the bottleneck assuming the average file size to be 30 KB. *Observe that for average loads less than 50%, the 3-bit scheme improves the AFCT by a factor of ∼1.8. However, as the average load increases, the improvement reduces to a factor of ∼1.4.*

2.2. ***Rate of Convergence to a Fair Share***

Once high utilization is achieved, the goal of the protocol is to converge to a fair bandwidth allocation, often using control laws such as AIMD, AI-II-MD etc. While achieving this end, a protocol should aim to satisfy three requirements: (a) high convergence rate, (b) smooth rate variations, and (c) high responsiveness to congestion. These requirements, however, cannot be satisfied in all network scenarios. For instance, in some cases, maintaining high responsiveness to congestion may necessarily require significant variations in the rates of flows. However, one can isolate cases in which one or two of the

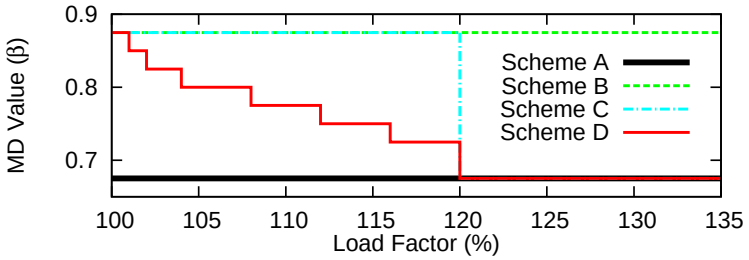

Figure 6: β as a function of load factor for different schemes

requirements are more desirable than the rest, allowing the protocol to focus on few *complimentary* goals. These cases are as follows:

- When the system is in equilibrium (*i.e.*, all flows have achieved their fair rates), the goal is to ensure (b) while (a) and (c) are not relevant.
- When new flows arrive, (a) and (c) are more important than (b).

A load factor based congestion control protocol may not be able to exactly discern between these cases. However, load factor values in the overload region ($> 100\%$) can provide for approximately identifying the above cases. The reason is that, for a fixed number of flows, the average overload remains the same. It only changes when a new flow arrives or an old flow leaves the network.

2.2.1. ***Quantifying rate of convergence to fairness and the smoothness properties of a scheme***

In order to quantify rate of convergence to fairness, we measure the time taken by a newly arriving flows to achieve 70% of its fair rate. For ease of comparison, we normalize the convergence time of all schemes by the convergence time of the best scheme. We call this quantity the *convergence ratio*. The smoothness property of a scheme is determined by the MD parameter value, β. In particular, flows that reduce their congestion windows by a factor of β in overload experience throughput variations by a factor of $1 - \beta$.

2.2.2. ***Determining the MD levels***

Let a round be a single cycle between two overload events. The duration, d, of a round is determined by the increase policy and the value of β, whereas the number of rounds, p, needed for convergence to fairness is determined by β only [18].[2] Thus, a high value of β leads to slow convergence and reduces responsiveness to congestion. On the contrary, a low value of β, improves convergence and responsiveness but introduces large throughput variations.

[2]Note that if the value of β depends on the increase policy then both of these determine p

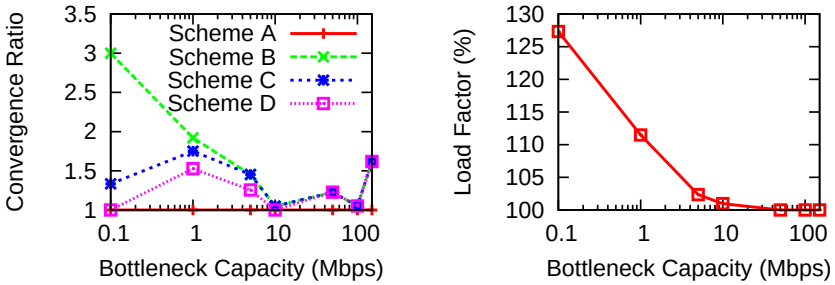

Figure 7: (a) Convergence ratio and (b) load factor in overload as a function of bottleneck capacity. N=2 flows, RTT=100 ms.

It is important to note that when a link is highly loaded (*i.e.*, σ is high in overload), responsiveness and convergence are more important goals than maintaining small rate variations. In order to achieve this end, we vary $\beta \in [\beta_{min}, \beta_{max}]$ with $\sigma \in [100\%, \sigma_{max}\%]$, where σ_{max} is the maximum value of σ after which sources apply β_{min}, and β_{min} and β_{max} are the minimum and maximum β values that can be used by sources. Two important factors must be considered when choosing these values. First, the minimum β value should be large enough to prevent the system from entering MI after MD because applying MI immediately after MD leads to high packet loss rate. In order to ensure this, note $\min(\sigma\beta(\sigma)) \geq 0.8$. Since for $\sigma \geq \sigma_{max}$, the smallest β is applied, therefore, β should be at least 2/3 (for $\sigma_{max} = 1.2$). We, therefore, set β_{min} to 0.675. Second, the maximum β value should be strictly less than 1 to allow for high rate of convergence to fairness. We set β_{max} to 0.875. Note that this choice ensures that for 70% convergence only nine congestion rounds are needed [18].

We now compare the performance of the following schemes:

- Scheme A uses a single β value of 0.675,
- Scheme B uses a single β value of 0.875,
- Scheme C uses two levels of MD depending on σ in overload, and
- Scheme D uses eight levels of MD as shown in Figure 6.

Varying Bottleneck Capacity: We start a single, long-lived MLCP flow at time $t = 0$. A new flow is started at $t = 50\,s$.[3] We vary bottleneck capacity and measure the convergence ratio for the second flow for each of the schemes. Figure 7 shows the convergence ratio of the four schemes along with the average

[3]Note that an inter-arrival time of 50 s ensures that the first flow is able to saturate the link before a new flow arrives

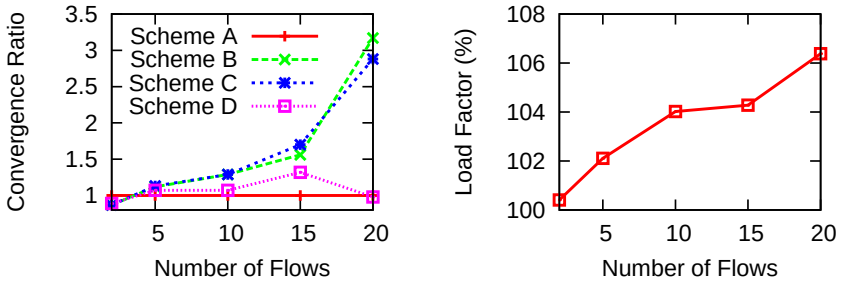

Figure 8: (a) Convergence ratio and (b) load factor in overload as a function of the number of flows. C=20 Mbps, RTT=100 ms.

load factor in overload, $\tilde{\sigma}$, at the bottleneck.[4] Observe that scheme A has the highest rate of convergence to fairness across a range of link capacities, followed by scheme D. However, scheme A also introduces the largest amount of variation in the throughput of flows; a characteristic that is highly undesirable for real-time and multimedia applications [19]. Scheme D, on the other hand, adapts β with $\tilde{\sigma}$. When C is small, scheme D applies a small β value (since $\tilde{\sigma}$ is high in overload). As the average overload decreases, scheme D increases the β value, therefore, reducing the rate variations. Scheme B takes the longest time to converge to a fair bandwidth allocation across a range of link capacities. This is due to the usage of a fixed, high β value. Scheme C improves upon the performance of scheme B. However, since it uses only two levels for representing overload, it is less aggressive than scheme D when $100\% \leq \tilde{\sigma} < 120\%$. Note that for link capacities exceeding 10 Mbps, average load factor values remain very close to 100%. This causes the sources to apply the maximum β value as allowed by each scheme; resulting in similar convergence ratios. However, this is only true for the case of two flows. Next, we show how each scheme performs when the number of flows are increased.

