This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2008 proceedings.

On the Design of Load Factor based Congestion Control Protocols for Next-Generation Networks

Ihsan Ayyub Qazi Taieb Znati
Department of Computer Science
University of Pittsburgh, Pittsburgh, PA 15260
Email: {ihsan,znati}@cs.pitt.edu

***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 levels for the multiplicative decrease policy enables the protocol to adjust its rate of convergence to fairness, rate variations and responsiveness to congestion based on the degree of congestion at the bottleneck. Based on these fundamental 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. These features coupled with MLCP's smooth rate variations make it a viable choice for many real-time applications. Using extensive packet-level simulations we show that the protocol is stable across a diverse range of network scenarios. A fluid model for the protocol shows that MLCP remains globally stable for the case of a single bottleneck link shared by identical round-trip time flows.

I. Introduction

Future trends in technology ($e.g.$, increases in link capacities 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 bound to impact Internet stability.

Pure end-to-end schemes typically rely on packet loss and/or delay to infer congestion [1], [2], [3]. The heavy reliance on these indicators to deal with congestion implies that actions can only be taken after congestion occurs, which should be avoided in the first place. Further, 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 [2], [4]. The use of *explicit network feedback* has been proposed to overcome the limitations of end-to-end congestion control schemes. Although, traditional congestion notification schemes such as TCP+AQM/ECN proposals are able to reduce loss rate and queue size, they still fall short in achieving efficient bandwidth allocation in high BDP networks [5], [6], [7], [8], [9]. XCP addresses this problem by having the routers estimate the fair rate and send this back to the sources [10]. However, XCP is a network-based solution, whereby fairness and congestion control are enforced inside the network. Therefore, it is likely to introduce more overhead on routers than end-to-end or limited-feedback based schemes. Moreover, schemes such as XCP require considerable changes in the routers whereas congestion notification schemes like TCP+AQM/ECN typically involve modification at the end-hosts with incremental support from the routers.

VCP uses load factor (the relative ratio of demand and capacity) as a signal of congestion and sends two bits of explicit feedback to the sources [11]. However, VCP's rate of convergence to an efficient bandwidth allocation is far from optimal (see Section II). VCP's usage of a single, fixed Multiplicative Decrease (MD) parameter causes slow convergence to fairness and reduces responsiveness to congestion. Further, in the presence of diverse RTT flows, VCP becomes considerably unfair as shown by simulation results in Section IV. 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 improve fairness while maintaining low queues. 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

978-1-4244-2026-1/08/$25.00 © 2008 IEEE

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 4-bits while employing *load factor* as a signal of congestion [12]. In addition, each router also computes the *mean RTT* of flows passing through it, to dynamically adjust its load measurement interval. These two pieces of information are tagged onto each outgoing packet using *only* 7 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 [10]. 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 [13] simulations, we show that MLCP achieves high utilization, low persistent queue length, negligible packet drop rate and good fairness. We use a simple 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 II, we present the feedback analysis for determining the number of congestion levels. We describe the components of the protocol in Section III. In Section IV, we evaluate the performance of MLCP using extensive packet-level simulations. Section V describes the stability conditions of MLCP using a fluid model. Section VI discusses related work and Section VII offers concluding thoughts and future work.

II. 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-related 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, increasing the number of bits 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 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.

A. *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 [10], [11]. 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 κ=0.35 is a stability constant. The stability result presented in Section V 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 $1+\xi(\sigma)$ [11].

Fig. 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, more aggressive can the sources 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.

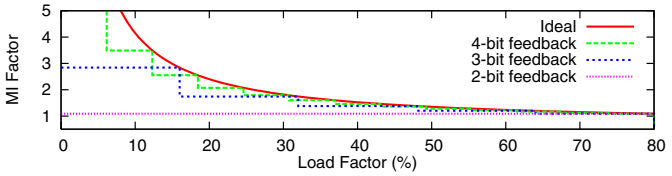

Fig. 1. **Comparison of MI factors of the ideal protocol with 2-bit, 3-bit and 4-bit feedback schemes**

