This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

Congestion Control using Efficient Explicit Feedback

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

Lachlan L. H. Andrew
Centre for Advanced Internet Architectures
Swinburne University of Technology
Hawthorn, Vic 3122, Australia
Email: landrew@swin.edu.au

***Abstract*—This paper proposes a framework for congestion control, called Binary Marking Congestion Control (BMCC) for high bandwidth-delay product networks. The basic components of BMCC are i) a packet marking scheme for obtaining high resolution congestion estimates using the existing bits available in the IP header for Explicit Congestion Notification (ECN) and ii) a set of load-dependent control laws that use these congestion estimates to achieve efficient and fair bandwidth allocations on high bandwidth-delay product networks, while maintaining a low persistent queue length and negligible packet loss rate. We present analytical models that predict and provide insights into the convergence properties of the protocol. Using extensive packet-level simulations, we assess the efficacy of BMCC and perform comparisons with several proposed schemes. BMCC outperforms VCP, MLCP, XCP, SACK+RED/ECN and in some cases RCP, in terms of average flow completion times for typical Internet flow sizes.**

I. Introduction

Achieving efficient and fair bandwidth allocation while minimizing packet loss, bottleneck queue and Average Flow Completion Time (AFCT) in high Bandwidth-Delay Product (BDP) networks has long been a daunting challenge. The congestion control algorithm in TCP has significant limitations in achieving this goal. With the increase in the deployment of very high speed links in the Internet, the need for a viable replacement has become increasing important.

To meet this challenge, several congestion control protocols have been proposed. Pure end-to-end schemes ($e.g.$, HighSpeed TCP [1], FAST [2]) typically rely on packet loss and/or delay as signals of congestion. This reliance causes such schemes to introduce artificial packet losses and/or queuing at the bottleneck, which should be avoided in the first place. Research studies have shown that using such signals poses fundamental limitations in achieving high utilization and fairness while keeping low bottleneck queue and negligible packet loss rate in high BDP paths [1], [3]. Network-based schemes ($e.g.$, XCP [4], RCP [5]) address this challenge by enforcing fairness and congestion control inside the routers. Such schemes incur higher per-packet overhead and are hard to deploy in today's Internet as they require more bits for feedback than are available in the IP header such as XCP (128 bits), RCP (96 bits). Changing the IP header requires a non-trivial and a time-consuming standardization process.

To approximate the performance of network-based schemes but with a lower deployment barrier and reduced per-packet overhead, limited feedback-based schemes such as TCP+RED/ECN [6], VCP [7], MLCP [8] have been proposed. These schemes use few bit(s) of explicit feedback from the network to aid end-hosts in making congestion control decisions. Such protocols typically require modifications at the end-hosts with incremental support from the routers.

TCP+RED/ECN uses one bit of explicit feedback for signalling whether there is congestion or not. Since the feedback is highly imprecise, sources must be conservative in their increase policy and aggressive in their decrease policy. In the case of 2-bit schemes (such as VCP), Qazi et al. [8] recently showed that they converge unnecessarily slowly, and that quantizing to at least 16 levels is required to achieve near-optimal convergence speed. In this paper, we propose the Binary Marking Congestion Control (BMCC) protocol, which uses the existing ECN bits (in a way compatible with existing RED/ECN) to achieve near-optimal performance in terms convergence to efficiency and fairness, surpassing the convergence speed of the MLCP scheme proposed in [8]. BMCC maintains low bottleneck queue and negligible packet loss rate. It outperforms VCP, MLCP, XCP, SACK+RED/ECN and in some cases RCP in terms of AFCT for typical Internet flow sizes.