Varying Number of Flows: We now vary the number of long-lived flows on a bottleneck with capacity 20 Mbps. For these experiments, $N-1$ flows are started at time $t = 0$ and flow N is started at $t = 50\,s$. We measure the convergence ratio for flow N. As the number of flows increases, average overload increases roughly linearly (see Figure 8). Observe that while the convergence ratios achieved by schemes A and D are similar, schemes B and C take a much longer time to converge. For twenty flows, schemes A and D have convergence times that are at least three times smaller than that of schemes B and C. The reason is that these two schemes apply a β of value 0.875, which leads to slower

[4]The convergence time of scheme A is used as the normalizing factor because it has the highest convergence rate

convergence.[5] Scheme D changes β dynamically with σ and therefore achieves the right tradeoff between convergence and rate variations. Based on these insights, we use eight levels for representing the overload region.

2.2.3. ***Determining the Increase Policy***

The increase policy indirectly impacts (a) and (b). A large increase per RTT causes (i) MD to be applied more often and (ii) a smaller β to be applied by the end hosts, leading to fast convergence but increased oscillations. On the other hand, small increase per RTT enables existing flows to sustain their bandwidth for a longer time. However, it may lead to slow convergence. In order to achieve the benefits of these two strategies, we employ the AI-II-MD control law. When $80\% < \sigma \leq 95\%$, AI is used and for $95\% \leq \sigma < 100\%$, II is employed. AI ensures that flows quickly achieve high sending rates especially on high BDP paths, whereas II helps flows in sustaining their sending rates for a longer period of time. Since, with II, flows increase inversely proportional to the square root of their window sizes, they cause mild increments in σ when in steady state and larger when new flows arrive that have small congestion window sizes.

3. Protocol

In this section, we describe the sender, receiver and router components of MLCP.

3.1. ***MLCP Sender: Control Laws***

3.1.1. ***Homogeneous RTT flows***

We first consider a link shared by homogeneous flows whose RTTs are equal to t_p, the load factor measurement interval. At any time t, a MLCP sender applies either MI, AI, II or MD, based on the value of the encoded load factor received from the network.

load factor region: 0-80% When the load factor at the bottleneck is below 80%, each MLCP sender applies load-factor guided MI. The MI factor applied at each transition point (*i.e.*, 16%, 32%, 48%, 64% and 80%) are shown in Fig. 1. This translates into the following window adjustment strategy:

$$\mathsf{MI}: cwnd(t + rtt) = cwnd(t) \times (1 + \xi(\sigma)) \tag{5}$$

where $\xi(\sigma) = \kappa \cdot \frac{1-\sigma}{\sigma}$, σ is the load factor and $\kappa = 0.35$.

load factor region: >80% When the system has achieved high utilization, senders use the AI-II-MD control law to converge to a fair share. Each sender, applies AI until σ becomes 95%, after which II is applied. When the system moves into the overload region ($\geq$100%), each sender applies MD. The

[5] Note that scheme C applies $\beta = 0.875$ because $\tilde{\sigma} < 120\%$ for $N \leq 20$

following equations describe these control laws in terms of congestion window adjustments:

$$\mathsf{AI}: cwnd(t + rtt) = cwnd(t) + \alpha \tag{6}$$

$$\mathsf{II}: cwnd(t + rtt) = cwnd(t) + \frac{\alpha}{\sqrt{cwnd(t)}} \tag{7}$$

$$\mathsf{MD}: cwnd(t + \delta t) = cwnd(t) \times \beta(\sigma) \tag{8}$$

where $rtt = t_p$, $\delta t \to 0$, $\alpha = 1.0$ and $0 < \beta(\sigma) < 1$. To avoid over reaction to the congestion signal, MD is applied only once per t_p interval.

3.1.2. ***Parameter scaling for Heterogeneous RTT flows***

So far, we considered the case where the competing flows had the same RTT, equal to t_p. We now consider the case of heterogeneous RTTs. To offset the impact of heterogeneity, we normalize the RTT of each flow with the common t_p value. This emulates the behaviour of all flows having an identical RTT equal to t_p, thus making the rate increases independent of the flows' RTTs. During an interval t_p, a flow with RTT value rtt increases by a factor of $(1+\xi_s)^{t_p/rtt}$ where ξ_s is the scaled parameter. To make the MI amount independent of a flow's RTT, $(1+\xi_s)^{t_p/rtt} = (1+\xi)$, which yields Eq.9. Similarly, the AI gain of a flow during a time interval t_p can be obtained by solving $1 + \alpha = 1 + (t_p/rtt)\alpha_s$. However, for II, we want the increase policy to depend only on the current congestion window size, while being independent of its RTT. Therefore, we apply the same parameter scaling for II as used for AI.

$$\text{For MI}: \ \xi_s = (1 + \xi)^{rtt/t_p} - 1, \tag{9}$$

$$\text{For AI and II}: \ \alpha_s = \alpha \cdot \left(\frac{rtt}{t_p}\right), \tag{10}$$

Scaling for fair rate allocation: The above RTT-based parameter scaling only ensures that the congestion windows of flows with different RTT converge to the same value in steady state. However, fairness cannot be guaranteed, since rate ($= cwnd/rtt$) is still inversely proportional to the RTT. We need an additional scaling of the α parameter to achieve a fair share. To illustrate this, consider the AI-II-MD control mechanism applied to two competing flows where each flow $i = (1, 2)$ uses a separate α_i parameter, but a common MD parameter β. At the end of the M-th congestion epoch that includes $n > 1$ rounds of AI, $m > 1$ rounds of II and one round of MD, we have:

$$c_i(M) = \beta \cdot (c_i(M-1) + n \cdot \alpha_i + m \cdot \frac{\alpha_i}{\sqrt{c_i(M-1)}}) \tag{11}$$

where $c_i(M)$ is the congestion window of flow i at the end of the M-th congestion epoch. Eventually, each flow i achieves a congestion window that is proportional

to α_i. Indeed, the ratio of congestion window of the two flows approaches α_1/α_2 for large values of M. In order to see this, note that $c_1(M)/c_2(M)$ equals

