Congestion control with multi-packet feedback

Ihsan Ayyub Qazi, Lachlan L. H. Andrew, and Taieb Znati

***Abstract*— Many congestion control protocols use explicit feedback from the network to achieve high performance. Most of these either require more bits for feedback than are available in the IP header or incur performance limitations due to inaccurate congestion feedback. There has been recent interest in protocols which obtain high resolution estimates of congestion by combining the ECN marks of multiple packets, and using this to guide MI-AI-MD window adaptation. This paper studies the potential of such approaches, both analytically and by simulation. The evaluation focuses on a new protocol called Binary Marking Congestion Control (BMCC). It is shown that these schemes can quickly acquire unused capacity, quickly approach a fair rate distribution, and have relatively smooth sending rates, even on high bandwidth-delay product networks. This is achieved while maintaining low average queue length and negligible packet loss. Using extensive simulations, we show that BMCC outperforms XCP, VCP MLCP, CUBIC, CTCP, SACK, and in some cases RCP, in terms of average flow completion times. Suggestions are also given for the incremental deployment of BMCC.**

***Index Terms*— Congestion Control, TCP, AQM, ECN**

I. Introduction

The Transmission Control Protocol (TCP) has been instrumental to the successful development and growth of the Internet and its applications. However, recent advances in wired and wireless communications technology have lead to a tremendous growth in the range of path bandwidth-delay products (BDP) in the Internet [1]. There has simultaneously been an increase in the diversity of applications carried over the Internet (e.g., voice over IP, video conferencing, and social networking). These advances have stressed the congestion control algorithm in TCP and the need for more efficient, fair, robust, and easy to deploy congestion control protocols is increasingly important.

While considerable research has gone into addressing the limitations of TCP, prior research has focused on two extreme points in the design space of congestion control protocols.

At one end are end-to-end schemes that rely on implicit congestion signals such as packet loss [2], delay [3], jitter [4] or combinations of these [5]. Since loss and delay only occur once a link is overloaded, these schemes must introduce unnecessary packet losses or queuing at the bottleneck.

On the other end are "network-based" schemes in which routers explicitly specify a rate (used by RCP [6] and ATM's Available Bit Rate (ABR) [7]) or change in window size (used by XCP [1]) for each individual flow. These schemes face two big deployment challenges on today's Internet: (1) They require more bits for feedback than are available in the IP header (128 for XCP [8] and 96 for RCP [9]) which requires either changing the IP header[1], use of IP options, or the addition of a "shim" layer. This makes universal deployment of such protocols challenging. (2) It is difficult for such protocols to co-exist with TCP traffic, often requiring complex router-level mechanisms for fair bandwidth sharing [8], [9].

Between these extremes are "limited feedback" schemes, such as TCP with random early discard (RED) and explicit congestion notification (ECN) [11], [12], and VCP [13], which require changes at the end-hosts with incremental support from the routers. In these schemes, each packet signals the congestion level with up to 2-bit resolution. VCP uses a 2-bit estimate of the load factor (ratio of input traffic plus queue length to capacity) to trigger a fixed multiplicative increase (MI), additive increase (AI) or multiplicative decrease (MD) window update, at low load, high load and overload. However, convergence speed improves significantly when the load is estimated to 4-bit resolution [14], for which there is insufficient space in the IP header of a single packet.

There are several methods for obtaining high resolution congestion estimates using the existing ECN bits of streams of packets. Random marking of packets based on the level of congestion was proposed in [15], and used by the REM protocol [16]. Higher resolution estimates can be obtained using side-information in packet headers to indicate how to interpret the ECN mark of a given packet [17], [18], [19]. Most of these schemes require a pre-specified resolution, which is traded against the number of packets needed to carry the ECN marks. Adaptive deterministic packet marking (ADPM [20]) implicitly adapts its effective quantization resolution based on the dynamics of the value and obtains a resolution of $1/n$ after receiving n packets, for n up to 2^{16} or beyond. ADPM provides a MSE that is several orders of magnitude smaller [20], [21] than the estimators based on random marking of packets [15], [16] or deterministic marking with static quantization [19], [22].

This paper investigates the ability of MI-AI-MD schemes (such as VCP [13]) to use high-resolution estimates of load factor based on ECN marks spread over multiple packets. To this end, we design the Binary Marking Congestion Control

Ihsan Ayyub Qazi (corresponding author) is with the Department of Computer Science at the LUMS School of Science and Engineering, Lahore, Pakistan. (Email: ihsan.qazi@lums.edu.pk). While performing a subset of this work, Ihsan Ayyub Qazi was with the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, 15260 USA and the Centre for Advanced Internet Architectures, Swinburne University of Technology, Hawthorn, VIC 3122, Australia.

Lachlan Andrew is with the Centre for Advanced Internet Architectures, Swinburne University of Technology, Hawthorn, VIC 3122, Australia (Email: landrew@swin.edu.au).

Taieb Znati is with the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, 15260 USA (Email: znati@cs.pitt.edu)

This work was supported by NSF grants 010536 and 010684, Australian Research Council (ARC) grants DP0985322 and FT0991594, and the LUMS Faculty Startup Grant.

[1]Modifying IP is slow: work on IPv6 started around 1992, but it accounted for less than 1% of Internet traffic as of December 2008 [10].

(BMCC) protocol that uses ADPM to obtain congestion estimates with up to 16-bit resolution using ECN, in a way compatible with existing RED/ECN. We present analytical models that provide insights into the convergence properties of MI-AI-MD algorithms, and study the incremental deployment of these protocols when they co-exist with TCP traffic and traverse different kinds of bottleneck routers.

With BMCC, each router periodically computes the load factor on each of its links. End-hosts obtain a high resolution estimate of the bottleneck load factor using ADPM. With ADPM, each packet signals a bound on the bottleneck load factor. The receiver's estimate of the load factor is updated whenever a new packet provides a tighter bound. This estimate is echoed back to the sources via acknowledgement packets using TCP options. Based on this value, sources apply load-dependent MI-AI-MD algorithm. BMCC achieves efficient and fairness bandwidth allocations on high BDP paths while maintaining low bottleneck queue and loss rate. In terms of average flow completion times (AFCT) for typical Internet flow sizes, BMCC outperforms XCP, VCP, MLCP, CUBIC, CTCP, SACK [23] using RED/ECN, and sometimes even RCP, which was optimized for AFCT.

The main contribution of this paper is the *design and analysis of a congestion control protocol that achieves performance comparable to that of feedback-rich protocols, such as RCP, using only incrementally deployable signaling.* The presentation and analysis of the proposed protocol involves three different components.

The first component, presented in Section II, focuses on the specification of the BMCC protocol, which uses the existing two ECN bits to obtain high resolution congestion estimates. The major issues related to the design of BMCC are discussed in Section III. BMCC achieves faster convergence by using ADPM to increase the feedback resolution, allowing faster rate increase during low load and more precise decreases during congestion.

The second component, presented in Section IV, focuses on the analysis of BMCC. It is worth noting that the analysis is applicable, not only to BMCC, but also to the class of MI-AI-MD algorithms with multi-bit resolution, including UNO [24] and MPCP [25].

The analysis shows that reducing the MI factor when load increases results in a concave window growth, which quickly acquires spare capacity with minimal overshoot. The ns-2 simulation results in Section V show that BMCC achieves better throughput, fairness and short AFCT than many benchmark protocols in a wide range of scenarios.

The final component, discussed in Section VI, studies the feasibility of BMCC's incremental deployment in the Internet. The study focuses on the co-existence of BMCC with different versions of TCP, such as CUBIC [2], CTCP [5], and TCP SACK, and on how to deal with situations where traffic may traverse network bottlenecks that do not all support BMCC marking. To prevent starvation of BMCC flows, a modification of the original BMCC protocol is proposed, whereby the decision of when to respond to congestion is decoupled from the decision of how to respond to congestion. This allows TCP and BMCC sources to back off based on a common signal while allowing BMCC sources to retain the benefit of basing the backoff factor on the estimated congestion level. This solution has applicability beyond BMCC.

II. Algorithm and Protocol

A congestion control protocol must specify how congestion is estimated, how that information is communicated to the sender, and how the sender should respond. BMCC estimates congestion in terms of the *load factor* at each link, which is a weighted sum of the link utilization and the queueing delay [13]. The maximum load factor along a flow's path is communicated using ADPM, to the sender which responds using load-dependent MI-AI-MD. The components of a BMCC system are as follows.

A. BMCC Router

A BMCC router divides time into intervals of length t_p, and computes the load factor in each interval as [7], [13], [16]

$$f = \frac{\lambda + \kappa_1 q_{av}}{\gamma_l C_l t_p} \tag{1}$$

where λ is the amount of bytes received during t_p, C_l is the link capacity in bytes/sec, $\gamma_l \leq 1$ is the target utilization[2] [26], q_{av} is an estimate of the average queue length in bytes[3], and $\kappa_1 \leq 1$ controls how fast to drain the queue [11], [16].

The router conveys its load factor to the sender by applying ADPM [20] to packets' ECN bits as follows: Recall that the ECN bits on an unmarked packet are initially $(10)_2$, and routers set these bits to $(11)_2$ to indicate congestion. Choose u to be a "severely congested" load factor. BMCC marks the packet with $(11)_2$ if $f \geq u$ or the packet already contains a mark $(11)_2$. Otherwise, ADPM computes a deterministic hash h of the packet contents, such as the 16-bit IPid field in the IPv4 header or the checksum of the payload in case of IPv6. This hash is compared to f, and the packet is marked with $(01)_2$ if $f > h$, or left unchanged otherwise. At the receiver, the ECN bits will reflect the state of the most congested router on the path.

Router Implementation Complexity: For each packet, a BMCC router calculates a hash of the packet (simply reversing the bits of the IPid field [20]), performs a comparison, and sets up to two header bits. At the larger timescale of t_p, it requires multiplication by two constants and an addition to calculate f. (Note that the denominator of (1) is a constant.) It also uses a count of the total bytes sent, which routers already record, and the average queue size, which can be measured at a frequency that does not scale up with bit-rate.