With BMCC, each router periodically computes the load factor (ratio of input traffic and queue length to capacity) on its output links. To achieve efficient and fair bandwidth allocation with high convergence speeds, a high resolution estimate of the computed load factor is needed. BMCC uses a packet marking scheme called Adaptive Deterministic Packet Marking (ADPM) [9] to obtain congestion estimates with up to 16-bit resolution using the existing two ECN bits. ADPM conveys signals with a lower Mean Square Error (MSE) than competitors such as REM [10] and RAM [11]. It monitors side information, such as the IPid field, and uses that to interpret the ECN bit of each packet differently, as proposed by Thommes and Coates [12]. Each arriving packet at the receiver carries a bound on the value of the load factor at the bottleneck. The receiver's estimate of the load factor is updated whenever a new packet provides a tighter bound. The load factor estimate is echoed back to the sources via acknowledgement packets using TCP options. Based on this value, sources apply load-dependent Multiplicative Increase (MI), Additive Increase (AI) and Multiplicative Decrease (MD). Unlike in

978-1-4244-3513-5/09/$25.00 ©2009 IEEE

VCP, the parameters of these control laws depend on the high resolution estimate of the load factor. When the load factor is small, sources increase their rates rapidly to fill the bottleneck capacity. Otherwise, sources apply AIMD to achieve fairness. The rates of adjustment are chosen to ensure RTT fairness.

To illustrate the convergence and fairness properties of BMCC, consider a simple network scenario with one bottleneck link of capacity 100 Mbps. Figure 1 shows the throughput of two BMCC flows that arrive with an inter-arrival time of 50 s and have round-trip propagation delays of 150 ms and 200 ms, respectively. Observe that the two flows rapidly achieve rates that are within 50% of their fair share before converging to a fair (≈50 Mbps each) and efficient (≈100 Mbps) bandwidth allocation. Figure 2 shows the corresponding load factor at the bottleneck and the flows' estimates. ADPM allows flows to quickly track changes in load factor at the bottleneck.

BMCC's use of load factor as a signal of congestion enables it to decouple loss recovery from congestion control. This facilitates distinguishing error losses from congestion-related losses, which is important in wireless environments. Furthermore, the scale-free nature of load factor allows it to be encoded using few bits.

Using extensive ns2 simulations, we show that BMCC achieves efficient and fair bandwidth allocation while minimizing packet loss, bottleneck queue and AFCT. We present analytical models which provide insights into the performance of BMCC. These insights are likely to lead to better designs for next-generation congestion control protocols. We also evaluate mechanisms for reducing the overhead of TCP options in conveying the load factor estimates from the receiver back to the sender. The resulting algorithm, employed by BMCC, introduces little overhead and is robust to lost ACKs.

The rest of the paper is organized as follows. Section II describes the control laws and algorithms used by BMCC. Some design issues related to BMCC are addressed in Section III. This is followed in Section IV by a model for understanding the convergence properties of BMCC. Section V analyzes the impact of ADPM on convergence and Section VI studies ways to reduce the overhead of using TCP options. Section VII, compares the performance of BMCC with that of other schemes by simulation. Related work and concluding remarks are discussed in Sections VIII and IX.

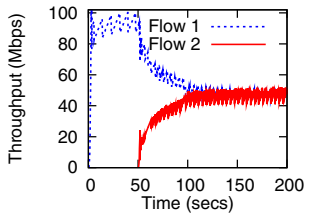


Fig. 1. Throughput of two flows on a 100 Mbps link

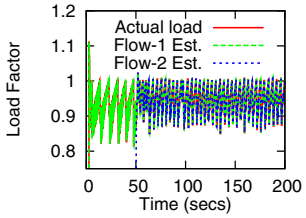


Fig. 2. Comparison of the load factor at the bottleneck and the flows' estimates of it

II. Algorithm and Protocol

BMCC uses ADPM to estimate the load factor f on the most congested link, and then uses the estimate, $\hat{f}$, to adjust the send window, w. Let us first consider how f is computed at the routers, then how the estimate $\hat{f}$ is determined, and finally how $\hat{f}$ is used by the senders. The values of the unspecified parameters will be discussed at the end of this section.

A. BMCC Router

During every time interval t_p, each router computes the load factor on each of its output links as

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

where λ is the amount of traffic received during t_p, C_l is the capacity of the link and $\gamma_l \leq 1$ is the target utilization. Also, q_{av} is the exponentially weighted moving average queue length, using a time-constant of $8t_p$, and $\kappa_1 = 0.75$ controls how fast to drain the queue [7], [10], [13], [14].