$$
\begin{aligned}
&= \frac{c_1(M-1) + \alpha_1(n + \frac{m}{\sqrt{c_1(M-1)}})}{c_2(M-1) + \alpha_2(n + \frac{m}{\sqrt{c_2(M-1)}})} \\
&= \frac{\beta c_1(M-2) + \alpha_1(n + \beta n + \frac{m}{k_1} + \frac{\beta m}{\sqrt{c_1(M-2)}})}{\beta c_2(M-2) + \alpha_2(n + \beta n + \frac{m}{k_2} + \frac{\beta m}{\sqrt{c_2(M-2)}})} \\
&= \frac{\beta^2 c_1(M-3) + \alpha_1(n + \beta n + \beta^2 n + \frac{m}{a_1} + \frac{\beta m}{b_1} + \frac{\beta^2 m}{c_1})}{\beta^2 c_2(M-3) + \alpha_2(n + \beta n + \beta^2 n + \frac{m}{a_2} + \frac{\beta m}{b_2} + \frac{\beta^2 m}{c_2})}
\end{aligned}
$$

where $k_i = (\beta c_i(M-2) + \alpha_i \beta n/\sqrt{c_i(M-2)} + \alpha_i \beta n)^{1/2}$, $c_i = \sqrt{c_i(M-3)}$, $b_i = \sqrt{c_i(M-2)}$ and $a_i = k_i$ with $c_i(M-2)$ expanded to the next level. For $M = k$ the expression takes the same form as the above equation, the left operand of the addition operator becomes $\beta^{k-1} c_i(M-k)$ which approaches zero as k becomes large since $\beta < 1$. The multiplicative factor of α_i's can then be eliminated since they assume the same values. Hence, the above expression approaches α_1/α_2. Therefore, to allocate the bandwidth fairly among two flows, we scale the α parameter of each flow by its own RTT.

$$\alpha_f = \alpha_s \cdot \left(\frac{rtt}{t_p}\right) = \alpha \cdot \left(\frac{rtt}{t_p}\right)^2 \tag{12}$$

Note that VCP [13] uses similar parameter scaling but it employs the AIMD control law while MLCP uses the AI-II-MD control law.

3.2. ***MLCP Router***

A MLCP router performs two functions: (1) it computes the load factor over an interval t_p and (2) it estimates the average RTT of flows to adapt the load factor measurement interval.

3.2.1. ***Estimating the load factor***

There are two conflicting requirements that the value of a load factor measurement interval (*i.e.*, t_p) should aim to satisfy. First, it should be larger than the RTTs of most flows to factor out the burstiness induced by flows' responses. Second, it should be small enough to allow for robust responses to congestion and hence avoid queue buildup. A single value for t_p may not be suitable for meeting both the requirements since they depend on the RTT of flows which can vary significantly across Internet links. For example, in [13], a fixed value of t_p is used, which results in significant queue buildup due to the MI gains of large RTT flows. To keep low queues, they bound the MI gains of such flows, which in turn results in considerable unfairness as shown Section 4.4. For small RTT ($\ll t_p$) flows, a fixed t_p results in small MI and AI gains which can considerably increase the AFCT of flows. Indeed, as the Internet incorporates more satellite

links and wireless WANs, the RTT variation is going to increase. At the same time, RTT variation could be small in some cases. To meet these requirements, we dynamically adapt t_p according to the mean RTT of flows passing through the router. Each router computes the load factor σ during every t_p interval of time for each of its output links l as [20, 8, 9, 16, 13]:

$$\sigma = \frac{\lambda_l + \kappa_q \cdot q_l}{\gamma_l \cdot C_l \cdot t_p} \tag{13}$$

where λ_l is the amount of traffic during the period t_p, q_l is the persistent queue length during this period, κ_q controls how fast the persistent queue length drains and is set to 0.75. γ_l is the target utilization, and C_l is the capacity of the link. λ_l is measured using a packet counter whereas q_l is measured using exponentially weighted moving average. The queue sample time is set at 10 ms.

3.2.2. ***Adapting t_p according to the mean RTT of flows***

Every packet passing through a router carries the source's estimate of its RTT. The router uses this to estimate the mean RTT of flows. To ensure stability of the RTT estimate under heterogeneous delays, it is maintained in the following way:

(1) The data path of the MLCP router computes the average RTT over all packets seen in the control interval:

$$\overline{T}_d = \frac{\sum_i rtt_i}{n_T} \tag{14}$$

where $\overline{T}_d$ is the average round-trip time over interval $T = 10$ ms, rtt_i is the RTT estimate carried in the packet header and n_T is the number of packets seen in the control interval.

(2) The control path (which performs load factor computations periodically with a period of t_p) takes $\overline{T}_d$ as input and keeps a smoother average RTT estimate, $\overline{T}_c$. The data path RTT average is used in determining the moving-average gain as follows:

$$\begin{aligned} &if(\overline{T}_d \geq \overline{T}_c) \\ &\quad \theta = T/\overline{T}_c \\ &else \\ &\quad \theta = T \cdot \overline{T}_d/(\phi \cdot \overline{T}_c^2) \end{aligned}$$

where $\phi = 50$. The control path RTT estimate is updated as follows:

$$\overline{T}_c = \overline{T}_c + \theta(\overline{T}_d - \overline{T}_c) \tag{15}$$

The intuition behind the above expressions is that the gain should be at most $T/\overline{T}_c$ since if average RTT is larger than T, more than one samples are received

within the average RTT, so each sample should be given smaller weight. However, if $\overline{T}_d$ is smaller than $\overline{T}_c$, we want to be cautious in decreasing the control path estimate suddenly, and so the gain is made smaller by weighing it with $\overline{T}_d/\overline{T}_c$. A similar algorithm for updating the average RTT is used in [21].

The value of t_p is then chosen as follows:

$$t_p = \begin{cases} min_{\forall i \in |S|}\{s_i : s_i \in S, s_i \geq \overline{T}_c\}, & \text{if } \overline{T}_c < 1400 \\ 1400, & \text{if } \overline{T}_c \geq 1400 \end{cases}$$

where $S = \{80, 200, 400, 600, 800, 1000, 1200, 1400\}$. There are three reasons for choosing the set S. First, we do not need precise values of t_p because rigorous experimentation has shown that if the RTT of a flow is within 2.0-2.5 times t_p, there is hardly any queue buildup. Second, the mean RTT of flows must change significantly for t_p to get changed, ensuring that t_p doesn't fluctuate due to minor variations in the mean RTT. Third, these values can be communicated to the sources using only three bits. The value of t_p that is sent back to the sources is the one being used by the bottleneck router (the initial value for t_p was set at 200 ms). Using network scenarios with diverse RTTs, we show in Section 4.4 that setting t_p to the mean RTT of flows improves fairness significantly.

3.3. ***MLCP Receiver***

The MLCP receiver is similar to a TCP receiver except that when acknowledging a packet, it copies the header information from the data packet to its acknowledgment.

4. Performance Evaluation

Our simulations use the packet-level simulator ns2 [17], which we have extended with an MLCP module. We evaluate the performance of MLCP for a wide range of network scenarios including varying the link capacities in the range [100 Kbps,10 Gbps], round-trip times in the range [1 ms,1 s], number of long-lived, FTP-like flows in the range [1,1000], and arrival rates of short-lived, web like flows in the range [$1\,s^{-1}$,$1500\,s^{-1}$]. We always use *two-way traffic*. For TCP SACK, we use RED [8, 22] (with ECN enabled at the routers) and RIO (RED with *in/out* bit) [23] with the default parameter values provided in ns2. The bottleneck buffer size is set to the bandwidth-delay product, or two packets per-flow, whichever is larger. The data packet size is 1000 bytes, while the ACK packet size is 40 bytes. All simulations are run for at least 100 s unless specified otherwise. The statistics neglect the first 5% of the simulation time.