B. BMCC Receiver

As part of ADPM, the receiver maintains the current estimate, $\hat{f}$ of the load factor at the bottleneck on the forward

[2]Note that this corresponds to a virtual queue with capacity $\gamma_l C_l$.

[3]We calculate the average as $q_{av}(t+t_q) = a.q_{av}(t) + (1-a).q(t+t_q)$, where $q(t)$ is the instantaneous queue length, and $t_q \ll t_p$. We set $t_q = 10\,\text{ms}$, $a = 0.875$, and $\kappa_1 = 0.5$ similar to [13].

path. When a packet is received, this estimate is updated as:

$$\hat{f} \leftarrow \begin{cases} u & \text{if } b_{ecn} = (11)_2 \\ h & \text{if } (b_{ecn} = (10)_2 \text{ and } h < \hat{f}) \\ & \text{or } (b_{ecn} = (01)_2 \text{ and } h > \hat{f}) \\ \hat{f} & \text{otherwise} \end{cases}$$

where b_{ecn} refers to the two ECN bits in the IP header of the received packet. The estimate $\hat{f}$ is sent to the sender using TCP options, as described in Section II-E. Note that the receiver's estimate will lag behind the true value [21], except that values over u are signaled immediately to indicate severe overload.

The resolution depends on the fraction of packets that hash to a particular range. For BMCC, values of f below a threshold η_0 are rounded up to η_0, and the hash is such that 1/4 of packets hash to values in (η_0, η) for some design parameter η, 1/4 of packets hash to $(\eta, 1)$ and 1/2 hash to $(1, u)$.

Remark 1: The systemic effect of ADPM is to indicate the maximum load factor to the receiver. On leaving a router, a packet will be marked if the load factor of that router or any upstream router exceeds the packet's hash value. When a marked packet is received, if $h > \hat{f}$ the receiver updates its estimate to h since it is a tighter lower bound on the maximum f. Conversely, the hash of an unmarked packet provides an upper bound on the maximum f on the path.

C. BMCC Sender

To achieve both high bottleneck utilization and fair bandwidth allocation with low fluctuation in rates, BMCC uses three modes of operation, based on whether the load is low ($f < \eta$), medium ($f \in [\eta, 1)$) or overload ($f > 1$).

1) Low Load ($\eta_0 \le \hat{f} < \eta$): To increase bottleneck utilization when the load is low, sources apply MI with factors proportional to $(1 - \hat{f})/\hat{f}$. In particular, flows with RTT equal to t_p apply

$$w(t + T) = w(t)\left(1 + \kappa_2 \frac{1 - \hat{f}}{\hat{f}}\right), \tag{2}$$

where T is the RTT of the flow and κ_2 is the step size. BMCC uses $\kappa_2 = 0.35$, which is in the range $\kappa_2 \in (0, 1/2)$ suggested by Theorem 1 of [13]. VCP [13] approximates this using a fixed MI parameter independent of $\hat{f}$.

BMCC aims to give equal rate to flows with different RTTs. Since flows with large RTTs update less often, the rule

$$w(t + T) = w(t)\left(1 + \kappa_2 \frac{1 - \hat{f}}{\hat{f}}\right)^{T/t_p} \tag{3}$$

is used so that windows grow at a rate independent of T. To mirror the benefits of traditional slow start, new flows remain in MI until $\hat{f}$ first reaches 1,

2) Medium Load ($\eta \le \hat{f} < 1$): When utilization is high, BMCC uses AIMD to achieve fairness. In medium load, sources apply AI:

$$w(t + T) = w(t) + \alpha, \tag{4}$$

with $\alpha = (T/t_p)^2$ chosen to cause the equilibrium window to be proportional to the flow's RTT, giving RTT fairness.

Parameter	Value	Purpose
t_p	200 ms	load factor estimation interval
κ_1	0.5	controls how fast to drain the queue
κ_2	0.35	stability constant
α	$(T/t_p)^2$	AI parameter of flow with RTT T
β_{max}	0.875	maximum back-off factor
β_{min}	0.65	minimum back-off factor
η_0	0.15	initial estimate of load factor
η	0.75	mode threshold

TABLE I
BMCC PARAMETER SETTINGS

3) Overload ($1 \le \hat{f} \le u$): When the load factor is greater than 1, sources use MD:

$$w(t + T) = w(t)\beta(\hat{f}), \tag{5}$$

where the decrease factor

$$\beta(\hat{f}) = \beta_{\max} - \frac{\Delta\beta(\hat{f} - 1)}{(u - 1)} \tag{6}$$

varies linearly in $[\beta_{\min}, \beta_{\max}] \subset (0, 1)$, u is the maximum value of f that ADPM can signal, and $\Delta\beta = \beta_{\max} - \beta_{\min}$. Note that the MD factor is a function of $\hat{f}$. This implies that when $\hat{f}$ is high, sources back off by a greater amount, which improves responsiveness to congestion (see Section IV). Otherwise, sources back off by a smaller amount to reduce fluctuations in sending rates and maintain high utilization.

We now discuss the choice of different parameter values used by BMCC.

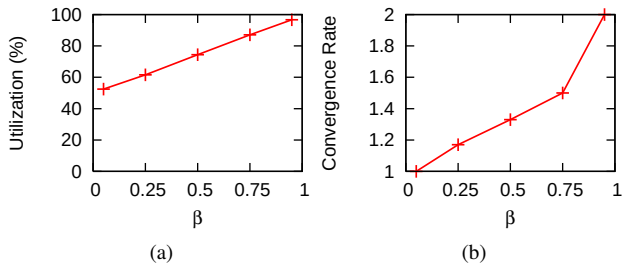
D. Parameter values

Important BMCC parameters are summarized in Table I. Particular choices are discussed below.

1) Measurement interval, t_p: The period t_p should be greater than the RTT of most flows to smooth out sub-RTT burstiness [27], but should be small enough to ensure responsiveness to congestion. The majority of Internet flows have RTTs less than 200 ms [28], [29]. Hence, BMCC uses $t_p = 200$ ms; the same value is used by VCP [13]. See [14] for an analysis on the impact of t_p on performance.

2) Backoff parameter, β: The backoff parameter varies from $\beta_{\min} = 0.65$ when $f = u = 1.2$ to $\beta_{\max} = 0.875$ when $f = 1$, for the following reasons: A very high value of $\beta_{\max}$ (e.g., 0.99) is undesirable as it leads to slow convergence [30], whereas a low value (e.g., 0.2) reduces average throughput[4] and increases variations in flow rates, which is undesirable for real-time applications [31], [32]. To balance these, BMCC uses $\beta_{\max} = 0.875$; the same value is used in the DECbit scheme [33] and VCP [13]. To ensure high responsiveness and fast convergence under high load, $\beta(f)$ is decreased linearly as f increases until $\beta(u) = \beta_{\min}$. The choice of $\beta_{\min}$ determines the range of values that $\beta(f)$ can assume. Figure 1 shows the bottleneck utilization and fairness convergence rate as a function of $\beta_{\min}$ for two long-lived flows on a 100 Mbps link with $T = t_p$. While link utilization increases with $\beta_{\min}$, the

[4]Since BMCC does not maintain a standing queue, reducing the window directly reduces the throughput.



(a) (b)

Fig. 1. Bottleneck utilization and fairness convergence rate as a function of $\beta(f) = \beta_{min}$ for two flows on a 100 Mbps with $T = t_p$.

convergence rate decreases. To balance these, BMCC uses $\beta_{\min} = 0.65$.

3) Mode threshold, η: The minimum value of $\beta(f)$ determines the parameter η, which defines the transition point between applying MI and AIMD. To encourage high utilization, η should be large, but small enough to prevent flows from entering MI after overload detection, which can lead to high packet loss rate. It suffices that $\eta < \min_{f \geq 1}(f\beta(f))$. The minimum of $f\beta(f) = 0.723$ occurs for $f = u$ (since $\beta(f) = \beta_{\min}$ for $f \geq u$, the quadratic $f\beta(f)$ for $f < u$ is concave, and $\beta(1) > u\beta(u)$). BMCC uses $\eta = 0.75$.

E. Reverse Signalling Overhead

The BMCC receiver communicates the estimated load factor, $\hat{f}$, to the sender using TCP options. Unlike on the forward path, TCP options are acceptable for this because they need not be processed by the routers. However, they can introduce overhead. The number of bytes needed in TCP options depends on the desired feedback resolution. If 16-bit feedback resolution is used, as in our implementation, then the feedback fits in a minimum sized (4 byte) TCP option. Rather than piggybacking $\hat{f}$ on every ACK, it is only necessary to send $\hat{f}$ if it changes. This approach (which we call "non-redundant") increases the sensitivity to lost ACKs.

The alternative used by BMCC is to send the estimate immediately after it changes, and then to send redundant copies with decreasing frequency. Each receiver maintains a counter i. The ith option is sent i packets after the previous one, so that options are sent on ACKs 1, 3, 6, 10,... The counter is reset to 1 each time $\hat{f}$ changes and incremented by 1 each time $\hat{f}$ is sent. This scheme is robust against losing a small number of ACKs, but if $\hat{f}$ changes once per n packets, the overhead is only $O(1/\sqrt{n})$ times that of naïvely echoing on every ACK.

The reduction in overhead was evaluated by ns2 simulations of a dumbbell topology with a bottleneck link of capacity 100 Mbps, carrying ten long-lived flows in each direction with T=80 ms. Table II shows statistics that correspond to the average of the ten flows in the forward path. Of 552942 ACKs generated by the receivers, the price estimate changed for 6914 (1.3%), which carried $\hat{f}$ under both schemes, whereas 12.9% carried $\hat{f}$ under BMCC's robust scheme.

	ACKs sent	ACKs with $\hat{f}$	Reduction(%)
non-redundant	552942	6914	98.7
BMCC	552942	71476	87.1

TABLE II
OVERHEAD OF SIGNALLING FROM RECEIVER TO SENDER.