B. BMCC Receiver and ADPM

The router conveys its load factor to the sender by applying ADPM [9] to the ECN bits. Recall that the ECN bits on an unmarked packet are initially $(10)_2$, and routers set these bits to $(11)_2$ to indicate congestion. If $f \geq u$ or the packet already contains a mark $(11)_2$, then BMCC marks the packet with $(11)_2$, where u is the maximum value of f that ADPM can signal. Otherwise, ADPM computes a deterministic hash h of the packet contents, such as the 16-bit IPid field. 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.

The receiver maintains the current estimate, $\hat{f}$ of the load factor at the bottleneck on the forward 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 VI. Note that the receiver's estimate will lag behind the true value [15], 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 $(1, u)$.

C. BMCC Sender

As argued in [7], the ideal window update strategy given a load factor of f is to increase the total load by a factor of $1 - f$, in a way which balances rates. However, as neither the bottleneck capacity nor number of competing flows are known, BMCC uses different control laws, based on whether the most loaded link is lightly-, heavily- or over-loaded, corresponding to $f \in [0, \eta)$, $[\eta, 1)$ or $[1, \infty)$ for some η. As it will be shown, this also allows both fairness and rapid acquisition of idle capacity.

1) Low Load ($0 \le \hat{f} < \eta$)*:* To achieve high utilization rapidly, sources apply MI with factors proportional to $1 - \hat{f}$. In particular,

$$w(t+T) = w(t)(1 + \xi(\hat{f})), \tag{2}$$

where T is the RTT of the flow, $\xi(\hat{f}) = \kappa_2(1-\hat{f})/\hat{f}$ and κ_2 is a step size. Since BMCC implements the algorithm VCP sought to, the analysis of [7] shows that stability is achieved for $\kappa_2 < 1/2$; BMCC uses $\kappa_2 = 0.35$.

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

$$w(t+T) = w(t)(1 + \xi(\hat{f}))^{T/t_p} \tag{3}$$

is used so that windows grow at a rate independent of T.

2) High Load ($\eta \le \hat{f} < 1$)*:* When the system has achieved high utilization, the algorithm must seek fairness. This is achieved using AIMD. In high 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 [7].

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

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

where

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

varies linearly in $[\beta_{\min}, \beta_{\max}] \subset (0,1)$, u is the maximum value of f that ADPM can signal, and $\Delta\beta = \beta_{\max} - \beta_{\min}$.

D. Parameter values

1) Measurement interval, t_p: The period t_p should be greater than the RTT of most flows to smooth out sub-RTT burstiness, but should also be small enough to ensure responsiveness to congestion. Internet measurements [16] report that vast majority of flows have RTTs less than 200 ms. Hence, BMCC uses $t_p = 200$ ms.

2) Mode threshold, η: A high value of η avoids underutilization, but limits AIMD's scope to induce fairness, and risks overshooting the link capacity and causing excess packet loss. To balance these, BMCC uses $\eta = 0.75$, and $\eta_0 = 0.15$. As an exception, new flows remain in MI until $\hat{f}$ first reaches 1.

3) Backoff parameter, β: The MD 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 high value of $\beta_{\max}$ (such as 0.99) leads to slow convergence and reduces responsiveness to congestion. In contrast, a low value of $\beta_{\max}$ (such as 0.5) reduces the average throughput ($\sim C(1+\beta)/2$) and introduces large variations in the throughput of flows, which is highly undesirable for real-time and multimedia applications [17]. To balance these, BMCC uses $\beta_{\max} = 0.875$ (see Section IV). However, to ensure high responsiveness and fast convergence under high load, $\beta(f)$ is decreased linearly with f until $\beta(f) = \beta_{\min}$. The choice of $\beta_{\min}$ determines the range of values that $\beta(f)$ can assume. This range should be large enough to reap the benefits of small β values but small enough to prevent flows from entering MI after overload detection, which can lead to high packet loss rate. To select the value to ensure this, note that $\min(f\beta(f)) \ge \eta$. Since for $f \ge 1.2$, the lowest β is applied, $\beta(f)$ should be at least $\eta/1.2 = 0.625$. BMCC uses $\beta_{\min} = 0.65$.