4.1. ***Single Bottleneck Topology***

We first evaluate the performance of MLCP for the case of a single bottleneck link shared by multiple MLCP flows. The basic setting is a 200 Mbps link with 80 ms RTT where the forward and reverse path each has 10 FTP flows. This corresponds to an average per-flow bandwidth of 20 Mbps. We evaluate the impact of each network parameter in isolation while retaining the others as the basic setting.

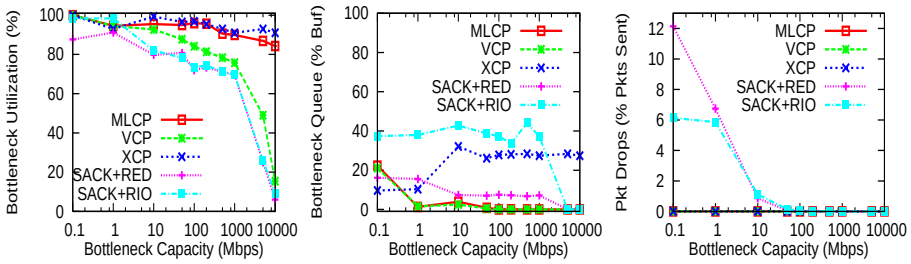

Figure 9: One bottleneck with capacity varying from 100 Kbps to 10 Gbps (Note the logarithmic scale on the x-axis).

4.1.1. ***Impact of Bottleneck Capacity***

MLCP achieves high utilization across a wide range of link capacities as shown in Figure 9. VCP, on the other hand, becomes inefficient at high link capacities. The utilization gap between MLCP and VCP starts widening when link capacities are increased beyond 10 Mbps. This difference becomes more than 60% on a 10 Gbps link. VCP's performance degrades because it uses a fixed MI factor of value 1.0625, which is too conservative for high link capacities. On the contrary, MLCP adapts its MI factor, increasing far more aggressively in low utilization regions, allowing it to remain efficient on high capacity links. Utilization with SACK+RED and SACK+RIO remains considerably lower than that of MLCP and VCP. This happens because TCP SACK uses a conservative increase policy of one packet/RTT and an aggressive decrease policy of halving the window on every congestion indication, leading to inefficiency on high BDP paths. Note that SACK+RED maintains a lower average queue length compared to SACK+RIO. This happens because unlike SACK+RIO, the adaptive RED algorithm, used by SACK+RED, dynamically adjusts its parameters to mark/drop packets more aggressively during times of congestion and is thus able to maintain a lower average queue length [22]. The average queue length for MLCP remains close to zero as we scale the link capacities. However, for very low capacities (*e.g.*,100 Kbps), MLCP results in an average queue length of about 20% despite keeping zero loss rate. This happens because the value of α is high for such capacities which leads to queue buildup. Note that while MLCP achieves roughly the same utilization as XCP, it is able to maintain a lower average bottleneck queue for link capacities $\geq$2 Mbps. Packet loss rate with VCP and XCP also remains close to zero whereas SACK+RED results in loss rates that are as high as 12% for low capacities.

4.1.2. ***Impact of Feedback Delay***

We fix the bottleneck capacity to 200 Mbps and vary the round-trip propagation delay from 1 ms to 1 s. As shown in Figure 10, MLCP scales better than VCP, XCP, SACK+RED, and SACK+RIO. For delays larger than 100 ms, the utilization gap between MLCP and VCP increases from roughly 5% to more than 40%. With SACK+RED, utilization drops most rapidly as delays are in-

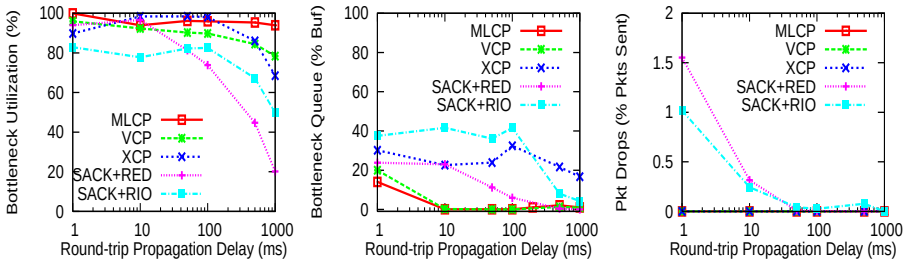

Figure 10: One bottleneck with round-trip propagation delay ranging from 1 ms to 1 s (Note the logarithmic scale on the x-axis).

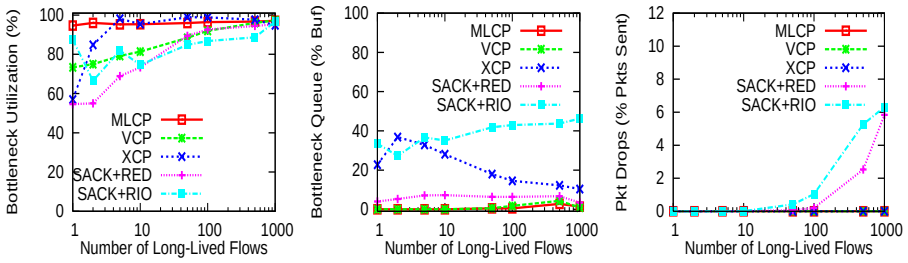

Figure 11: One bottleneck with the number of long-lived, FTP-like flows increasing from 1 to 1000 (Note the logarithmic scale on the x-axis).

creased. The difference between MLCP and SACK+RED increases from 20% for 100 ms to more than 60% for 1 s. With SACK+RIO, this difference increases from ≈20% to more than 40% for delays larger than 500 ms. Observe that when delays are less 50 ms, SACK+RED results in higher utilization than SACK+RIO. However, as delays are increased beyond 50 ms, adaptive RED becomes more aggressive than RIO, and therefore, SACK+RED achieves lower utilization. This is also evidenced by the fact that RED maintains much lower average queue length than RIO. It should be noted that the average queue length remains less than 15% for MLCP across the entire RTT range. These results indicate that MLCP could be effectively used in long-delay satellite networks.

4.1.3. ***Impact of Number of Long-lived Flows***

Figure 11 shows that as we increase the number of long-lived flows (in either direction), MLCP is able to maintain high utilization (≥90%), with negligible average queue length and near-zero packet drop rate. For small flow aggregates [1-50], SACK+RED's utilization remains lower than that of MLCP, VCP, XCP, and SACK+RIO (due to large per-flow bandwidth), whereas the difference between them grows to as large as 20%. When the number of flows is less than five, MLCP achieves a higher average utilization than XCP. However, as the number

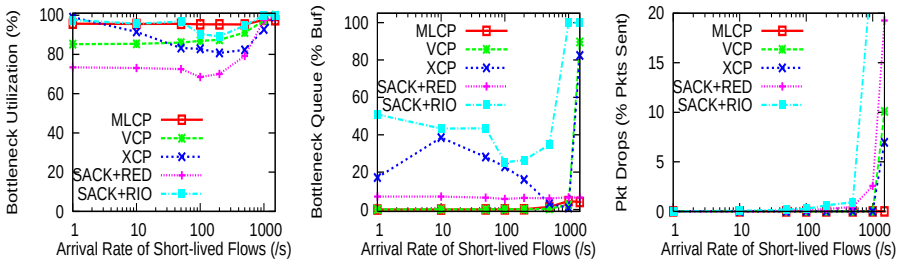

Figure 12: One bottleneck with short-lived, web-like flows arriving/departing at a rate from 1/s to 1500/s

of flows is increased, XCP and MLCP achieve similar average utilization. Note that MLCP has a much lower average queue size compared to XCP even though they have similar loss rates. Observe that, in most cases, SACK+RIO results in the highest average queue length. Moreover, as the number of flows increases, loss rate with SACK+RED and SACK+RIO also increases, reaching a value of ≈6% for 1000 flows.