III. Design Issues

The design of BMCC gives rise to several questions whose answers requires careful consideration of network load and operating conditions. The objective of this section is to address these questions and describe the approaches BMCC uses to manage traffic efficiently and achieve high network performance.

a) What is the congestion level assumed by new flows?: ADPM needs an initial estimate of the congestion level. New flows initially estimate $\hat{f} = \eta_0 = 0.15$, and thus increase their windows by a factor of $1 + \kappa_2(1 - \eta_0)/\eta_0 \approx 3$ per t_p. This is a faster increase than existing TCP slow start when $T = t_p$.

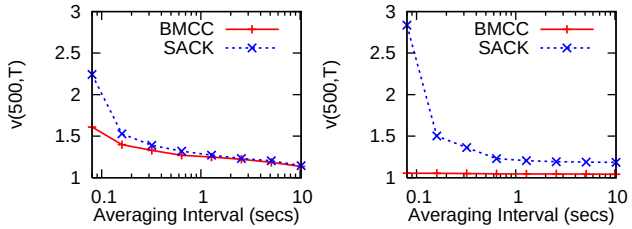
b) Can new flows cause overload before ADPM has been able to signal congestion?: The estimates of f signalled by ADPM lag behind the true value. Hence, a flow may not detect overload and apply MD in a given t_p interval [20]. In the presence of a large number of flows, congestion will be avoided if most reduce their windows, even if some miss the congestion signal. However, if $f > u = 1.2$, each flow decreases its window deterministically (using the standard ECN "Congestion Experienced" codepoint) which prevents persistent congestion. Note that ADPM provides feedback well before the buffer overflows, and so new flows need not cause large bursts of loss and timeouts, even if there is some delay in detecting congestion.

c) Sources may apply different backoff factors at the same time; does this lead to unfairness?: The backoff factor applied by a BMCC source depends on its estimate of the load factor. This estimate can be different across sources due to ADPM. This may lead to short-term unfairness (on the scale of a few RTTs). To quantify this, we compare the level of unfairness caused by TCP SACK and BMCC for a range of time scales.

Consider an averaging interval s. For two SACK flows, let $X_i(t, s)$ be the average rate of flow i over the interval $(t, t+s)$, and let the "unfairness rate" be

$$v(\tau, s) = \frac{1}{\tau} \int_0^\tau \frac{\max_i(X_i(t,s))}{\min_j(X_j(t,s))}\, dt,$$

where $i, j \in \{1, 2\}$ and τ is the total observation period. Figure 2 shows $v(500, s)$ against the averaging interval s for two link capacities and $T = 80$ ms (The buffer size was set to the path bandwidth-delay product). Observe that the v curve for BMCC remains below that of SACK implying that it is always fairer than SACK on short time scales. The unfairness rate is higher on the 1 Mbps link due to higher average f in overload. Since $\hat{f}$ can vary randomly across sources due to ADPM and backoff is a function of f, this increases the range of backoff factors that can be applied by the sources; leading to higher v values. We validated this using ns-2 simulations which showed that the average f in overload was 105.5% and



(a) C=1 Mbps (b) C=45 Mbps

Fig. 2. Unfairness rate as a function of the averaging interval ($T = 80$ ms)

100.5% and the average backoff factors were 0.84 and 0.87 corresponding to 1 Mbps and 45 Mbps links, respectively.

d) Why use a higher Mode threshold when flows start?: A long-lived BMCC flow uses AIMD in steady-state. In the presence of such a flow, a newly arriving flow would normally apply AIMD too, which can cause slow convergence. To help improve convergence in this scenario, a larger initial η is used due to the availability of high resolution load estimates. This allows new BMCC flows to apply the more aggressive, load-factor guided MI until the end of the first congestion epoch.

IV. BMCC Protocol Analysis

In this section, we analyse two important properties of BMCC. The first deals with the rate of convergence to fairness when a new flow starts competing with multiple long-lived BMCC flows, while the second is concerned with the rate at which a new BMCC flow acquires the spare (unutilized) bandwidth. The main objective of the analysis is to derive conservative, yet sufficiently accurate, estimates of these rates to guide BMCC parameter selection.

A. Convergence to fairness

To estimate the rate at which a new flow achieves approximately its "fair share" of the capacity of a single bottleneck, we consider a network model, where N BMCC flows, indexed by $n = 1, \ldots, N$, share a single high BDP link. We model this as an AIMD system with deterministic rate increases, and random back-off factors, similar to the ones proposed in [30] and [34].

1) Model: Let $\mathbb{R}_+ = [0, \infty)$ and let $(\cdot)'$ denote transpose. The state of the system is a vector $w(k) \in \mathbb{R}_+^N$ of window sizes indexed by a discrete time variable k. All amounts of data will be measured in units of packets (assumed to be equal for all flows), and all times will be measured in units of the slot, t_p.

Model the rate of flow n as $y_n(k) = w_n(k)/T_n$, where T_n is the RTT of the nth flow, in slots of length t_p. Queueing delay is ignored since BMCC's virtual queue causes queueing to be negligible. Define a backoff event to be a slot in which the total arrival rate $\lambda(k) = \sum_{n=1}^{N} y_n(k)$ satisfies $\lambda(k) \geq \gamma_l C_l$, and let $\mathcal{T}_i$ be the time slot of the ith backoff event.

By (4), $\alpha_n = T_n^2$ is the amount by which w_n increases every RTT T_n when no backoff occurs. We model this increase as an increase by $\hat{\alpha}_n = T_n$ per timeslot; this is the same average rate, differing only in quantization.

In each time slot, the router calculates a load factor. When there is no backoff, each flow i increases its rate by approximately $\hat{\alpha}_n/T_n = 1$ packet per time slot. However, due to factors such as operating system jitter and integer arithmetic, this is not the exact increase in each time slot. It is reasonable to model the increase as $\hat{\alpha}_n/T_n + \Delta$ where Δ is a zero-mean random variable, independent of the history. If the period between backoff events is much larger than one slot (which it is for a large BDP link) then the sum of the rates modulo N becomes a mixing process. Since the randomness of Δ has a mixing effect, we appeal to the intuition that non-intrusive mixing arrivals see time averages (NIMASTA, [35]) to make the modeling assumption that this remainder modulo N has its time-average distribution, namely $U[0, N]$.

The arrival rates (measured in packets/slot) in the backoff time slots $\mathcal{T}_i$ are modelled as i.i.d. random variables. We assume that in each backoff event, at least one flow reduces its rate enough to ensure that the following slot is not also a backoff slot. Then the arrival rate at $\mathcal{T}_i$ can be modeled as

$$\lambda(\mathcal{T}_i) \sim \gamma_l C_l + U[0, N]. \tag{7}$$

Since the queue was empty in the slot before the backoff,

$$q_{av}(\mathcal{T}_i) = (\lambda(\mathcal{T}_i) - C_l)^+ \in [0, N].$$

The load factor is then

$$\begin{aligned} f(\mathcal{T}_i) &= \frac{\lambda(\mathcal{T}_i) + \kappa_1 q_{av}(\mathcal{T}_i)}{\gamma_l C_l} \\ &= 1 + \Phi \end{aligned}$$

where Φ is piecewise uniform on $[0, \min(N, C_l(1-\gamma_l))]$ and $[\min(N, C_l(1-\gamma_l)), N]$. For simplicity, we take as a model

$$f(\mathcal{T}_i) = 1 + \phi U[0, N] \tag{8}$$

where $\phi \in [1/C_l, (1+\kappa_1)/C_l]$. We consider the case that

$$N\phi < u - 1 \tag{9}$$

so that backoff is only caused by ADPM, and not by receiving a $(11)_2$ ECN mark.

Each flow has a probability of not backing off. Although that depends on the instantaneous rate, we model it as dependent only on f, as follows. Let c be the fraction of hash values in the interval $[1, u]$. (In Section II, $c = 1/2$.) Let $p_f = c\frac{f-1}{u-1}$ be the probability that a given packet's hash value is in the range $[1, f]$ corresponding to overload. For a flow sending d packets per slot, the probability of detecting overload in a given slot given a true load factor f, is

$$P(\hat{f} \geq 1 | f) = 1 - (1 - p_f)^d \tag{10}$$

Since the probability of backing off increases with the rate d, it is conservative to make the simplifying approximation that each flow instead backs off with probability at most the average backoff probability.

This requires that we find a d such that

$$\left(1 - c\frac{f-1}{u-1}\right)^d \geq \frac{1}{N}\sum_{n=1}^{N}\left(1 - c\frac{f-1}{u-1}\right)^{y_n(i)}$$

for all i. By Lemma 1 in the appendix, it is sufficient that d satisfy this for the largest f, namely $f = 1 + N/\phi$. Since the right hand side is larger when y_n are more spread out, and we expect y_n to converge, it seems conservative to set d to satisfy this for $y_n(0)$. Thus we take

$$d = \frac{\log\left(\frac{1}{N}\sum_{n=1}^{N}\left(1 - c\frac{N\phi}{u-1}\right)^{y_n(0)}\right)}{\log\left(1 - c\frac{N\phi}{u-1}\right)}. \quad (11)$$

A simpler and more insightful, though less conservative, model would simply use

$$d = \frac{\gamma_l C_l}{N}.$$

We model flows' $\hat{f}$ at each stage as having identical marginal distribution satisfying (10), and a joint distribution such that $\hat{f} > 1$ for at least one flow, independent of previous backoffs. By (6), this implies that flow n backs off by a factor β_n, where the vectors $\beta(\mathcal{T}_i) = (\beta_1(\mathcal{T}_i), \ldots, \beta_N(\mathcal{T}_i))'$ at different back-off instants are i.i.d. random variables. Flows n for which $\hat{f} < 1$ have $\beta_n = 1$. Then the dynamics of the entire network of sources are described by [30]

$$Y(i+1) = A(i)Y(i) \quad (12)$$

where

$$A(i) = \mathrm{diag}(\beta(\mathcal{T}_i)) + \frac{1}{N}e(e' - \beta'(\mathcal{T}_i)), \quad (13)$$

$Y(i) = [y_1(i), y_2(i), ..., y_N(i)]'$ and $e = (1, \ldots, 1)' \in \mathbb{R}^{N\times 1}$.