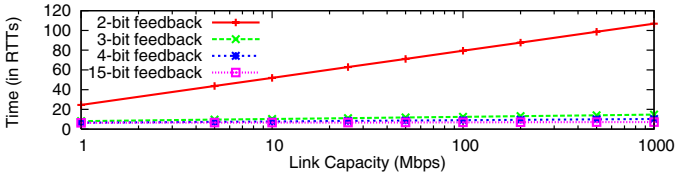

Fig. 2. **Comparison of the time required to achieve 80% utilization for 2-bit, 3-bit, 4-bit and 15-bit feedback schemes. The RTT was assumed to be 80ms.**

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 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 uses $l=2^n-3$ levels for representing the target utilization region. The rest of the symbols are used for representing load factor values above the target utilization region. Consider a single flow traversing a 1Gbps link with RTT=200ms and an initial congestion window size of 1KB.

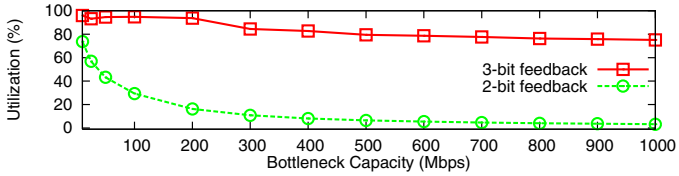

Fig. 3. **The figure shows the bottleneck utilization at t=10s as a function of link capacity for the 2-bit and 3-bit feedback schemes.**

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. Fig. 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 helps a small portion of the target utilization region (<10%, see Fig.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. Fig. 3 shows the bottleneck utilization at time t=10s 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 1Mbps to 1Gbps, whereas, for the 2-bit protocol, utilization falls significantly as link capacity is increased.

B. *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 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.

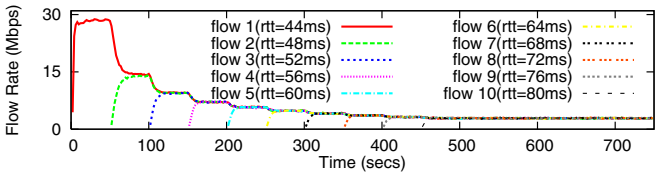

Fig. 4. **Flow rate as a function of time with link capacity of 30Mbps**

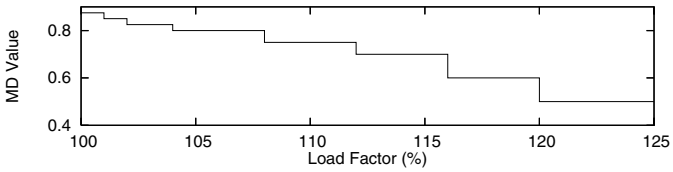

Fig. 5. β **values as a function of load factor for the 8-level MD scheme**

1) ***Measuring rate of convergence to fairness and the smoothness properties of a scheme****:* The definition of a metric that captures the convergence and smoothness properties of a given scheme is complicated by the fact that protocols using AIMD, IIMD, etc are inherently oscillatory and thus a notion of the rate of convergence to fairness is rather difficult to capture. On the other hand, *eventual* fairness is well captured by indices such as Jain's fairness index [14]. When defining such a metric, it is important to observe that *slow convergence* and *large rate variations*, both cause a flow to spend time away from its fair share. Slow convergence implies that an old flow takes time to release bandwidth and a new flow takes time to capture the bandwidth, whereas, large rate variations imply that flows spend time away from their fair share due to oscillations. Defining the metric as the fraction of bytes that a flow sends (if it was above its fair share) or could have sent (if it was below its fair share) is able to capture both these notions. Let $f_i(t)$ and $N(t)$ be the rate of flow i and the number of flows in the system at time t, respectively. The fair share of each flow is then $C/N(t)$. Suppose, flow i arrives at time t_i. Then, for flow i, the metric is defined as:

$$F_i = \frac{1}{C \cdot T} \cdot \sum_{t=t_i}^{T} |f_i(t) - \frac{C}{N(t)}| \cdot RTT_i \qquad (5)$$

where T is the total time, C is the link capacity, and RTT_i is the round-trip time of flow i.