III. Design Issues

The foregoing design raises some questions, which are now addressed.

a) What is the congestion level assumed by new flows?: ADPM needs an initial estimate of the congestion level. New flows initially estimate $\hat{f} = \eta_0$, and thus increase their windows by a factor of $\xi(\eta_0) \approx 3$ per t_p. This helps short flows to finish quickly, and does not induce excessive burstiness, since it is only slightly faster than existing slow start.

b) Can new flows cause overload before ADPM has been able to signal congestion?: When $f \in [1, u)$, sources enter MD *probabilistically* using ADPM. 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.

c) Sources may apply different β values at the same time; does this lead to unfairness?: Sources with ADPM use β that varies in $\left[\beta(f), \beta(\hat{f})\right]$ depending on the estimated load factor. This may lead to short-term unfairness (on the scale of a few RTTs) but causes no long-term harm. 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 with unsynchronized losses, let $X_i(t,s)$ be the average rate of each flow i over the interval $(t, t+s)$, and let the "fairness rate" be

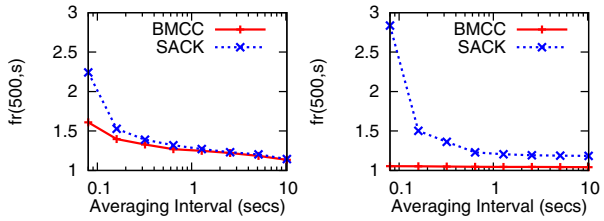
$$fr(\tau, s) = \frac{1}{\tau}\int_0^\tau \frac{\max_i(X_i(t,s))}{\min_j(X_j(t,s))}\,dt,$$

where $i, j \in \{1, 2\}$ and τ is the total observation period. Figure 3 shows $fr(500, s)$ against the averaging interval s for two link capacities and $T = 80$ ms. Observe that the fr curve for BMCC remains below that of SACK implying that it always fairer than SACK on short time scales. The value is higher on the 1 Mbps link; this is because the average β value is higher in this case and hence the variation also is.

d) Why use a higher MI 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 causes slow convergence. In order to improve the AFCT of short flows in this scenario, a larger initial η is used. This allows new BMCC flows to apply the more aggressive, load-factor guided MI until the end of the first congestion epoch.

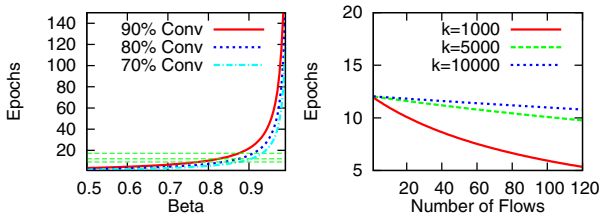
IV. Analysis

The properties of BMCC will now be studied by considering the rate achieved by a newly starting flow in two important cases: a link already carrying N long-lived flows, and an idle link.



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

Fig. 3. Fairness rate as a function of the averaging interval ($T = 80$ ms)



(a) (b)

Fig. 4. Number of epochs needed for (a) 70%, 80% and 80% convergence as a function of β. The green lines show the epochs for $\beta_{\max} = 0.875$ (b) 80% convergence as a function of the number of flows for different BDPs, $k \in \{1000, 5000, 1000\}$ pkts (where $\beta_{\max} = 0.875$ and $\beta_{\min} = 0.65$)

A. Loaded link

This section considers the rate of convergence to fairness when a new flow starts competing with N long-lived BMCC flows. This scenario reflects the fact that most data in the Internet comes from "elephant" flows, and so a highly multiplexed bottleneck is likely to have long-lived flows.