2) Rate of Convergence: We will now use the model (6)–(13) to determine the rate of convergence to fairness.

The main result of this section is

Theorem 1: Under the model given by the numbered equations (6)–(13), the expected values of the rates converge as

$$\mathbb{E}[Y(i)] - \frac{\gamma_l C_l}{N}e = \mu^{i-1}\left(\mathbb{E}[Y(0)] - \left(\frac{\gamma_l C_l}{N} + \frac{1}{2}\right)e\right) \quad (14)$$

where

$$\mu = \beta_{\max}(1 - Z(d+1)) + (1 + M\phi N)Z(d+1) +$$
$$M\left(\frac{Z(d+2)-1}{\zeta(d+1)} - \frac{\phi N}{2}\right), \zeta = \frac{c}{u-1}, M = \frac{\Delta\beta}{2(u-1)},$$

and

$$Z(x) = \frac{1 - (1 - \zeta\phi N)^x}{\zeta x \phi N}.$$

Proof: Since β, and hence A, are i.i.d.,

$$\mathbb{E}[Y(i)] = \prod_{j=0}^{i-1} A(j)Y(0) = (\mathbb{E}[A(0)])^{i-1}\mathbb{E}[Y(0)]. \quad (15)$$

The convergence is thus determined by the eigenvalues of $\mathbb{E}[A(0)]$. The dominant eigenvalue of $\mathbb{E}[A(0)]$ is 1, with eigenvector e. This implies that the component of $Y(i)$ in the direction of e is constant, and by (7) is equal to $(\gamma_l C_l/N + 1/2)e$. All other eigenvalues are $\mathbb{E}[\beta(\hat{f})]$. It remains to show that $\mathbb{E}[\beta(\hat{f})] = \mu$.

Recall that $\hat{f}$ is the estimate of f at a given backoff time. The expected backoff factor in overload conditioned on f is

$$\mathbb{E}[\beta(\hat{f})|f] = P(\hat{f} < 1|f)\mathbb{E}[\beta(\hat{f})|\hat{f} < 1, f] +$$
$$P(\hat{f} \geq 1|f)\mathbb{E}[\beta(\hat{f})|\hat{f} \geq 1, f]. \quad (16)$$

Now $\mathbb{E}[\beta(\hat{f})|\hat{f} < 1, f] = 1$ since sources do not backoff when $\hat{f} < 1$, and by (10), $P(\hat{f} < 1|f) = (1 - p_f)^d$.

Finally, by (6) and (8),

$$\mathbb{E}[\beta(\hat{f})|\hat{f} \geq 1, f] = \frac{1}{f-1}\int_0^{f-1}(\beta_{\max} - \frac{\Delta\beta\psi}{u-1})d\psi$$
$$= \beta_{\max} - \frac{\Delta\beta(f-1)}{2(u-1)}. \quad (17)$$

Let $\zeta = \frac{c}{u-1}$. Then $\mathbb{E}[\beta(\hat{f})|f]$ can be averaged over $f - 1 \sim \phi U(0, N)$ to give

$$\mathbb{E}[\beta(\hat{f})] = \frac{1}{\phi N}\int_0^{\phi N}[(1 - (1 - \zeta\psi)^d)\times \quad (18)$$
$$\left(\beta_{\max} - \frac{\Delta\beta\psi}{2(u-1)}\right) + (1 - \zeta\psi)^d]d\psi$$

$$= \beta_{\max}(1 - Z(d+1)) + (1 + M\phi N)Z(d+1)$$
$$+M\left(\frac{Z(d+2)-1}{\zeta(d+1)} - \frac{\phi N}{2}\right)$$

■

Figure 3 shows the average window size of two BMCC flows that have an inter-arrival time of 50 s. The bottleneck capacity is 20 Mbps and $T = t_p$. The data points corresponding to the 'Model' curves are determined as follows: The rates in the model evolve in terms of congestion epochs. To plot the window versus time in Figure 3, we assume that each epoch has the mean duration. To calculate the latter we divide the reduction in the sum of window sizes every backoff slot (assuming independence) by the increase in the sum of window sizes per slot. Each data point corresponding to the 'Measured' curve represents the average of 200 simulation runs with random flow starting times $t_1 \sim U[0, t_p]$ and $t_2 \sim 50U[1, 1 + t_p]$, where t_1 and t_2 are the starting times of Flow 1 and Flow 2, respectively. For the model, each flow uses $d = C_l/N$. Observe that the expected window size of the flows converge to the same value and the model conservatively estimates the rate of convergence.

Impact of u on backoff: The value of u determines the maximum value of f that can be signalled by ADPM. Figure 4(a) shows that the expected backoff factor $\mathbb{E}[\beta(\hat{f})]$ increases with u. This happens because p_f, which is the probability that a packet detects overload, decreases with u (see Figure 4(b)). Observe that when $u = 1.2$, flows detect overload with probability close to 1.

B. How Fast does BMCC Acquire Spare Bandwidth?

A well known problem of TCP SACK is that a flow sending on a path with large BDP and spare bandwidth takes too long to start up [6], and then causes many packet losses when the window finally reaches the BDP [36]. BMCC addresses

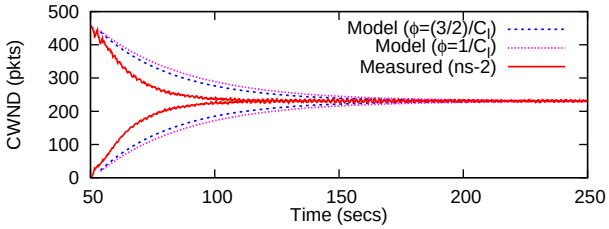
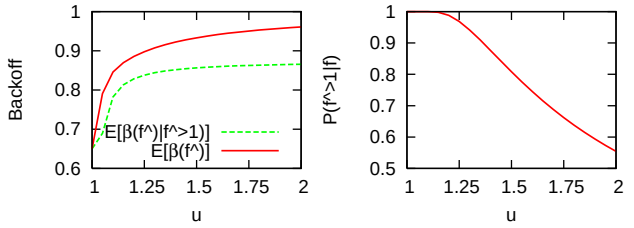


Fig. 3. Comparison of the convergence rate of two flows that start with an inter-arrival time of 50 s. The bottleneck capacity is 20 Mbps, $T = t_p$, and the buffer size is set to the path BDP.



(a) Average Backoff (b) Probability Overload Detection

Fig. 4. $\mathbb{E}[\beta(\hat{f})]$, $\mathbb{E}[\beta(\hat{f})|\hat{f} \geq 1, f]$, and $P(\hat{f} \geq 1|f)$ as a function of u, where C =10000 pkts/sec, $N = 550$, $\kappa_1 = 0.5$, and $\phi = (3/2)/C_l$.

these by increasing its rate faster until incipient congestion is explicitly signaled, and then slowing down the rate of increase.

To obtain insights into the rate at which a new BMCC flow acquires idle capacity, we use a simplified fluid approximation model. This analysis applies to both UNO [24] and MPCP [25]. We consider a single bottleneck being traversed by a BMCC flow with a constant RTT, $T = t_p$. We assume the flow window size $w(t)$ is a positive, continuous, and differentiable function and that the BDP is large enough for ADPM to accurately estimate the load factor.

Because a new BMCC flow skips the AI mode, it adapts its window in two phases until the first congestion epoch. In the first, with $f = w/(C_l T) < \eta_0$, it increases the window by a factor of $1+\kappa_2(1-\eta_0)/\eta_0 \approx 3$ each RTT. In the second, with $f \in (\eta_0, 1)$, the window increases by a factor of $\kappa_2(1-f)/f = \kappa_2(C_l T - w)/w$ each RTT. These phases can be modeled as:

$$\frac{dw}{dt} = \frac{\log(3)}{T} w(t), \qquad f < \eta_0 \tag{19}$$

$$\frac{dw}{dt} = \frac{\kappa_2}{T}(C_l T - w(t)), \qquad f \in (\eta_0, 1) \tag{20}$$

Under the model (19), (20), (1), with $w(0) = 1$, the amount of data $d(t) = \int_0^t w(\tau)/T \, d\tau$, delivered by time a t before the load factor reaches 1 is

$$d(t) = \begin{cases} 3^{t/T}/\log(3) & t \leq t_1 \\ d(t_1) + C_l t - (1-\eta_0)e^{-\kappa_2 \Delta t_1/T}/\kappa_2 & t > t_1 \end{cases}$$

where $t_1 T \log_3(\eta_0 C_l T)$ and $\Delta t_1 = t - t_1$. Solving (19) and applying $w(0) = 1$ gives

$$w(t) = 3^{t/T}, \quad t \leq t_1 \tag{21}$$

for the first phase. For the second phase, solving (20) an applying the continuity of w at t_1 gives

$$w(t) = C_l T \left(1 - (1-\eta_0)e^{-\kappa_2(t-t_1)/T}\right) \quad t > t_1 \tag{22}$$

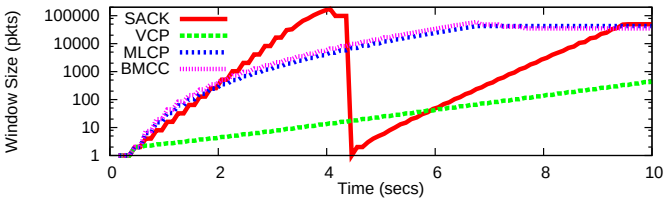


Fig. 5. Comparison of the growth rate of the congestion window sizes for SACK, VCP, MLCP and BMCC on a 2 Gbps link with $T = 200$ ms

Integrating over $[0, t]$ and applying the initial condition $d(0) = 0$ and continuity of d at t_1 establishes the result. Note that, unlike regular multiplicative increase, the window under (22) undergoes *concave* negative exponential growth.