2) ***Determining the MD levels****:* When the load factor at the bottleneck exceeds 100%, each flow applies MD. The MD parameter ($i.e.$, β) value directly impacts (a), (b) and (c). A high value of $\beta \approx 1$, leads to smooth rate variations but causes slow convergence and reduces responsiveness to congestion. A low value of $\beta \approx 0.5$, while improving the convergence rate and responsiveness to congestion, makes the protocol highly oscillatory, which is not desirable for real-time applications. Recall, our goal is to be able to discern between the two network scenarios. A single-level MD does not allow a protocol to distinguish between the two cases since it causes end-hosts to consider both the cases as being same. Another possibility is to use two levels of MD $i.e.$, a flow applies β_l for $100 \leq \sigma < d$ and β_h for $\sigma \geq d$, where d is the threshold used for applying different values of β. A drawback of this scheme is that since a load factor protocol cannot exactly discern between the two network scenarios, picking a fixed value for d is likely to lead to either slow convergence or highly oscillatory behaviour in many network scenarios. And it is unclear how d can be varied dynamically to yield the right response from sources. Another option is to use more levels of MD ($e.g.$, 8). This option allows the protocol to adjust its smoothness and convergence properties depending on the dynamic behaviour of the network.

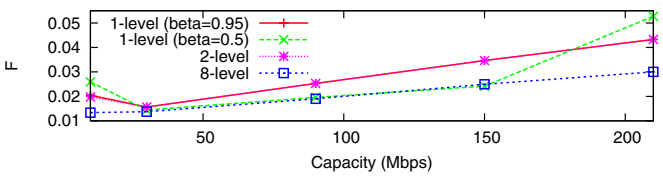

Fig. 6. F_1 **as a function of link capacity**

In order to compare these schemes, we use the following network scenario. Ten flows with heterogeneous RTTs are generated with an inter-arrival time of $50s$ as shown in Fig. 4. This scenario is the most interesting one since it gives each flow sufficient time to achieve its fair rate before new flows arrive. For the 2-level scheme, we set d=120%, β_l=0.5 and β_h=0.95. Fig. 5 shows the β values as a function of σ for the 8-level scheme. The 1-level schemes apply a single value of β for $\sigma \geq 100$. Fig. 6 shows the comparison of these schemes for different link capacities for the first flow. The 8-level scheme performs better than all schemes across a large range of link capacities. In comparison, the 1-level scheme (β=0.5) has similar performance for $30 \leq C \leq 150$, however, for $C<30$ and $C>150$, the 8-level scheme performs much better. The reason is that for small C, the 1-level scheme is more aggressive, applying a small β for each congestion signal, irrespective of the degree of congestion. For $C>150$, its performance degrades because as C increases, per-flow bandwidth increases, therefore, the magnitude of the rate variations goes up. The 2-level and the 1-level (β=0.95) schemes have identical performance for $C \geq 30$ because they get to apply the same β value. For C=10, the 2-level scheme performs slightly better because the load factor values exceed 120% in some cases, allowing the 2-level scheme to more be aggressive in those cases. The 8-level scheme performs best because it provides gentle variations in β values, thus allowing end-hosts to adapt the aggressiveness of their response according to the degree of congestion at the bottleneck.

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 small 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.

III. PROTOCOL

In this section, we describe the components of MLCP.

A. MLCP Sender: Control Laws

*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:

$$\mathsf{MI}: cwnd(t+rtt) = cwnd(t) \times (1+\xi(\sigma)) \tag{6}$$

where $\xi(\sigma)=\kappa \cdot \frac{1-\sigma}{\sigma}$, σ is the load factor and κ=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 following equations describe these control laws in terms of congestion window adjustments:

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

$$\mathsf{II}: cwnd(t+rtt) = cwnd(t) + \alpha/(cwnd(t))^{\frac{1}{2}} \tag{8}$$

$$\mathsf{MD}: cwnd(t+\delta t) = cwnd(t) \times \beta(\sigma) \tag{9}$$

where rtt=t_p, $\delta t \to 0$, α=5.0 and $0<\beta(\sigma)<1$. To avoid over reaction to the congestion signal, MD is applied only once per t_p interval.