4.1.4. ***Impact of Short-lived, Web-like Traffic***

To study the performance of MLCP in the presence of variability and burstiness in flow arrivals, we add web traffic into the network. These flows arrive according to a Poisson process, with an average arrival rate varying from 1/s to 1500/s. Their transfer size obeys the Pareto distribution with an average of 30 packets. This setting is consistent with the real-world web traffic model [24]. Figure 12 illustrates the performance of MLCP in comparison to VCP, XCP and TCP SACK. When the arrival rate is less than 1000/s, MLCP achieves higher utilization than VCP, XCP and SACK+RED/RIO. However, note that XCP, VCP, and SACK+RIO achieve more than 80% in all cases. When the arrival rate is increased beyond 1000/s, loss rate for VCP and XCP increases almost linearly to 10% and 7%, respectively. The average queue length for VCP and XCP rises to about 90% and 80% of the buffer size, respectively. This illustrates VCP's low responsiveness to high congestion; a consequence of using a single, high value of $\beta = 0.875$. MLCP, on the hand, is able to maintain almost 100% utilization, with negligible average queue length and near zero packet drop rate even under heavy congestion. Using multiple levels of MD allows MLCP to be more aggressive in its decrease policy than VCP, resulting in high responsiveness to congestion. Moreover, the AI parameter setting in VCP is too large when the link is heavily congested. MLCP, on the hand, applies II after the load factor exceeds 95%, which tends to lower the rate at which flows increase their rates. SACK+RED results in low link utilization when the arrival rate is smaller than 500/s. However, as we increase the load of short flows, utilization improves at the cost of a higher loss rate. SACK+RIO achieves high utilization but results in the highest average queue length across all protocols. Note that

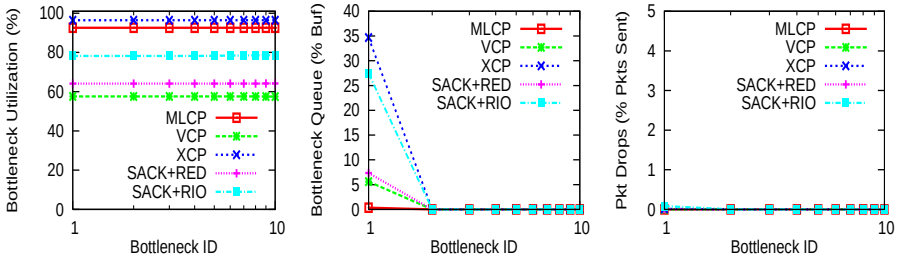

Figure 13: Multiple congested bottlenecks with capacity 100 Mbps

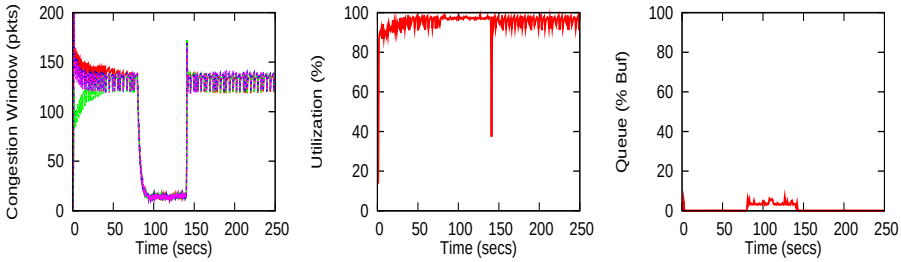

Figure 14: MLCP is robust against and responsive to sudden, traffic demand changes.

the minimum average queue length with SACK+RIO is ≈30%. For arrival rates higher than 500/s, the queue length rises to ≈100% and the loss rate becomes more than 20%.

4.2. ***Multiple Bottleneck Topology***

Next, we study the performance of MLCP with a more complex topology of multiple bottlenecks. For this purpose, we use a typical parking-lot topology with 10 bottlenecks, where each router-router link has capacity 100 Mbps and the propagation delay of each link is set at 20 ms. There are 30 long FTP flows traversing all the links in the forward direction, and 30 FTP flows in the reverse direction. In addition, each link has 5 cross FTP flows traversing the forward direction. The round-trip propagation delay for the 30 long-lived, FTP flows is set at 440 ms, whereas for the cross flows, it is 60 ms. Figure 13 shows that MLCP achieves roughly 30% higher utilization when compared with VCP and SACK+RED on all the 10 bottleneck links. With SACK+RIO, this difference is ≈15%. While XCP has a higher utilization than MLCP (about 4%), MLCP has a much smaller average queue length (about 35% less) at the first bottleneck. Note that all protocols achieve zero packet loss rate on all the links.

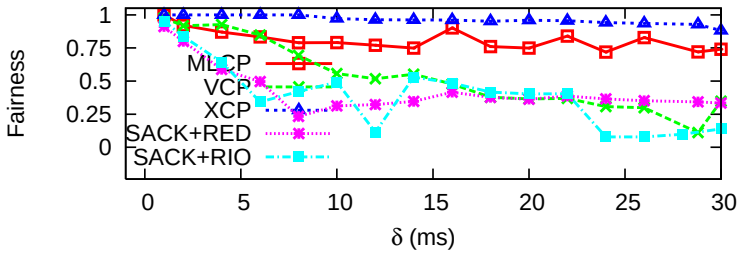

Figure 15: Jain's fairness index $\{(\sum_{i=1}^{N} x_i)^2/N \cdot \sum_{i=1}^{N} x_i^2$ for flow rates x_i, i∈[1,N]$\}$ under scenarios of one bottleneck link shared by 30 flows, whose RTT are in the ranges varying from [40 ms, 156 ms] to [40 ms, 3520 ms]

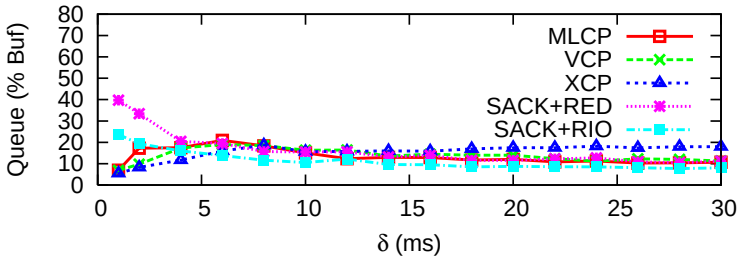

Figure 16: Bottleneck queue as a function of the RTT variation

4.3. ***Dynamics***

All the previous simulations focus on the steady-state behaviour of MLCP. Now, we investigate its short-term dynamics.