Similar calculations show that SACK sends data

$$d_R(t) = 2^{t/T}/\log(2)$$

by $t < T \log_2(C_l T)$, while VCP sets $\Upsilon = 1.0625$ and sends

$$d_V(t) = \begin{cases} \Upsilon^{t/T}/\log(\Upsilon) & t \leq t_{1,V} \\ d_V(t_{1,V}) + \Delta t_{1,V}(\Upsilon^{t_{1,V}/T} + \Delta t_{1,V}) & t > t_{1,V} \end{cases}$$

where $t_{1,V} = T \log_\Upsilon (C_l T \eta_v)$ and $\eta_v = 0.8$.

Figure 5 compares the growth of the congestion window of a single SACK, VCP, MLCP and BMCC flow starting on an idle 2 Gbps link of RTT=200 ms. The buffer is set to the BDP of 50000 pkts. The SACK window doubles every round-trip time (a straight line on the exponential plot in Figure 5). When it fills the router buffer, it has a window size of ∼100000 pkts. This doubles in the next RTT, resulting in a loss of ∼90000 pkts and consequent timeouts. In the first 2 s, BMCC attains a larger window than SACK. As the available bandwidth decreases, BMCC slows its growth and avoids any loss. It achieves similar growth to that of MLCP [14], using only the existing feedback bits. In contrast, VCP is too conservative; by 6.5 s, the BMCC window is the BDP, but the VCP window is well under 1% of BDP.

V. Performance Evaluation

To evaluate the performance and fairness of BMCC in diverse network settings, a ns-2 based simulation study is carried out. In this study, the focus is on comparing the performance of BMCC with multiple protocols designed to overcome the limitations of TCP in high BDP networks including XCP [1], RCP [6], VCP [13], MLCP [14], UNO [24], MPCP [25], TCP SACK, CUBIC [2], and CTCP [5]. The interaction of BMCC with TCP SACK is investigated in Section VI.

To ensure fair comparison with all protocols which obtain router support, TCP SACK, CUBIC, and CTCP are always used in conjunction with RED and ECN. Furthermore, the bottleneck buffer size is set to the larger of the BDP (i.e., the product of the bottleneck capacity and the round-trip propagation delay) or two packets per-flow. Data packet are 1000 bytes, while the ACKs are 40 bytes. All simulations are run for 100 s except those in which the round-trip propagation delay is varied. In this case, the simulations are run for 500 s. The results represent the average of ten simulation runs, where

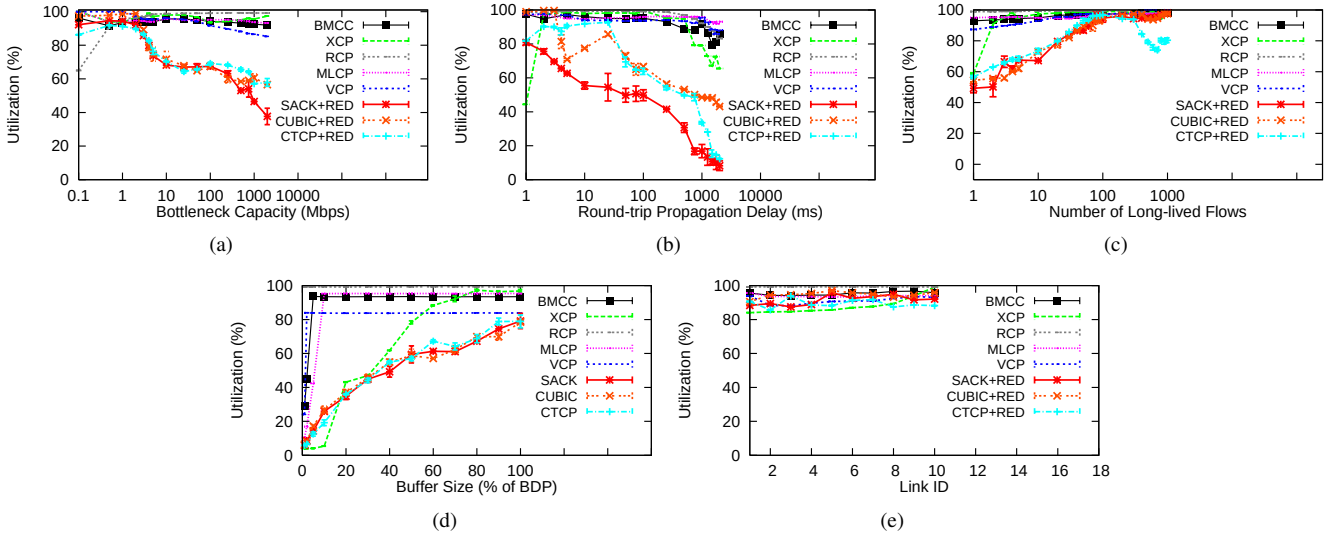


Fig. 6. Bottleneck utilization with varying capacity, RTT, number of flows, and buffer size on a single bottleneck (Fig. 6(a)-(d)) and with varying number of bottlenecks on a multiple bottleneck topology (Fig. 6(e)). BMCC, like XCP, RCP, MLCP, and VCP, maintains high utilization across a range of scenarios. It overcomes the limitations of SACK, CUBIC, and CTCP using RED/ECN. Note that BMCC achieves $\geq 95\%$ utilization even with buffers as small as 5% of the path BDP. Below this, utilization is low for all protocols.

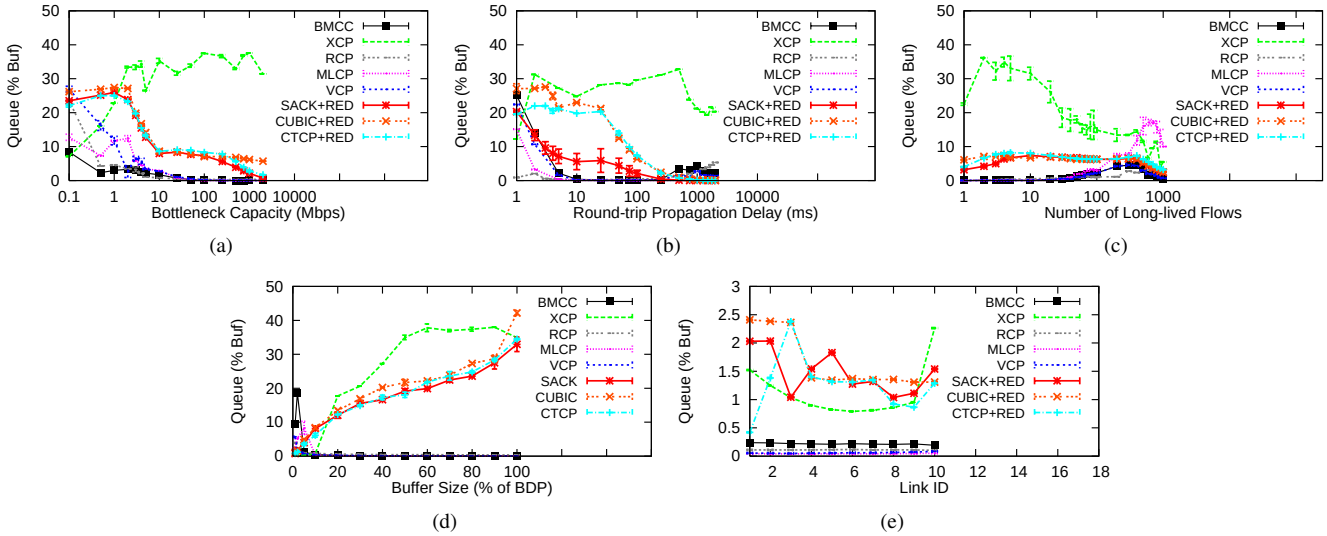


Fig. 7. Bottleneck queue with varying capacity, RTT, number of flows, and buffer size on a single bottleneck (Fig. 7(a)-(d)) and with varying number of bottlenecks on a multiple bottleneck topology (Fig. 7(e)). BMCC, like RCP, MLCP, and VCP, maintains low bottleneck queue ($< 10\%$) in all scenarios expect when RTT ≤ 1 ms or the buffer size is $<5\%$ of the path BDP. XCP, SACK, CUBIC, and CTCP result in a much larger queue length in several scenarios. Note that RED/ECN is not used when the bottleneck buffer size is varied.

in each run the starting time of flows is chosen uniformly at random from $[0, 2]$ s. For each result, we also determine the 95% confidence interval. The statistics neglect the first 5% of the simulation time.

A. Single Bottleneck Topology

We first consider several variants based on a dumbbell topology with a 155 Mbps bottleneck link and 80 ms round-trip propagation delay. Unless stated otherwise, we use two-way traffic with five FTP flows on both the forward and reverse paths. We vary different parameters and study their impact on performance.

a) Varying Bottleneck Capacity: First the bottleneck capacity is varied between 100 kbps and 2 Gbps, keeping everything else fixed. Figures 6(a), 7(a), and 8(a) show that BMCC maintains high utilization ($\geq$90%), with small queues ($<$20% BDP) and negligible packet loss. While VCP, MLCP, XCP and RCP all also achieve $\geq$85% utilization, average utilization for SACK, CUBIC, and CTCP remains below 70% for links faster than 10 Mbps. As the link capacity increases, these protocols take longer to converge. Consequently, the average utilization decreases with link capacity as all simulations are run for a fixed duration of 100 s. In addition, the presence of reverse traffic exacerbates this as it increases the RTT. Note that since buffers are sized based on round-trip propagation delay, which does not account for the queueing delays on the forward and reverse paths, they are not sufficient for achieving full link utilization. CUBIC and CTCP achieve about 20%