Consider a bottleneck link of capacity C_l shared by N flows with equal RTTs $T = t_p$. Define a round as a single AIMD cycle, the duration of which is d_e RTTs. Note that BMCC causes these rounds to be synchronized between flows, because congestion is signaled to all flows, rather than random packet drops for a small number of flows. Let $\Delta w_{ij}(e) = w_i(e) - w_j(e)$ for flows i and j in the eth round. Since $\beta_{\min}$ is chosen so that flows do not re-enter MI mode after an MD, this difference is affected only by MD events. Thus

$$\begin{aligned}\mathbb{E}[\log(\Delta w_{ij}(e))] &= \mathbb{E}[\log(\beta(f))] + \mathbb{E}[\log(\Delta w_{ij}(e-1))] \\ &= e\mathbb{E}[\log(\beta(f))] + \log(\Delta w_{ij}(0)). \end{aligned} \quad (7)$$

For the new flow i to reach parity with a flow j with $w_j(0) = B \equiv C_l T/N$, in the sense of $\Delta w_{ij}(e)/B \leq \delta$, it would take

$$m = \frac{\log(1/\delta)}{\mathbb{E}[\log(1/\beta(f))]} \quad (8)$$

AIMD rounds, which is constant with respect to BDP [18].

The duration of an AIMD round can be calculated by noting that, when the link utilization exceeds 100%, the aggregate congestion window size, $w_a = \sum_{i=1}^{N} w_i(t)$ is at least $k = C_l T$. Therefore, it would take at least $d_e = k(1-\mathbb{E}[\beta(f)])/N\alpha$ RTTs for w_a to become greater than k after applying MD.

Note that m is negatively correlated with the durations of the m epochs, and so the actual mean convergence time is slightly less than $\mathbb{E}[m]\mathbb{E}[d_e]$. Thus, the total number of RTTs needed to converge to a fair bandwidth allocation is bounded above, and approximated, by

$$r = B\frac{\log(1/\delta)(1-\mathbb{E}[\beta(f)])}{\alpha\mathbb{E}[\log(1/\beta(f))]}. \quad (9)$$

Thus, the convergence time until absolute fairness varies with BDP as $O(B)$.

In order to derive the expected value of $\beta(f)$, we will use the following lemma, established in Section V.

Lemma 1: In steady-state, flows with homogenous round-trip times detect overload with high probability ($\geq$0.8) in one t_p with ADPM.

To determine the expected value of $\beta(f)$ in overload, we first derive the maximum value of f when it exceeds 100%.

In steady-state, each BMCC flow achieves the same rate independent of its RTT. Since f is measured over t_p, this implies that in overload $\lambda \geq t_p N(C_l/N)$. Further, with AI, the aggregate rate increases by N pkts per t_p. Thus, when overload is first detected, up to t_p after onset, λ is uniformly distributed on $(C_l t_p, C_l t_p + N)$, and the average queue length, q_{av}, will be uniformly distributed in $(0, N)$. From (1), the first congested load factor is thus distributed as

$$\begin{aligned} f &\sim \frac{C_l t_p + N(1+\kappa_1)U(0,1)}{\gamma_l C_l t_p} \\ &\sim 1 + \frac{2N}{C_l t_p}U(0,1), \end{aligned} \quad (10)$$

where $U(0,1)$ is a uniformly distributed random variable on $(0,1)$, $\gamma_l = 1$ and $\kappa_1 \leq 1$. Since $\beta(f)$ is linear by (6), $\mathbb{E}[\beta(f)] = \beta(\mathbb{E}[f])$, which by (10) gives

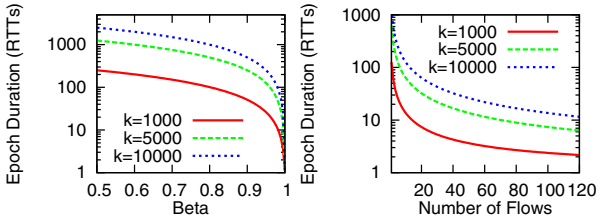
$$\mathbb{E}[\beta(f)] = \begin{cases} \beta_{\max} - \frac{N\Delta\beta}{k(u-1)} & f \leq u \\ \beta_{\min} & f > u \end{cases}$$