*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 $(1+\xi_s)^{\frac{t_p}{rtt}}$ where ξ_s is the scaled parameter. To make the MI amount independent of a flow's RTT, $(1+\xi_s)^{\frac{t_p}{rtt}}=(1+\xi)$, which yields Eq.10. Similarly, the AI gain of a flow during a time interval t_p can be obtained by solving $1+\alpha=1+\frac{t_p}{rtt}\alpha_s$. 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.

$$\text{For MI}: \ \xi_s = (1+\xi)^{\frac{rtt}{t_p}} - 1, \tag{10}$$

$$\text{For AI and II}: \ \alpha_s = \alpha \cdot (rtt/t_p), \tag{11}$$

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 each 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:

$$c_i(M) = \beta \cdot (c_i(M-1) + n \cdot \alpha_i + m \cdot \frac{\alpha_i}{\sqrt{c_i(M-1)}}) \tag{12}$$

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, as shown next:

$$\frac{c_1(M)}{c_2(M)} = \frac{c_1(M-1)+\alpha_1(n+\frac{m}{\sqrt{c_1(M-1)}})}{c_2(M-1)+\alpha_2(n+\frac{m}{\sqrt{c_2(M-1)}})}$$

$$= \frac{\beta c_1(M-2)+\alpha_1(n+\beta n+\frac{m}{k_1}+\frac{\beta m}{\sqrt{c_1(M-2)}})}{\beta c_2(M-2)+\alpha_2(n+\beta n+\frac{m}{k_2}+\frac{\beta m}{\sqrt{c_2(M-2)}})}$$

$$\text{where } k_i = (\beta c_i(M-2) + \frac{\alpha_i \beta n}{\sqrt{c_i(M-2)}} + \alpha_i \beta n)^{0.5}$$

$$= \frac{\beta^2 c_1(M-3)+\alpha_1(n+\beta n+\beta^2 n+\frac{m}{a_1}+\frac{\beta m}{b_1}+\frac{\beta^2 m}{c_1})}{\beta^2 c_2(M-3)+\alpha_2(n+\beta n+\beta^2 n+\frac{m}{a_2}+\frac{\beta m}{b_2}+\frac{\beta^2 m}{c_2})}$$

where 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 $\frac{\alpha_1}{\alpha_2}$. Therefore, to allocate the bandwidth fairly among two flows, we scale the α parameter of each flow by its own RTT.

$$\alpha_f = \alpha_s \cdot (rtt/t_p) = \alpha \cdot (rtt/t_p)^2 \tag{13}$$

B. MLCP Router

A MLCP router performs two main functions. It computes the load factor and the mean RTT of flows passing through it.

*1) **Estimating the load factor**:* Load factor is estimated over an interval t_p. However, there are two conflicting requirements that a value of t_p should 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 highly on the RTT of flows, which varies significantly across Internet links. For example, in [11], 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 IV-D. 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 other 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 [15], [5], [6], [12], [11]:

$$\sigma = \frac{\lambda_l + \kappa_q \cdot q_l}{\gamma_l \cdot C_l \cdot t_p} \tag{14}$$

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 10ms.

*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 update the moving average, $\tilde{m}$, as follows:

$$\tilde{m} = a \cdot rtt_{pkt} + (1 - a) \cdot \tilde{m} \tag{15}$$

where a=0.02. The running average gives an estimate of the average RTT across all the packets passing through the router. This skews the RTT estimate towards flows which have large number of packets. This is desired since the flows with a large number of packets will last for many RTTs. The value of t_p is then chosen as follows:

$$t_p = \begin{cases} min_{\forall i \in |S|}\{s_i : s_i \in S, s_i \geq \tilde{m} - 1\}, & \text{if } \tilde{m} < 1400 \\ 1400, & \text{if } \tilde{m} \geq 1400 \end{cases}$$

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 3 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 200ms). Using network scenarios with diverse RTTs, we show in Section IV-D that setting t_p to the mean RTT of flows improves fairness significantly.

C. 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.

IV. Performance Evaluation

Our simulations use the packet-level simulator ns2 [13], 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 [100Kbps,10Gbps], round-trip times in the range [1ms,5s], number of long-lived, FTP-like flows in the range [1,1000], and arrival rates of short-lived, web like flows in the range [$1s^{-1}$,$2000s^{-1}$]. We always use *two-way traffic*. For TCP SACK, we always use RED with ECN enabled at the routers. 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 atleast 100s unless specified otherwise. The statistics neglect the first 5s of the simulation time.