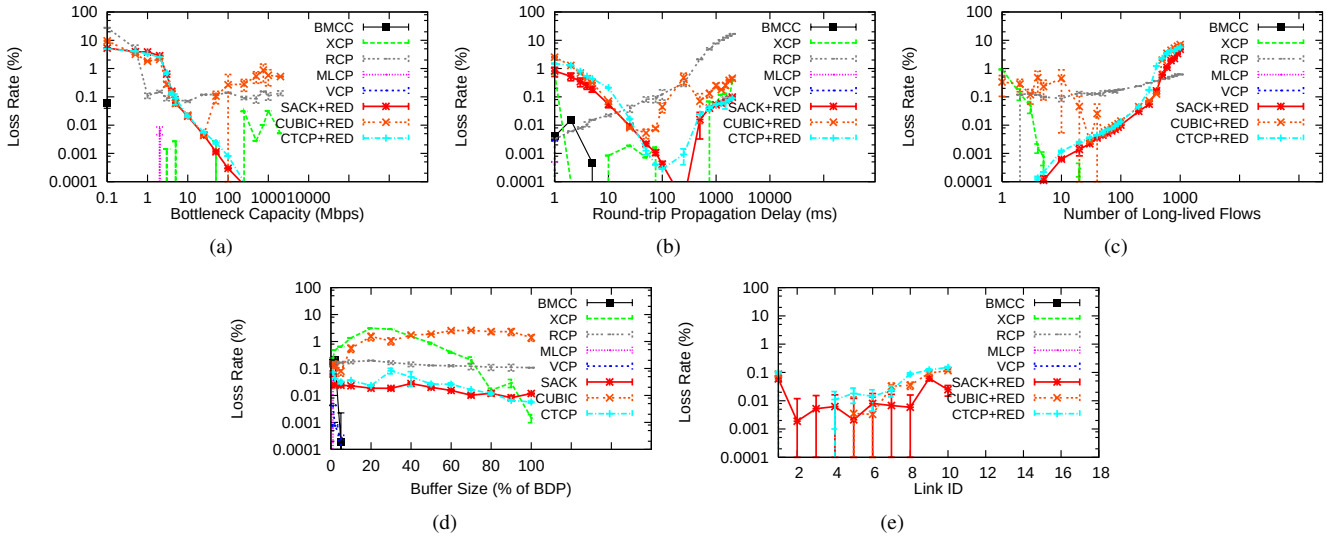


Fig. 8. Loss rate with varying capacity, RTT, number of flows, and buffer size on a single bottleneck (Fig. 8(a)-(d)) and with varying number of bottlenecks on a multiple bottleneck topology (Fig. 8(e)). BMCC, like MLCP and VCP, introduces low loss rate ($< 0.0001\%$) in all scenarios except when RTT $\leq$ 1 ms, capacity is $\leq$ 0.1 Mbps, and the buffer size is $< 5\%$., in which loss rate is up to 0.1%. In many scenarios, no loss was observed with BMCC. XCP, RCP, SACK, CUBIC, and CTCP result in larger loss rates in several scenarios.

higher average utilization than SACK when the link capacity is 2 Gbps, because they are designed to be more aggressive than SACK on large BDP paths. RCP, SACK, CUBIC, and CTCP result in high loss rates for small capacities, despite the use of ECN, whereas all other schemes experience negligible loss rates. The average queue length with XCP (5∼40% BDP) is much higher than other schemes for link capacities higher than 1 Mbps. This was also observed by [13] but the reason for this is not clear. We conjecture that this is due to the presence of reverse traffic.

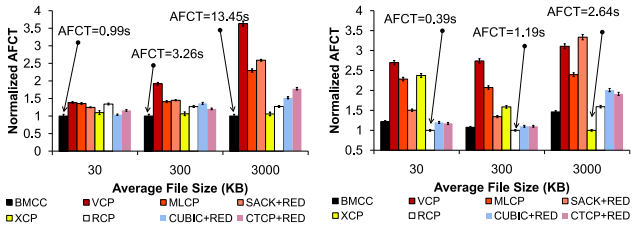
b) Varying Feedback Delay: Next the round-trip time is varied between 1 ms and 2 s. Figures 6(b), 7(b), and 8(b) show that BMCC, MLCP, and VCP again maintain high utilization ($\geq$80%), with small queues ($\leq$25% BDP) and negligible packet loss. While CUBIC and CTCP outperform SACK across a range of RTTs, their average utilization remains $<$70% for RTTs larger than 100 ms. XCP results in a higher average queue length (20∼35% BDP) than other schemes and CUBIC and CTCP lead to an average queue length of about 20% for RTTs less than 25 ms. For low RTTs (e.g., $<$2 ms) the average queue length for BMCC, MLCP and VCP rises to about 5∼25% BDP, because the AI rate becomes large relative to the BDP. Conversely, for large RTTs, RCP results in a loss rate of up to 15% due to its aggressive rate allocation.

c) Varying Number of Long-lived Flows: The number of long-lived flows is now varied from 1 to 1000. Figure 6(c) shows that while BMCC, MLCP, VCP, XCP and RCP again maintain high utilization ($\geq$90%), SACK, CUBIC, and CTCP achieve less than 90% utilization when there are fewer than 40 flows. For most protocols, the average queue length remains below than 10% (Figure 7(c)). It is higher for XCP (∼10%−40%) and in some cases for MLCP (up to ∼20%). Note that the loss rate for SACK, CUBIC, and CTCP becomes more than 5% for 1000 flows as shown in Figure 8(c).

d) Pareto-Distributed Traffic: To study how BMCC handles typical bursty traffic, we generate short-lived flows with Pareto distributed lengths (shape=1.4) which arrive as a Poisson process, as studied in [6]. We vary the average file size from 30 kB to 3000 kB on bottleneck capacities of 10 Mbps and 100 Mbps and measure the average file completion time (AFCT). Figure 9 shows the AFCT normalized by the smallest AFCT, as a function of the average file size. On a 10 Mbps link, BMCC outperforms all schemes across a range of file sizes. This is noteworthy, since RCP was designed to optimize this performance measure. VCP, SACK, MLCP, CTCP, and CUBIC stretch the AFCT by factors of approximately 3.5, 2.5, 2, 1.8, and 1.5 over BMCC, respectively. On a 100 Mbps link, RCP performs best when the file size is 30 kB and 300 kB. XCP outperforms other schemes when the average file size is 3 MB. Note that BMCC is the second best performing scheme in all cases. For an average file size of 30 kB, VCP, XCP and MLCP stretch the AFCT by factors of about 2.7, 2.4 and 2.3.

VCP results in the highest AFCT because of chooses a conservative MI factor, due to its 2-bit feedback. BMCC obtains load factor estimates of up to 16-bit resolution which allows larger MI factors in low-load. With SACK's slow start algorithm, flows use a fixed MI factor of two. In high load, this is too aggressive, and leads to larger average queue length and higher loss rate, which increases the AFCT. CUBIC and CTCP's use of delay-based congestion adaptation improves convergence speed, which improves their AFCT relative to SACK. XCP results in a large AFCT for smaller file sizes because newly arriving flows apply AIMD. RCP gives higher rates to new flows which helps short flows to finish quickly.

e) RTT Fairness: We now investigate the fairness of BMCC flows with different round trip times. We consider a single 60 Mbps bottleneck link with 20 long-lived flows in each direction. Forward flow $j \in \{1, \ldots, 20\}$ has RTT $T_j = 40 + 4j\delta$ ms, where δ varies between 1 ms and 5 ms.



(a) $C_l = 10$ Mbps (b) $C_l = 100$ Mbps

Fig. 9. Normalized AFCT as a function of the average file size for bottleneck capacities of 10 Mbps and 100 Mbps. The arrows indicate the scheme with the best AFCT whereas the arrow labels show the actual AFCT. The normalized AFCT is defined as the ratio of the AFCT of a scheme and the smallest AFCT across all schemes.

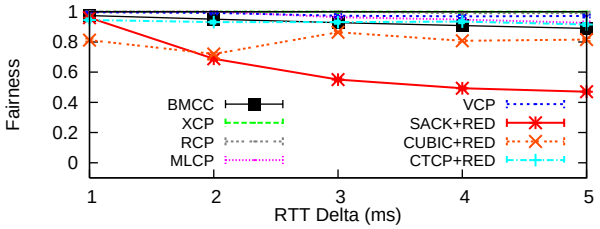


Fig. 10. Jain's fairness index $[(\sum_{i=1}^{N} x_i)^2/(N\sum_{i=1}^{N} {x_i}^2)$, where x_i is the throughput of flow i and $i \in \{1, .., N\}]$ as a function of δ.

When δ is 1 ms, RTTs are in the range [40 ms, 116 ms]. When δ is 5 ms, the RTTs are in the range [40 ms, 420 ms]. Observe that BMCC, MLCP, VCP, XCP, RCP, and CTCP achieve a Jain fairness index of at least 0.9 across a large range of RTT variations (see Figure 10). With CUBIC, the fairness varies in [0.7, 0.9], however, SACK becomes very unfair as δ is increased.

f) Varying Bottleneck Buffer Size: Since all the protocols considered have oscillatory window dynamics, the buffer size affects performance greatly. Figures 6(d), 7(d), and 8(d) show the performance as the buffer varies between 0 and 100% of the BDP. For buffers at least 5% of the BDP, BMCC and RCP maintain more than 95% bottleneck utilization, small queues, and negligible loss. To achieve this utilization, XCP requires more than 70% of BDP, while SACK, CUBIC, and CTCP only achieve 80% utilization even with 100% BDP buffers, due to reverse traffic. When the buffer size is less than 2%, link utilization degrades for all protocols, because even short-term packet bursts cannot be smoothed.

B. Multiple Bottleneck Topology

Many protocol issues are only apparent in multi-link topologies. We now consider a linear topology with ten bottleneck links, where 30 long-lived flows traverse all links and each links is also used by five cross long-lived flows. The capacities of the links follow a stair-case pattern with link $i \in \{0, \ldots, 9\}$ having capacity $(155 - 10i)$ Mbps. Figure 6(e) shows that BMCC, MLCP, and RCP all achieve at least 94% utilization, and SACK, CUBIC, CTCP, and VCP vary between 85%–95%. While XCP achieves at least 83% utilization on average, its utilization is lower on some links ($i < 9$) as a side effect of its "bandwidth shuffling" used to obtain fairness [37] All

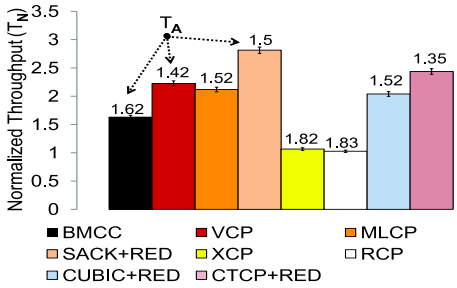


Fig. 11. Comparison of T_N, the average throughput achieved by cross flows, normalized by the throughput of forward flows. The number above the bars represents T_A, the average throughput of forward flows in Mbps.

protocols maintain low average queue length (<3% BDP) and experience low loss (<0.25%) as shown in Figure 7(e) and 8(e).