This establishes the mean values

- Duration of an epoch: $\mathbb{E}[d_e] = \frac{k}{N\alpha}(1-\beta_{\max}) + \frac{\Delta\beta}{\alpha(u-1)}$
- Number of epochs: $\mathbb{E}[m] = \dfrac{\log(1/\delta)}{\mathbb{E}[\log(1/\beta(f))]}$

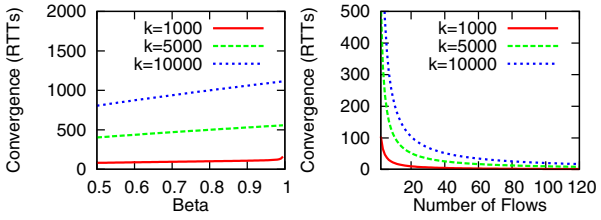
Figures 4 and 5 show $\mathbb{E}[m]$ and $\mathbb{E}[d_e]$ respectively, as β is varied in $[0.5, 1)$ (with $\beta = \beta_{\min} = \beta_{\max}$), and as a function of $N \in [2, 120]$ for different BDPs, $k \in \{1000, 5000, 10000\}$ pkts. Observe that while $\mathbb{E}[d_e]$ decreases linearly with β, $\mathbb{E}[m]$ increases exponentially with it, leading to slower convergence as shown in Figure 6. Further, if $\beta(f)$ is made a linear function of f then $\mathbb{E}[m]$, $\mathbb{E}[d_e]$ and r decrease with N. The reason is that as N increases, the average load factor value at the bottleneck increases causing the sources to apply a smaller $\beta(f)$ value. This results in improved convergence. Also note that convergence becomes slower as the BDP of the path increases.

The above analysis is related to the model presented in [18]. However, they consider the back-off parameter, $\beta(f)$, to be a binary random variable, whereas in our case it is continuous. The analysis can be extended to flows with heterogeneous RTTs but at the price of a more involved development (see [18]). The qualitative insights relating to the influence of the AIMD parameters, however, remain unchanged.



(a) (b)

Fig. 5. Duration of an epoch as a function of (a) β, with $N = 2$ (b) number of flows for different BDPs, k (where $\beta_{\max} = 0.875$ and $\beta_{\min} = 0.65$)



(a) (b)

Fig. 6. Convergence time as a function of (a) β with $N = 2$ (b) number of flows for different BDPs, k (where $\beta_{\max} = 0.875$ and $\beta_{\min} = 0.65$)

B. Flow starting on an idle link

A well known problem of TCP SACK is that a flow sending on an idle path with large BDP takes too long to start up [5], and then causes many packet losses when the window finally reaches the BDP. BMCC addresses this issue by increasing its rate faster until incipient congestion is explicitly signaled, and then slowing down the rate of increase, in three phases. These will now be examined for the fluid limit of large BDPs with packet pacing and without delayed acknowledgements, and in the simple case that $T = t_p$. Other cases are similar, except that small BDPs require the behavior of ADPM to be modelled.

The first phase, with $f = w/(C_l T) < \eta_0$, increases the window by a factor of $\kappa_2(1-\eta_0)/\eta_0 \approx 3$ each RTT, giving

$$w(t) = 3^{t/T} \qquad t \le t_1 \equiv T\log_3(\eta_0 C_l T) \qquad (11)$$

If the BDP is sufficiently large, $\eta_0 C_l T$ is large enough for ADPM to estimate the load factor accurately, and to enter the second phase. If the BDP is small, this aggressive phase will last longer giving BMCC a faster start-up. The second phase, with $f = w/(C_l T) \in (\eta_0, \eta)$, increases the window by $w \cdot \kappa_2(1-f)/f = w \cdot \kappa_2(C_l T - w)/w$ each RTT, giving

$$\frac{dw}{dt} = \frac{\kappa_2}{T}(C_l T - w).$$

Solving this, and applying the initial condition

$$w(t) = C_l T\left(1 - (1-\eta_0)e^{-\kappa_2(t-t_1)/T}\right) \quad t_1 < t \le t_2 \quad (12)$$