A. 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 200Mbps link with 80ms RTT where the forward and reverse path each has 10 FTP flows. This corresponds to an average per-flow bandwidth of 20Mbps. We evaluate the impact of each network parameter in isolation while retaining the others as the basic setting.

*1) **Impact of Bottleneck Capacity**:* MLCP achieves high utilization across a wide range of link capacities as shown in Fig. 7. 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 10Mbps. This difference becomes more than 60% on a 10Gbps 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 TCP SACK remains considerably lower than that of MLCP and VCP. This happens because TCP 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. The average queue length for MLCP remains close to zero as we scale the link capacities. However, for very low capacities ($e.g.$,100Kbps), 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. Packet loss with VCP also remains close to zero whereas SACK results in packet loss rates that are as high as 12% for low capacities.

*2) **Impact of Feedback Delay**:* We fix the bottleneck capacity to 200Mbps and vary the round-trip propagation delay from 1ms to 5s. As shown in Fig. 8, MLCP scales better than VCP and SACK. For delays larger than 100ms, the utilization gap between MLCP and VCP increases from roughly 5% to more than 40%. With TCP SACK, utilization drops more rapidly as delays are increased. The difference between MLCP and SACK increases from 20% for 100ms to more than 60% for 5s. 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.

*3) **Impact of Number of Long-lived Flows**:* Fig. 9 shows that as we increase the number of long-lived flows (in either direction), MLCP is able to maintain high utilization ($\geq$90%), with negligible average queue length and near-zero packet drop rate. For small flow aggregates [1-50], TCP SACK's utilization remains lower than that of MLCP and VCP (due to larger available per-flow bandwidth), whereas the difference

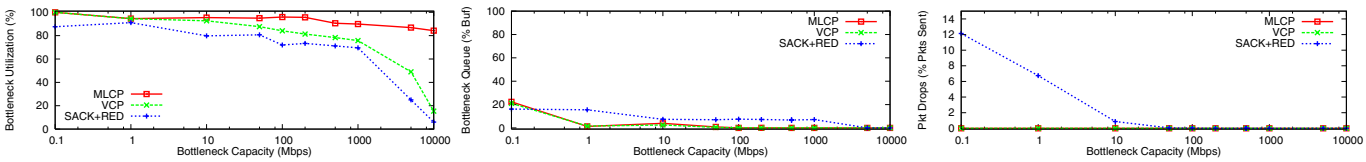

Fig. 7. **One bottleneck with capacity varying from 100Kbps to 10Gbps (Note the logarithmic scale on the x-axis).**

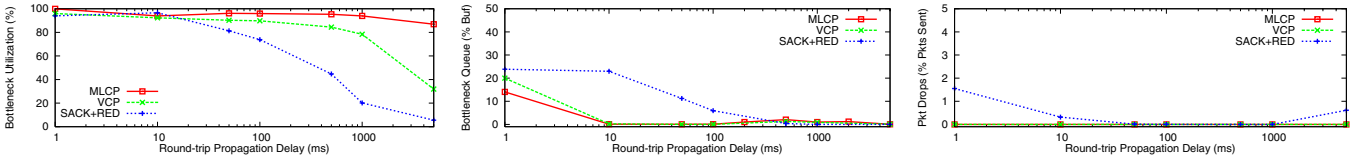

Fig. 8. **One bottleneck with round-trip propagation delay ranging from 1ms to 5s (Note the logarithmic scale on the x-axis).**

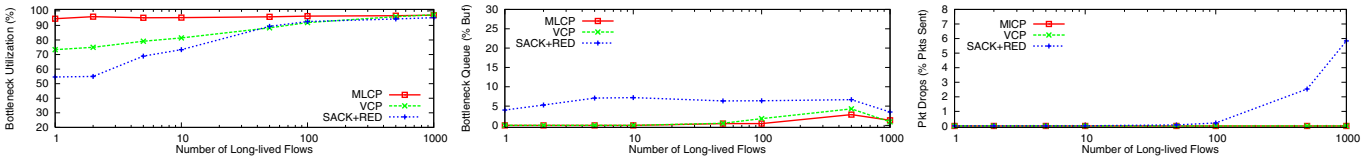

Fig. 9. **One bottleneck with the number of long-lived, FTP-like flows increasing from 1 to 1000 (Note the logarithmic scale on the x-axis).**