Per-Flow Throughput: Figure 11 shows the average throughput achieved by cross flows, normalized by the average throughput of forward flows. Under XCP and RCP, which aim for max-min fairness, forward and cross flows achieve similar throughput. The remaining protocols give higher throughput to the flows with fewer hops, more in line with goals such as proportional fairness [15]. Of these remaining protocols, BMCC is the closest to max-min fair to forward flows, since ADPM signals the highest level of congestion of any link on a flow's path.

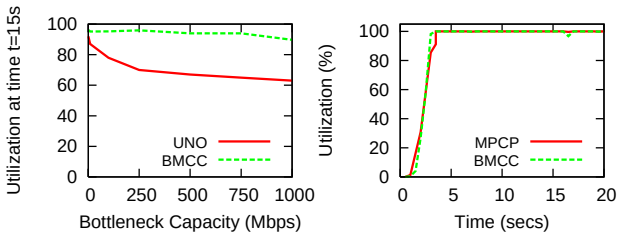
C. Comparison with UNO and MPCP

UNO [24] and MPCP [25], which were developed simultaneously with BMCC, address a similar problem to BMCC but differ from it in the following ways. UNO uses a variant of [22] to control a MI-AI-MD algorithm, but limits itself to 3-bit resolution. This causes it to use a constant MD factor, which does not allow it to both respond quickly to congestion and also avoid large oscillations during AI-MD. MPCP and DCP-EW [38] use a more complex, though more fragile, signaling method; MPCP achieves 4-bit resolution after observing only two packets, but requires per-flow state in the routers, and uses both ECN bits in a way incompatible with the standard [12], both of which make deployment infeasible. BMCC avoids these issues by using the simple, precise and robust ADPM marking scheme, allowing MI-AI-MD window updates in which both the MI and MD factors adapt to the load factor.

We now compare the performance of BMCC with published results for UNO [24] and MPCP [25]. Figure 12(a) shows the bottleneck utilization 15 s after the start of a single BMCC or UNO flow over a path with RTT 200 ms using data from Figure 4 of [24]. BMCC achieves up to 30% higher utilization than UNO, because its higher feedback resolution allows it to use more aggressive MI. Figure 12(b) shows that BMCC and MPCP give similar bottleneck utilization on a 10 Mbps link with RTT 400 ms, using data from Figure 3 of [25]. However, MPCP remains less easily deployed.

VI. Incremental Deployment

There are many challenges in deploying new congestion control mechanisms [39]. One of those is how to signal



(a) BMCC and UNO. (b) BMCC and MPCP.

Fig. 12. Comparison of BMCC with UNO and MPCP.

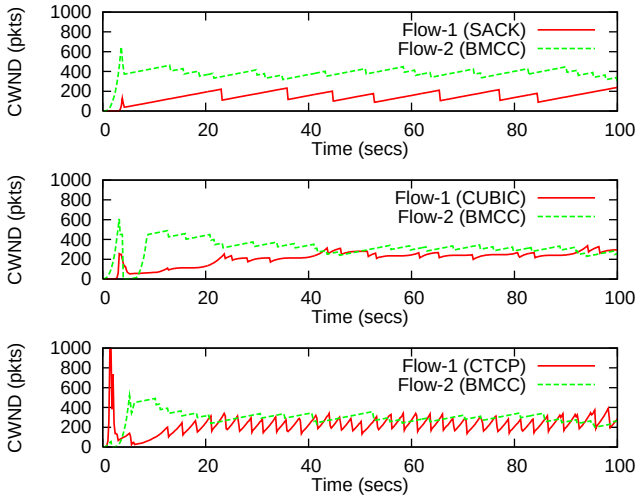


Fig. 13. BMCC and {SACK, CUBIC, CTCP} flows sharing a bottleneck link of capacity 45 Mbps with a modified BMCC router, with RTT $T = 100$ ms.

congestion. Many protocols that use explicit feedback from the network require additional fields, either in an IP option, a TCP option [40] or modified header [1], or a "shim layer" [6]. This makes the universal deployment of such protocols challenging because many routers are configured to drop packets containing IP options [41], and IP payloads may be encrypted. BMCC is suited to deployment in the Internet because its signalling is compatible with the existing IP header.

However, for any new transport protocol to be incrementally deployable on the Internet, it should also be able to (a) run over existing infrastructure, which does not necessarily provide the feedback expected by the protocol; and (b) co-exist with existing protocols, such as SACK, CUBIC, and CTCP, without leading to starvation or causing one protocol or the other to achieve unfairly low throughput. We now describe some modifications to BMCC that could address these issues.

As a stop-gap to address the former issue, we propose that BMCC revert to SACK-like window adjustment when the heuristics of [42] indicate that the bottleneck is not BMCC-enabled, as done by high-speed TCP variants [2], [5] when a path has a low BDP. While this is clearly suboptimal, it may be useful for migration towards more efficient congestion control.

When the bottleneck is BMCC-enabled, BMCC flows can get starved by TCP traffic because algorithms such as TCP SACK seek to fill router buffers. This leads to a high load factor, causing BMCC sources to back-off frequently. To address this, an alternative would be to replace the use of the $(11)_2$ mark to indicate load factor $f > u$, by its use to signal standard AQM (such as RED/ECN) marking. The sender would be modified to back off only when a $(11)_2$ mark is echoed by the receiver but, to retain the benefits of the high resolution congestion estimate, the *amount* by which a BMCC sender backs off would be given by the high resolution load factor estimate [42]. This allows fairness between BMCC and TCP on small BDP paths, and on high BDP paths, allows BMCC flows to achieve higher throughput without causing starvation of TCP.

Figure 13 illustrates this behavior for the case of a BMCC-enabled bottleneck of capacity 45 Mbps that a single BMCC shares with either a TCP SACK, CUBIC or CTCP flow, each with RTT, $T = 100$ ms. Observe that BMCC maintains a larger average congestion window than all other protocols, as needed to give an incentive for its adoption, without starving them. On higher BDP paths, BMCC flows an achieve higher throughput. We show that the degree of unfairness caused by BMCC when it co-exists with TCP is comparable to the degree of unfairness introduced by TCP's own RTT-unfairness. For instance, TCP SACK flows achieve bandwidth allocations that are proportional to $1/RTT^z$, where $1 \leq z \leq 2$ [13].

To compare the degree of unfairness when BMCC and TCP flows co-exist with the case when TCP flows with different RTTs compete at a bottleneck, we measure the gain, G, in the throughput of a more aggressive flow in these scenarios. To achieve this, we consider the following cases: In the first case, called $BASELINE$, we run two TCP flows each with RTT T. In the second case, called MIX, we run two experiments: (a) one TCP flow with RTT T competes with another TCP flow with a 10 ms RTT and (b) one BMCC flow competes with one TCP flow, each with RTT T. The gain is then

$$G = \frac{\mathcal{T}(B)_{MIX}}{\mathcal{T}(B)_{BASELINE}} \tag{23}$$

where $\mathcal{T}(B)_i$ is the throughput achieved in case $i \in \{MIX, BASELINE\}$. Figure 14 plots the throughput gained by a more aggressive flow as a function of T. Note that a TCP flow with a 10 ms RTT and a BMCC flow are the more aggressive flows in (a) and (b), respectively. Observe that the unfairness in case (a) is always larger than in case (b) with all versions of TCP. This indicates that, in the cases considered, BMCC flows are no more unfair to TCP flows than are short RTT TCP flows to the longer ones. Note that since CUBIC and CTCP employ more aggressive control laws, the throughput gain by BMCC flows is less than the throughput gain over SACK flows.

VII. Conclusion

This paper has demonstrated that simple MI-AI-MD can achieve a substantial improvement over the current state of the art in congestion control, if the MI and MD factors adapt to moderately accurate estimates of the current load on the network. Moreover, such accuracy can be achieved for a wide range of network conditions by aggregating feedback from a single bit per packet. This is in contrast to earlier proposals such as REM, in which the dynamic range of congestion

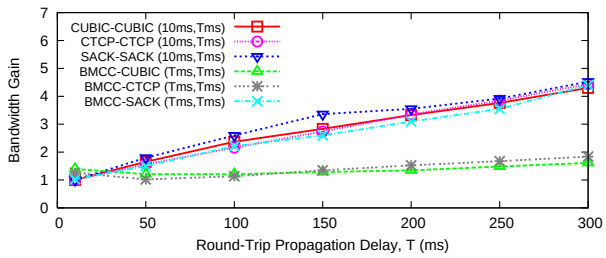


Fig. 14. Comparison of the bandwidth gained by a more aggressive flow when (a) two TCP flows with RTT, T and 10 ms compete; (b) a BMCC flow competes with a TCP flow, each with RTT, T. The 10 ms RTT flow and BMCC are the more aggressive flows in the former and latter cases, respectively.

signals limits the flows to a narrow range of bit rates. In particular, the algorithm BMCC has been presented, which outperforms XCP and, in some cases, RCP in terms of average flow completion times for typical Internet flow sizes.

Incremental deployment is a standard challenge for new congestion control protocols. The marking required by BMCC does not interfere with existing traffic, but starvation of either BMCC or competing SACK flows is possible. A variant of BMCC has been proposed which achieves a better level of fairness than the original presented in [43].

Many extensions of this work are possible. First, the distinction between MI and AI modes may be unnecessary given an accurate estimate of the load factor; better performance may be possible using an increase rule, such as increasing proportional to $w^{1-\hat{f}}$, that makes a smoother transition from an aggressive increase proportional to the current estimated capacity to ensure efficiency, towards a gentle increase independent of the current rate to ensure fairness.