where t_2 is defined by

$$t_2 - t_1 = \frac{T}{\kappa_2}\log\left(\frac{1-\eta_0}{1-\eta}\right).$$

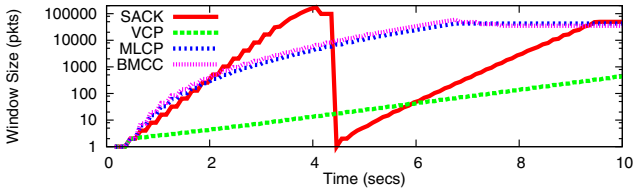


Fig. 7. Comparison of the growth rate of the congestion window sizes for SACK, VCP, MLCP and BMCC on a 2 Gbps link with $T = 200$ ms

Note that, unlike regular multiplicative increase, this results in concave *negative* exponential growth. The derivative is continuous at the point of inflection, t_1.

In the third phase, w grows at $\alpha = 1$ packet per RTT, giving $w(t) = \eta C_l T + \alpha(t-t_2)/T$. This decreases the rate of increase of the window unless $\alpha > \kappa_2 C_l T(1-\eta)$, corresponding to a window of $\alpha/(\kappa_2(1-\eta)) \approx 12$ packets.

The total amount of data transmitted until time t is $d = \int_0^t w(\tau)/T\, d\tau$, given by

$$d(t) = \begin{cases} 3^{t/T}/\log(3) & t \le t_1 \\ d(t_1) + C_l(t-t_1) + H(t)T/\kappa_2 & t_1 < t \le t_2 \\ d(t_2) + \eta C_l(t-t_2) + A(t) & t > t_2 \end{cases}$$

where $H(t) = (e^{-\kappa_2(t-t_1)/T} - 1)(1-\eta_0)$ and $A(t) = \alpha t(t - 2t_2)/(2T)$. In contrast, SACK has a window of $w_R(t) = 2^{t/T}$ for all $t < T\log_2(C_l T)$, and sends data $d_R(t) = 2^{t/T}/\log(2)$. Similarly, VCP sets $G = 1.0625$ and has

$$w_V(t) = \begin{cases} G^{t/T} & t \le t_V \\ G^{t_V/T} + \alpha(t-t_V)/T & t > t_V \end{cases}$$

and

$$d_V(t) = \begin{cases} G^{t/T}/\log(G) & t \le t_V \\ d_V(t_V) + 0.8C_l(t-t_V) + A_V(t) & t > t_V \end{cases}$$

where $t_V = T\log_G(0.8C_l T)$ and $A_V(t) = \alpha t(t - 2t_V)/(2T)$.

Figure 7 compares the growth of the congestion window of a single SACK, VCP, MLCP and BMCC flow starting on an idle link. The bottleneck is a 2 Gbps link with RTT=200 ms yielding a BDP of 50000 pkts. The bottleneck buffer size was set to the BDP of the path as specified by the BDP rule. Observe that in the first 2 s, BMCC attains a larger window size than SACK. As the available bandwidth decreases, BMCC adjusts its growth rate and therefore, has a steadier increase. SACK, on the other hand, continues to grow its window size by a factor of two every round-trip time (a straight line on the exponential plot in Figure 7). When it completely fills the router buffers, it has a window size of ∼100000 pkts. Since SACK does not know that it has saturated the path, it continues to increase its window, which results in a loss of ∼90000 pkts giving rise to timeouts. BMCC does not experience a single packet loss. MLCP achieves near-optimal rate of convergence to efficiency for load factor based schemes [8] but uses 4 bits for feedback. Observe that BMCC very closely approximates the performance of MLCP using only the existing bits. In contrast, VCP is too conservative. In the first 10 s, it is able

to attain a congestion window size of only 475 packets (∼1% of the BDP of the path) while a BMCC flow attains a window size of 50000 pkts in less than 6.5 s.

V. Impact of ADPM

In the previous section, we used Lemma 1 to assume that sources detect overload within t_p of when f exceeds 100% using ADPM. In this section, we justify that assumption.