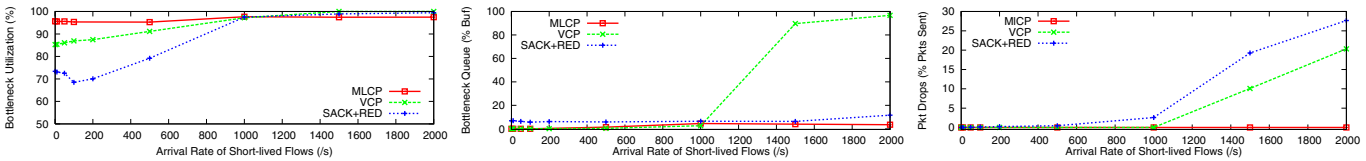

Fig. 10. **One bottleneck with short-lived, web-like flows arriving/departing at a rate from 1/s to 2000/s**

between them grows to as large as 20%. SACK results in higher average queue length than MLCP and VCP. Loss rate for SACK, however, increases to only as high as 6%. This relatively low loss rate for SACK is a consequence of using RED with ECN enabled at the routers.

*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 2000/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 [16]. Fig. 10 illustrates the performance of MLCP in comparison to VCP and TCP SACK. When the arrival rate is less than 1000/s, the performance of MLCP and VCP is quite similar. However, when the arrival rate is increased beyond 1000/s, VCP's loss rate increases almost linearly to 20% for 2000 flows/s and the average queue length rises to almost 100% of the buffer size. This illustrates VCP's low responsiveness to high congestion; a consequence of using a single, high value of β=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. TCP SACK results in low link utilization when the arrival rate is smaller than 500/s. However, as the arrival rate increases, the traffic becomes more bursty due to many flows being in slow-start which causes packet losses to increase.

B. 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 100Mbps and the propagation delay of each link is set at 20ms. 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 440ms, whereas for the cross flows, it is 60ms. Fig. 11 shows that compared with VCP and TCP SACK, MLCP achieves $\geq$15% utilization on all the 10 bottleneck links. Moreover, MLCP maintains low average queue length and zero packet loss rate on all links.

C. 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 bevahiour of MLCP when the demand at the bottleneck link changes suddenly, we used the following network settings. We consider 10 forward FTP flows (in either direction) with varying RTTs (uniformly chosen in the range [44ms,116ms]) sharing a 150 Mbps bottleneck link. At t=80s, 100 new forward FTP flows are made active; they leave at t=140s. Fig. 12 clearly shows that MLCP can quickly adapt to sudden fluctuations in the

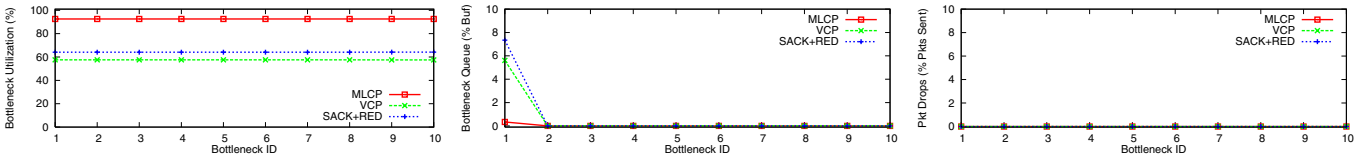

Fig. 11. **Multiple congested bottlenecks with capacity 100Mbps**

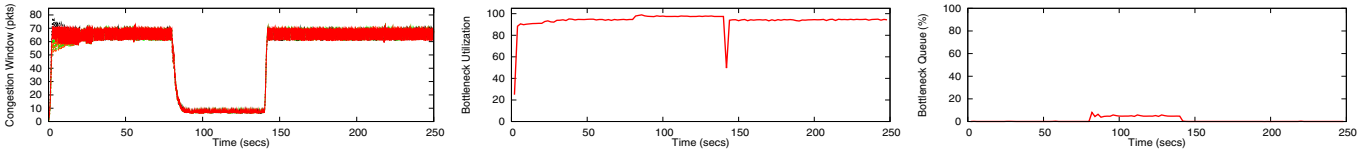

Fig. 12. **MLCP is robust against and responsive to sudden, traffic demand changes.**