Second, a more detailed analysis of the effect of the estimation error due to ADPM would provide guidance in setting its parameters more effectively.

Finally, in common with many schemes in which routers signal incipient congestion, BMCC assumes routers know the capacity of each of its links. This need not be the case when links have a varying capacity, such as wireless links, links with power saving modes, or shared media such as cable modem links. In such cases, the load factor could be estimated based on the fraction of time the link is busy, without explicit knowledge of the capacity.

VIII. Acknowledgements

We thank Fahad Rafique Dogar, Zafar Ayyub Qazi, Craig Partridge, Daniel Mosse and the anonymous reviewers for their insightful comments and suggestions.

References

[1] D. Katabi, M. Handley, and C. Rohrs, "Internet congestion control for high bandwidth-delay product networks," in *ACM SIGCOMM*, 2002.
[2] I. Rhee and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," in *PFLDNet*, 2005.
[3] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, architecture, algorithms and performance," in *IEEE INFOCOM*, 2004.
[4] V. Konda and J. Kaur, "RAPID: Shrinking the congestion-control timescale." in *IEEE INFOCOM*, 2009.
[5] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," in *IEEE INFOCOM*, 2006.
[6] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown, "Procesor sharing flows in the internet," in *IWQoS*, 2005.
[7] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, and B. Vandalore, "The ERICA switch algorithm for ABR traffic management in ATM networks," *IEEE/ACM Trans. Netw.*, vol. 8, no. 1, pp. 87–98, Feb 2000.
[8] A. Falk, D. Katabi, and Y. Pryadkin, "Specification for the explicit control protocol (XCP)," in *draft-falk-xcp-03.txt*, 2007.
[9] C.-H. Tai, J. Zhu, and N. Dukkipati, "Making large scale deployment of RCP practical for real networks," in *IEEE INFOCOM*, 2008.
[10] S. H. Gunderson, "IPv6 deployment," in *RIPE 57*, 2008.
[11] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug 1993.
[12] K. K. Ramakrishnan and S. Floyd, "The addition of explicit congestion notification (ECN) to IP," in *IETF RFC 3168*, Sep 2001.
[13] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One more bit is enough," in *ACM SIGCOMM*, 2005.
[14] I. A. Qazi and T. Znati, "On the design of load factor based congestion control protocols for next-generation networks," *Comput. Netw.*, vol. 55, pp. 45–60, January 2011.
[15] F. Kelly, "Charging and rate control for elastic traffic," *European Trans. on Telecommunications*, vol. 8, pp. 33–37, 1997.
[16] S. Athuraliya, V. Li, S. Low, and Q. Yin, "REM: Active queue management," *IEEE Network*, vol. 15, no. 3, pp. 48–53, May 2001.
[17] M. Sharma, D. Katabi, R. Pan, and P. Prabhakar, "A general multiplexed ECN channel and its use for wireless loss notification," in *IEEE Globecom*, 2004.
[18] M. Adler, J. yi Cai, J. K. Shapiro, and D. Towsley, "Estimation of congestion price using probabilistic packet marking," in *IEEE INFOCOM*, 2003.
[19] R. W. Thommes and M. J. Coates, "Deterministic packet marking for time-varying congestion price estimation," *IEEE/ACM Trans. Netw.*, vol. 14, no. 3, pp. 592–602, June 2006.
[20] L. L. H. Andrew, S. V. Hanly, S. Chan, and T. Cui, "Adaptive deterministic packet marking," *IEEE Comm. Letters*, vol. 10, no. 11, pp. 790–792, Nov. 2006.
[21] L. L. H. Andrew and S. V. Hanly, "The estimation error of adaptive deterministic packet marking," in *Proc. Allerton Conf. Commun. Contr. Comput.*, 2006.
[22] H.-K. Ryu and S. Chong, "Deterministic packet marking for max-min flow control," *IEEE Comm. Letters*, vol. 9, no. 9, pp. 856–858, Sep 2005.
[23] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgement options," IETF, Tech. Rep. RFC2018, 1996.
[24] N. Vasic, S. Kuntimaddi, and D. Kostic, "One bit is enough: a framework for deploying explicit feedback congestion control protocols," in *COMSNETS*, 2009.
[25] X. Li and H. Yousefi'zadeh, "MPCP: multi packet congestion-control protocol," *SIGCOMM CCR*, vol. 39, no. 5, pp. 5–11, 2009.
[26] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management," in *ACM SIGCOMM*, Aug 2001.
[27] X. D. Wei, "Microscopic behavior of internet congestion control," Dissertation (Ph.D.), California Institute of Technology, 2007.
[28] V. Paxson, "End-to-end internet packet dynamics," in *ACM SIGCOMM*, 1997.
[29] J. A. Jasleen, J. Aikat, J. Kaur, F. Donelson, and S. K. Jeffay, "Variability in tcp round-trip times," in *IMC*, 2003.
[30] R. Shorten, F. Wirth, and D. Leith, "A positive systems model of TCP-like congestion control: Asymptotic results," *IEEE/ACM Trans. Netw.*, vol. 14, pp. 616–629, 2006.
[31] W.-T. Tan and A. Zakhor, "Real-time internet video using error resilient scalable compression and TCP-friendly transport protocol," in *IEEE Trans. on Multimedia*, Jun 1999.
[32] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck, and X. Zhang, "Delving into internet streaming media delivery: a quality and resource utilization perspective," in *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2006, pp. 217–230.
[33] K. K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks," *ACM Trans. Comput. Syst.*, vol. 8, pp. 158–181, May 1990.
[34] M. Corless and R. Shorten, "Deterministic and stochastic convergence properties of aimd algorithms with nonlinear back-off functions," *Automatica*, 2011.

[35] F. Baccelli, S. Machiraju, D. Veitch, and J. Bolot, "The role of pasta in network measurement," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1340–1353, Aug. 2009.
[36] S. Ha and I. Rhee, "Hybrid slow start for high-bandwidth and long-distance networks," in *PFLDnet*, 2008.
[37] S. H. Low, L. L. H. Andrew, and B. P. Wydrowski, "Understanding XCP: Equilibrium and fairness," in *IEEE INFOCOM*, 2005.
[38] X. Li and H. Yousefi'zadeh, "DCP-EW: Distributed congestion-control protocol for encrypted wireless networks," in *IEEE WCNC*, 2010.
[39] D. Papadimitriou, M. Welzl, M. Scharf, and B. Briscoe, "Open research issues in internet congestion control," in *RFC 6077*, Feb 2011.
[40] M. Suchara, L. L. H. Andrew, R. Witt, K. Jacobsson, B. P. Wydrowski, and S. H. Low, "Implementation of provably stable maxnet," in *Proc. Broadnets*, 2008.
[41] R. Fonseca, G. M. Porter, R. H. Katz, S. Shenker, and I. Stoica, "IP options are not an option," UC Berkeley, Tech. Rep., December 2005.
[42] I. A. Qazi, L. L. H. Andrew, and T. Znati, "Incremental deployment of new ECN-compatible congestion control," in *PFLDNeT*, 2009.
[43] ——, "Congestion control using efficient explicit feedback," in *IEEE INFOCOM*, 2009.

APPENDIX

Lemma 1: For any $0 \le a \le b \le 1$, any integer N, real number d and collection of real numbers y_n,

$$a^d \ge \frac{1}{N}\sum_{n=1}^{N} a^{y_n} \qquad \text{implies} \qquad b^d \ge \frac{1}{N}\sum_{n=1}^{N} b^{y_n}.$$

Proof: Since a^x is convex in x, for all $k \ge 1$ and all x, we have $a^x/k + a^0(k-1)/k \ge a^{x/k}$, whence

$$\frac{1}{k}\sum_{n=1}^{N}(a^{y_n(i)-d} - 1) \ge \sum_{n=1}^{N}(a^{(y_n(i)-d)/k} - 1).$$

Substituting $k = \log(a)/\log(b) \ge 1$ gives that

$$\sum_{n=1}^{N}(a^{y_n(i)-d} - 1) \le 0 \qquad \text{implies} \qquad \sum_{n=1}^{N}(b^{y_n(i)-d} - 1) \le 0$$

from which the result follows. ■

Ihsan Ayyub Qazi received the BSc. (Hons) degree in Computer Science and Mathematics from the Lahore University of Management Sciences (LUMS), Pakistan, in 2005 and the Ph.D. degree in Computer Science from the University of Pittsburgh, PA, USA in August 2010.

He is an Assistant Professor in the Department of Computer Science at the LUMS School of Science and Engineering, Lahore, Pakistan. From 2010 to 2011, he was a Postdoctoral Research Fellow at the Centre for Advanced Internet Architectures, Swinburne University of Technology, Australia. He is the recipient of the Andrew Mellon Fellowship and the Best Graduate Student Research Award from the University of Pittsburgh in 2009. His current research interests include future Internet design, green networking, wireless networks, and performance modeling of networked systems. He is a member of the ACM.

Lachlan L. H. Andrew (M'97-SM'05) received the B.Sc., B.E. and Ph.D. degrees in 1992, 1993, and 1997, from the University of Melbourne, Australia. Since 2008, he has been an associate professor at Swinburne University of Technology, Australia, and since 2010 he has been an ARC Future Fellow. From 2005 to 2008, he was a senior research engineer in the Department of Computer Science at Caltech. Prior to that, he was a senior research fellow at the University of Melbourne and a lecturer at RMIT, Australia. His research interests include energy-efficient networking and performance analysis of resource allocation algorithms. He was co-recipient of the best paper award at IEEE INFOCOM 2011 and IEEE MASS 2007. He is a member of the ACM.

Taieb Znati received the M.S degree from Purdue University, West Lafayette, Indiana and the Ph.D. degree in Computer Science from Michigan State University, East Lansing in 1988.

He 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 has served as general chair for several 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, Journal of Adhoc 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, and performance analysis of network protocols.