Sudden Demand Changes: To study the behaviour of MLCP when the demand at the bottleneck link changes suddenly, we used the following network settings. We consider 10 FTP flows (in either direction) with RTT=80 ms sharing a 150 Mbps bottleneck link. At $t = 80\,s$, 100 new forward FTP flows are made active; they leave at $t = 140\,s$. Figure 14 clearly shows that MLCP can quickly adapt to sudden fluctuations in the traffic demand. (The left figure draws the congestion window dynamics for four randomly chosen flows.) When the new flows enter the system, the flows adjust their rates to the new fair share while maintaining the link at high utilization. At $t = 140\,s$, when 100 flows depart creating a sudden drop in the utilization, the system quickly discovers this and ramps up to almost 100% utilization within a couple of seconds. Notice that during the adjustment period the bottleneck queue remains low. The result shows that MLCP is very responsive to sudden variations in the available bandwidth.

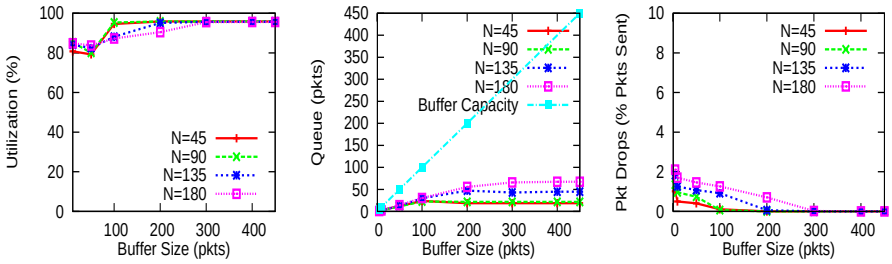

Figure 17: **One bottleneck with C=45 Mbps, RTT=80 ms and the number of long-lived flows varying from 45 to 180.**

4.4. ***Fairness***

We now compare the fairness properties of MLCP, VCP, SACK+RED, and SACK+RIO. We have 30 FTP flows (in both directions) sharing a single 60 Mbps bottleneck link. Each forward flow j's RTT is chosen according to $rtt_j = 40 + j * 4 * \delta$ ms for $j = 0, .., 29$, where δ is the one-way propagation delay for a non-bottleneck link. We perform simulations with δ varying from 1 ms to 30 ms. When δ is 1 ms, RTTs are in the range [40 ms,156 ms]. When $\delta = 30$, the RTTs are in the range [40 ms,3520 ms]. Figure 15 shows the comparison of the fairness achieved by MLCP, VCP, XCP and SACK+RED/RIO. While XCP achieves the highest level of fairness across a large range of RTT variations, MLCP, on the hand, achieves good fairness ($\geq$0.75, $\forall\delta$) while maintaining $<$20% average queue length (see Figure 16). With VCP and SACK+RED/RIO, fairness decreases considerably as the network incorporates more diverse RTT flows. In the case of VCP, this occurs due to the bounding of the MI and AI gains. With TCP, flows achieves RTT-proportional sending rates, therefore, large RTT flows achieve much smaller throughput than small RTT flows. This reduces the overall fairness for SACK+RED/RIO.

4.5. ***Impact of Buffer Size***

Recent advances in technology suggest that all-optical networks are likely to become commonplace in the future. However, optical packet buffers can only hold a few dozen packets in an integrated opto-electronic chip. Larger all-optical buffers remain infeasible, except with unwieldy spools of optical fiber (that can only implement delay lines, not true FCFS packet buffers) [25]. In order to make TCP amenable to small buffers, researchers have suggested using paced TCP [25], proposed changes in the AIMD parameters [26] and proposed new protocols to make it friendly with small buffers [27]. Dhamdhere et al. [28] showed that with small buffers (sized according to the square-root rule [29]) TCP observes loss rates as high as 15% on congested Internet links. The key problem with TCP is the coupling of packet loss with congestion indication that creates self-induced losses. MLCP decouples congestion indication from packet loss and thus only requires packet buffers for absorbing short-term packet bursts.

In this section, we analyze the performance of MLCP with small buffers. We consider a 45 Mbps link with a round-trip propagation delay of 80 ms. This gives a BDP of 450 packets where each packet has a size of 1 KB. We vary the buffer size from 10 pkts to 450 pkts and the number of long-lived flows, N, from 45 to 180 yielding a range of per-flow bandwidths in [250 Kbps,1 Mbps]. Figure 17 shows that even in the most congested network scenario (when N=180), with a buffer size of 10 packets (≈2% of the BDP of the path), MLCP is able to achieve >80% utilization and maintain a loss rate of less than 2%. As we increase the buffer size, utilization improves and the loss rate decreases sharply. Note that the average queue size remains less than 70 packets (<16% of the BDP of the path) even in the most congested scenario. These results indicate that MLCP can achieve high performance even with as small as 10 packets buffers.

5. STABILITY ANALYSIS

MLCP employs an aggressive load factor guided MI control law, which tracks the available bandwidth exponentially fast. This naturally raises stability concerns[6]. In this section, we observe that the fluid approximation model of the traffic presented in [13] can be used to show that such a control law does not result in instability when used in conjunction with a load factor dependent MD. The fluid model considers a single link shared by multiple flows and is described by the following differential equation:

$$\dot{w}_i(t) = \frac{1}{RTT} \cdot [w_i(t) \cdot \xi(\sigma(t)) + \alpha] \tag{16}$$

where $w_i(t)$ is the congestion window of flow i at time t and $\xi(\sigma(t)) = (1 - \sigma(t))/\sigma(t)$ and α are the MI and AI parameters, respectively.

The above differential equation models (1) the load-factor guided MI control law (characterized by $\xi(\sigma(t))$), (2) the AI control law (characterized by α), and (3) the MD control law whose parameter value depends on the load factor (characterized by $\xi(\sigma(t))$ for $\sigma(t) \geq 1$) and thus models the impact of multiple back-off factors. All of these features are used by MLCP. The stability conditions of the above model are given by the following theorem[7]:

Theorem 1. Under the model given by Equation 16, where a single bottleneck is shared by a set of synchronous flows with the same RTT, if $\kappa \leq \frac{1}{2}$, then the delayed differential equation described in [13] is globally asymptotically stable with a unique equilibrium $w^* = \gamma C \cdot RTT + N\frac{\alpha}{\kappa}$, and all the flows have the same steady-state rate $r_i^* = \frac{\gamma C}{N} + \frac{\alpha}{\kappa \cdot RTT}$

[6]Note that the results presented in Section 4 demonstrate the stability of MLCP across a wide range of network scenarios using simulation.

[7]For the proof of the model, we refer the reader to [13]

The above result holds for any link capacity, feedback delay, and number of flows. Moreover, the global stability result does not depend on the network parameters. It demonstrates that the use of the load factor guided MI, AI and multiple back-off factors does not cause instability as long as $\kappa \leq \frac{1}{2}$ for the case of a single bottleneck link. Intuitively, as the load factor at the bottleneck approaches 1, $\xi(\sigma(t))$ approaches zero, causing sources to become less aggressive as the available bandwidth decreases. When $\sigma(t)$ exceeds one, sources backoff in proportion to the amount of overload. This helps in keeping low persistent queue length. Note that this is unlike TCP's unguided exponential increase during slow-start, which becomes unstable as capacity or delay increases [12, 30].