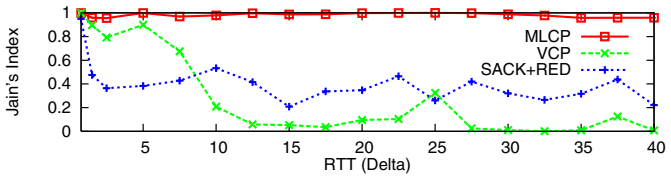

Fig. 13. **Jain's fairness index $\{(\sum_{i=1}^{N} x_i)^2/N \cdot \sum_{i=1}^{N} x_i^2$ for flow rates x_i, i∈[1,N]$\}$ under scenarios of one bottleneck link shared by 30 flows, whose RTT are in the ranges varying from [43ms, 130ms] to [200ms, 4840ms]**

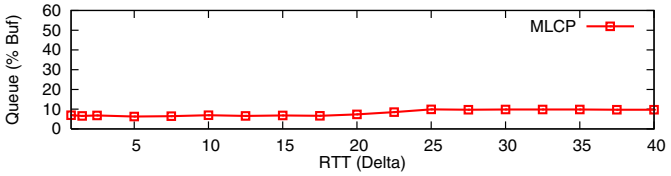

Fig. 14. **Bottleneck queue as a function of the RTT variation**

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=140s, 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.

D. *Fairness*

Here, we compare the fairness properties of MLCP, VCP and TCP SACK. We have 30 FTP flows (in both directions) sharing a single 90Mbps bottleneck link. Each forward flow j's RTT is chosen according to rtt_j=40+j∗4∗δ ms for j=1, .., 30, where δ is the one-way propagation delay for a non-bottleneck link. We perform simulations with δ varying from 0.75ms to 40ms. When δ is 0.75ms, RTTs are in the range [43ms,130ms]. When δ=40, the RTTs are in the range [200ms,4840ms]. MLCP achieves high level of fairness ($\geq$0.95) across a large range of RTT variations while maintaining <15% average queue length as shown in Figures 13 and 14. With VCP, fairness decreases considerably as the network incorporates more diverse RTT flows. TCP SACK outperforms VCP for $\delta \geq$7.5. This occurs due to the fact that with TCP, flows receive RTT-proportional throughput whereas in VCP, large RTT flows receive considerably less than that amount.

V. STABILITY ANALYSIS

We use a fluid model of the traffic to analyze the stability of MLCP. Our analysis considers a single link shared by multiple MLCP flows. We show by *reduction* that the analytic model in [11] can be used to describe the stability properties of MLCP. The following differential equation can be used to approximate the behaviour of MLCP as defined by (6), (7), (8) and (9).

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

where $w_i(t)$ is the congestion window of flow i at time t and $\xi(\sigma(t))$ and α are the MI and AI parameters, respectively. However, the above differential equation assumes that the protocol uses MI, AI and II factor terms together at any given time. Since this is not true and given that $\alpha \geq \frac{\alpha}{w_i(t)}$ $\forall w_i(t) \geq 1$, the following differential equation would describe the behaviour of MLCP more closely:

$$\dot{w}_i(t) = \frac{1}{RTT} \cdot [\xi(\sigma(t)) + \alpha] \qquad (17)$$

The above model is the same as used by [11]. Therefore, we state the stability conditions without going into further details. We refer the reader to [11] for the proofs of the model.

Theorem 1: *Under the above model 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 [11] 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 $r_i^* = \frac{\gamma C}{N} + \frac{\alpha}{\kappa RTT}$*

VI. 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 [17]. XCP regulates the sending rate by making routers send precise window increment/decrements in feedback to each flow [10]. 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 [18]. 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 [11].

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 [19]. HighSpeed TCP adaptively sets the increase/decrease parameters according to the congestion window size [2]. FAST TCP uses queuing delay as a signal of congestion and improves on TCP Vegas's AIAD policy with a proportional controller [1], [20]. LTCP layers congestion control of two scales for high speed, large RTT networks [21]. BIC adds a binary search phase into the standard TCP for probing the available bandwidth in a logarithmic manner [3]. DCCP provides a framework for implementing congestion control protocols without reliability [22]. 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.

VII. Conclusion

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 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. A fluid model of the protocol showed that the protocol remains globally stable for the case of single bottleneck link shared by identical RTT flows.

As part of our ongoing work, we are investigating the efficacy of packet marking schemes in providing high resolution link price estimates using the two 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.

VIII. Acknowledgments

We would like to thank Sonia Fahmi, Daniel Mosse and the anonymous reviewers for comments on earlier drafts of the paper. This work was supported by NSF under grant 05-010684.

References

[1] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, Architecture, Algorithms and Performance," in *IEEE INFOCOM*, Mar 2004.
[2] S. Floyd, "HighSpeed TCP for Large Congestion Windows," in *IETF RFC 3649*, Dec 2003.
[3] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks," in *IEEE INFOCOM*, Mar 2004.
[4] H. Bullot and R. L. Cottrell, "Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks." [Online]. Available: http://www.slac.stanford.edu/grp/scs/net/talk03/tcp-slac-nov03.pdf
[5] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," in *IEEE/ACM Trans. Networking, 1(4):397-413*, Aug 1993.
[6] S. Athuraliya, V. Li, S. Low, and Q. Yin, "REM: Active Queue Management," in *IEEE Network, 15(3):48-53*, May 2001.
[7] K. K. Ramakrishnan and S. Floyd, "The Addition of Explicit Congestion Notification (ECN) to IP," in *IETF RFC 3168*, Sep 2001.
[8] C. Hollot, V. Misra, D. Towsley, and W. Gong, "Analysis and Design of Controllers for AQM Routers Supporting TCP Flows," in *IEEE/ACM Trans. Automatic Control, 47(6):945-959*, Jun 2002.
[9] S. Low, F. Paganini, J. Wang, and J. Doyle, "Linear Stability of TCP/RED and a Scalable Control," in *Computer Networks Journal, 43(5):633-647*, Dec 2003.
[10] D. Katabi, M. Handley, and C. Rohrs, "Internet Congestion Control for High Bandwidth-Delay Product Networks," in *Processings of ACM SIGCOMM*, Aug 2002.
[11] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One More Bit Is Enough," in *Processings of ACM SIGCOMM*, Aug 2005.
[12] R. Jain, S. Kalyanaraman, and R. Viswanathan, "The OSU Scheme for Congestion Avoidance in ATM Networks: Lessons Learnt and Extensions," in *Performance Evaluation, 31(1):67-88*, Nov 1997.
[13] "ns-2 Network Simulator," http://www.isi.edu/nsnam/ns/.
[14] D.-M. Chiu and R. Jain, "Analysis of Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," in *Computer Networks and ISDN Systems*, Jun 1989.
[15] S. Kunniyur and R. Srikant, "Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management," in *Proceedings of ACM SIGCOMM*, Aug 2001.
[16] M. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," in *IEEE/ACM Trans. Networking*, Dec 1997.
[17] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown, "Processor Sharing Flows in the Internet," in *Thirteenth International Workshop on Quality of Service 2005*, Jun 2005.
[18] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, and B. Vandalore, "The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks," in *IEEE/ACM Trans. Networking, 8(1)*, Feb 2000.
[19] T. Anderson, A. Collins, A. Krishnamurthy, and J. Zahorjan, "PCP: Efficient Endpoint Congestion Control," in *NSDI'06*, 2006.
[20] L. Brakmo and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," in *IEEE J. Selected Areas in Communications*, Oct 1995.
[21] S. Bhandarkar, S. Jain, and A. Reddy, "Improving TCP Performance in High Bandwidth High RTT Links Using Layered Congestion Control," in *PFLDNet*, Feb 2005.
[22] E. Kohler, M. Handley, and S. Floyd, "Designing DCCP: Congestion Control Without Reliability," in *Proceedings of ACM SIGCOMM*, 2006.