The above model makes some simplifications in order to make the analysis tractable and differs from MLCP in the following ways: First, it uses MI and AI together at any given time. MLCP, on the other hand, uses either MI, AI, or II at a given time. In MLCP, AI results in a faster window growth than II because $\alpha \geq \frac{\alpha}{w_i(t)}$, $\forall w_i(t) \geq 1$, therefore, the MLCP window growth is less aggressive than the growth given by the above differential equation. Second, it uses the exact load factor, whereas MLCP uses a quantized value of the load factor to choose the control laws and their parameters values.

6. Related Work

In this section, we discuss and relate MLCP to two categories of congestion control schemes.

Explicit rate based/Congestion notification schemes: In RCP, each router assigns a single rate to all flows passing through it. Determining a single rate, however, requires an accurate estimate of the number of ongoing flows, a difficult task considering the dynamic nature of the Internet [15]. XCP regulates the sending rate by making routers send precise window increment/decrements in feedback to each flow [14]. ATM ABR service, previously, also proposed explicit rate control, however, ABR protocols usually maintain per-flow state at the switches and are essentially rate-based whereas MLCP is a window-based protocol and maintains no per-flow state in the routers [31]. VCP, like MLCP, uses load factor as a signal of congestion, however, it differs from MLCP in three ways: (1) MLCP uses 4-bits for feedback instead of 2, which allows it to obtain near-optimal performance in terms of rate of convergence to efficiency and fairness. (2) VCP uses a fixed t_p, which presents a trade-off between fairness and low queues, VCP chose the latter. MLCP, on the other hand, adapts t_p, which allows it to remain fair in the presence of diverse RTT flows while maintaining low queues and (3) VCP uses AIMD in steady-state, whereas MLCP employs AI-II-MD. This has two benefits. First, II enables smooth rate variations while improving fairness. Second, it considerably increases robustness to congestion [13]. BMCC uses the existing ECN bits along with a packet marking scheme to obtain load factor estimates of up to 16-bit resolution [32, 33]. It considerably reduces the response time of flows. However, unlike MLCP, it uses the MI-AI-MD control law and employs a fixed t_p. DCTCP has been proposed for use

in data center networks [34]. DCTCP employs a simple AQM scheme at the routers that marks packets based on the queue length. Like MLCP, DCTCP sources also adaptively backoff depending on the *degree* of congestion. Other than this, DCTCP uses the same algorithms and parameters as TCP SACK and therefore, doesn't address TCP's slow convergence and fairness issues in large BDP networks.

Pure end-to-end schemes: PCP chooses the sending rate for a flow by using a sequence of packets to determine the rate that the network can support. However, this requires accurate timers and small jitter for determining the available bandwidth correctly. While, PCP performs well in lightly loaded links, it is unclear how PCP's performance and stability properties vary under high load [35]. RAPID [36], like PCP, is a rate-based protocol that uses an available bandwidth estimation technique to determine a sending rate for flows. However, unlike PCP, it is able to probe for multiple rates within one RTT[8]. In order to scale to gigabit and multi-gigabit networks, it requires high-precision packet time-stamping and packet-spacing, which is challenging in today's networks. HighSpeed TCP adaptively sets the increase/decrease parameters according to the congestion window size [4]. FAST TCP uses queuing delay as a signal of congestion and improves on TCP Vegas's AIAD policy with a proportional controller [3, 6]. CTCP [37] uses loss and delay for congestion window growth. LTCP layers congestion control of two scales for high speed, large RTT networks [38]. BIC adds a binary search phase into the standard TCP for probing the available bandwidth in a logarithmic manner [5]. DCCP provides a framework for implementing congestion control protocols without reliability [39]. Since, MLCP builds on TCP in terms of reliability features, it would be a relatively simple task to incorporate it into the DCCP framework. However, since MLCP maintains low packet loss rate, real-time applications are likely to benefit from its reliability features too. Pure end-to-end schemes do not require explicit feedback. Therefore, it is hard for them to remain efficient and fair while keeping low queues and low loss rate. MLCP requires *only* four bits of congestion-related feedback and is able to achieve these goals in all likely network scenarios. Finally, [33], [40] and [41] present methods that can be used to implement MLCP using the existing ECN bits.

7. Conclusion and Future Work

In this paper, we analyzed the trade-off between increasing the amount of feedback information and the resulting performance improvements for load factor based congestion control protocols. We showed that while 2-bit scheme is far from optimal, using 3 bits is sufficient for achieving near-optimal performance in terms of rate of convergence to efficiency. We also showed that introducing multiple levels of MD allows a load factor based congestion protocols to achieve high

[8]PCP probes for only a single larger rate in one RTT

rate of convergence to fairness, smooth rate variations and increased robustness to congestion. Using these fundamental insights we designed a low-complexity protocol that achieves efficient and fair bandwidth allocations, minimizes packet loss and maintains low average queue size in high BDP networks.

We are currently investigating the efficacy of packet marking schemes in conveying high resolution congestion estimates using the existing ECN bits available in the IP header. Moreover, we plan to evaluate MLCP's performance using a real implementation which will allow us to assess its strengths and limitations in more practical settings. As future work, it would also be useful to study the stability properties of the fluid model, presented in Section 5, for the case of heterogeneous RTT flows.

The ns2 implementation code of MLCP is available at `http://www.cs.pitt.edu/~ihsan/`.

References

[1] ITU's Asia-Pacific Telecommunication and ICT Indicators Report focuses on broadband connectivity: Too much or too little?, `http://www.itu.int/newsroom/press_releases/2008/25.html`, 2008.

[2] V. Jacobson, Congestion Avoidance and Control, in: Proceedings of ACM SIGCOMM, 1988.

[3] C. Jin, D. Wei, S. Low, FAST TCP: Motivation, Architecture, Algorithms and Performance, in: IEEE INFOCOM, 2004.

[4] S. Floyd, HighSpeed TCP for Large Congestion Windows, in: IETF RFC 3649, 2003.

[5] L. Xu, K. Harfoush, I. Rhee, Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks, in: IEEE INFOCOM, 2004.

[6] L. Brakmo, L. Peterson, TCP Vegas: End to End Congestion Avoidance on a Global Internet, in: IEEE J. Selected Areas in Communications, 1995.

[7] H. Bullot, R. L. Cottrell, Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks., URL `http://www.slac.stanford.edu/grp/scs/net/talk03/tcp-slac-nov03.pdf`, 2003.

[8] S. Floyd, V. Jacobson, Random Early Detection Gateways for Congestion Avoidance, in: IEEE/ACM Trans. Networking, 1(4):397-413, 1993.

[9] S. Athuraliya, V. Li, S. Low, Q. Yin, REM: Active Queue Management, in: IEEE Network, 15(3):48-53, 2001.

[10] K. K. Ramakrishnan, S. Floyd, The Addition of Explicit Congestion Notification (ECN) to IP, in: IETF RFC 3168, 2001.

[11] C. Hollot, V. Misra, D. Towsley, W. Gong, Analysis and Design of Controllers for AQM Routers Supporting TCP Flows, in: IEEE/ACM Trans. Automatic Control, 47(6):945-959, 2002.

[12] S. Low, F. Paganini, J. Wang, J. Doyle, Linear Stability of TCP/RED and a Scalable Control, in: Computer Networks Journal, 43(5):633-647, 2003.

[13] Y. Xia, L. Subramanian, I. Stoica, S. Kalyanaraman, One More Bit Is Enough, in: Processings of ACM SIGCOMM, 2005.

[14] D. Katabi, M. Handley, C. Rohrs, Internet Congestion Control for High Bandwidth-Delay Product Networks, in: Processings of ACM SIGCOMM, 2002.

[15] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, N. McKeown, Procesor Sharing Flows in the Internet, in: Thirteenth International Workshop on Quality of Service 2005, 2005.

[16] R. Jain, S. Kalyanaraman, R. Viswanathan, The OSU Scheme for Congestion Avoidance in ATM Networks: Lessons Learnt and Extensions, in: Performance Evaluation, 31(1):67-88, 1997.

[17] ns-2 Network Simulator, URL `http://www.isi.edu/nsnam/ns/`, .

[18] R. Shorten, F. Wirth, D. Leith, A positive systems model of TCP-like congestion control: Asymptotic results, IEEE/ACM Trans. Networking 14 (2006) 616–629.

[19] W.-T. Tan, A. Zakhor, Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol, in: IEEE Trans. on Multimedia, 1999.

[20] S. Kunniyur, R. Srikant, Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management, in: Proceedings of ACM SIGCOMM, 2001.

[21] N. Dukkipati, Rate Control Protocol (RCP): Congestion control to make flows complete quickly, Ph.D. thesis, Department of Electrical Engineering, Stanford University, 2008.

[22] S. Floyd, R. Gummadi, S. Shenker, Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management, Tech. Rep., URL `http://www.icir.org/floyd/papers/adaptiveRed.pdf`, 2001.

[23] D. D. Clark, W. Fang, Explicit allocation of best-effort packet delivery service, IEEE/ACM Trans. Networking 6 (4) (1998) 362–373, ISSN 1063-6692.

[24] M. Crovella, A. Bestavros, Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes, in: IEEE/ACM Trans. Networking, 1997.

[25] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, T. Roughgarden, Routers with Very Small Buffers, in: IEEE INFOCOM, 2006.

[26] R. N. Shorten, D. J. Leith, On queue provisioning, network efficiency and the transmission control protocol, IEEE/ACM Trans. Netw. 15 (4) (2007) 866–877, ISSN 1063-6692.

[27] Y. Gu, D. Towsley, C. V. Hollot, H. Zhang, Congestion Control for Small Buffer High Speed Networks, IEEE INFOCOM .

[28] A. Dhamdhere, C. Dovrolis, Open issues in router buffer sizing, SIGCOMM Comput. Commun. Rev. 36 (1) (2006) 87–92, ISSN 0146-4833.

[29] G. Appenzeller, I. Keslassy, N. McKeown, Sizing router buffers, SIGCOMM Comput. Commun. Rev. 34 (4) (2004) 281–292, ISSN 0146-4833.

[30] S. Ha, I. Rhee, Hybrid Slow Start for High-Bandwidth and Long-Distance Networks, in: PFLDnet, 2008.

[31] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, B. Vandalore, The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks, in: IEEE/ACM Trans. Networking, 8(1), 2000.

[32] I. A. Qazi, L. L. H. Andrew, T. Znati, Two bits are enough, in: ACM SIGCOMM, Seattle, WA, (Extended Abstract), 2008.

[33] I. A. Qazi, L. L. H. Andrew, T. Znati, Congestion Control using Efficient Explicit Feedback, in: Proc. IEEE INFOCOM, Rio de Janeiro, Brazil, 2009.

[34] M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan, DCTCP: Efficient Packet Transport for the Commoditized Data Center, in: ACM SIGCOMM, 2010.

[35] T. Anderson, A. Collins, A. Krishnamurthy, J. Zahorjan, PCP: Efficient Endpoint Congestion Control, in: NSDI'06, 2006.

[36] V. V. R. Konda, J. Kaur, RAPID: Shrinking the Congestion-Control Timescale., in: IEEE INFOCOM, 2009.

[37] K. Tan, J. Song, A Compound TCP Approach for High-speed and Long Distance Networks, in: IEEE INFOCOM, 2006.

[38] S. Bhandarkar, S. Jain, A. Reddy, Improving TCP Performance in High Bandwidth High RTT Links Using Layered Congestion Control, in: PFLDNet, 2005.

[39] E. Kohler, M. Handley, S. Floyd, Designing DCCP: Congestion Control Without Reliability, in: Proceedings of ACM SIGCOMM, 2006.

[40] X. Li, H. Yousefi'zadeh, MPCP: multi packet congestion-control protocol, SIGCOMM Comput. Commun. Rev. 39 (5) (2009) 5–11, ISSN 0146-4833.

[41] N. Vasic, S. Kuntimaddi, D. Kostic, One Bit Is Enough: a Framework for Deploying Explicit Feedback Congestion Control Protocols, in: Proceedings of The First International Conference on COMmunication Systems and NETworkS (COMSNETS), 2009.

8. Authors' Biographies

Ihsan Ayyub Qazi Ihsan Ayyub Qazi is a Ph.D. Candidate in the Department of Computer Science at the University of Pittsburgh, PA, USA. He received his BSc (Hons) degree in Computer Science and Mathematics from the Lahore University of Management Sciences (LUMS), Lahore, Pakistan in 2005. He is a recipient of the prestigious Andrew Mellon Predoctoral Fellowship for the year 2009-2010. His current research interests include congestion control, routing, and MAC design for wired and wireless networks, virtualized large-scale testbeds for experimentation, and performance modeling of networked systems. For more information, please see: http://www.cs.pitt.edu/∼ihsan

Taieb Znati Taieb Znati is a Professor in the Department of Computer Science at the University of Pittsburgh with a joint appointment in Telecommunications in the Department of Information Science. He obtained his Ph.D. degree in Computer Science from Michigan State University, East Lansing in 1988, and a Master of Science Degree from Purdue University, West Lafayette, Indiana. He is currently on leave from the University of Pittsburgh to serve as Division Director in the Division of Computer and Network Systems (CNS) at the National Science Foundation. He has

served as general chair for a number of networking conferences including INFOCOM 2005 and SECON 2004. He is a member of the steering committee of ACM SenSys and has served on the editorial board of several journals including IEEE Transactions of Parallel and Distributed Systems, Wireless Networks Journal of Mobile Communication, Computation and Information, Journal of Adhoc Networks, Pervasive and Mobile Computing Journal. International Journal of Parallel and Distributed Systems and Networks, and Journal on Wireless Systems and Mobile Computing. His current research interests include routing and congestion in high speed networks, QoS in wired and wireless networks, data dissemination in wireless sensor networks, performance analysis of network protocols, and distributed systems. For more information, please see: http://www.cs.pitt.edu/∼znati