Because f increases by $F = 2N/C_l t_p$ per t_p, and is sampled once per t_p, we can model the first congested load factor as

$$f \sim U(1, 1+F).$$

Given f, the probability that a packet of flow i detects overload using ADPM is equal to the probability that the packet's hash value is $h \in [1, f]$. Since half of the hash values are uniformly distributed in $[1, u]$, this is $p_f = 0.5(f-1)/(u-1)$. The probability that overload remains undetected after d packets is then $(1-p_f)^d$. Thus, the overall probability of overload being undetected after d packets is

$$P(R > d) = \mathbb{E}_f[(1-p_f)^d],$$

where R is the number of packets from flow i until overload is detected. Let $D = 0.5F/(u-1)$. Then $p_f \sim U(0, D)$, and

$$P(R > d) = \frac{1}{D}\int_0^D (1-x)^d\, dx = \frac{1-(1-D)^{d+1}}{D(d+1)}. \quad (13)$$

Using the above expressions, we now prove Lemma 1.

Proof of Lemma 1: Consider a single bottleneck link with N long-lived BMCC flows in steady-state, each with $T = t_p$. In the first t_p interval after overload, each flow sends $d = (C_l t_p/N + 1) = (k/N + 1)$ packets. Substituting the values of d and D in (13), we get

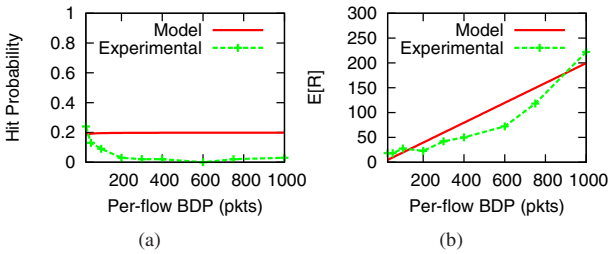
$$P(R > d) = \frac{1-(1-N/(k(u-1)))^{k/N+2}}{(1+2N/k)/(u-1))} \le \frac{1}{1/(u-1)} = 0.2.$$

■

A. Experimental Validation

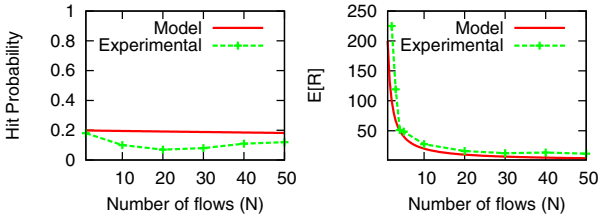
In order to validate the above model, we run extensive ns2 simulations and compare the results against the predicted values. In the first set of experiments, we vary the average per-flow BDP of the path from 25 pkts to 1000 pkts. We maintain ten long-lived BMCC flows with heterogenous round-trip times that vary in [25 ms, 295 ms] (each with a fixed difference of 30 ms). We call the values computed using the above expressions "Model" and those through ns2 simulations "Experimental".

Figures 8 and 9 show values of $P(R > d)$ and $\mathbb{E}[R]$ as a function of the average per-flow BDP of the path. Observe that $P(R > d)$ remains below 0.2 across a range of per-flow BDPs and $\mathbb{E}[R]$ increases almost linearly with the per-flow BDP as predicted by the model. Simulations results yield a smaller value for $P(R > d)$ because F assumes a higher average value than predicted by the model. This may be because flows with $T > t_p$ tend to be bursty, leading to a larger queue buildup and thus a higher value for F. Note that for small per-flow BDPs (<50 pkts), the value of $P(R > d)$ is close to 0.2. The reason is that for such small BDPs, f becomes greater than 1.2 many times in which case ADPM is not used for overload detection. In these experiments we only consider overload detected through ADPM. If overload is allowed to recover through ADPM for f>1.2, $P(R > d)$ would become much lower than 0.2.



(a) (b)

Fig. 8. (a) Probability of overload detection in one t_p with ADPM and (b) Average number of packets needed to detect overload as a function of the average per-flow BDP of the path. $N = 10$ and $T \in [25\text{ ms}, 295\text{ ms}]$.



(a) (b)

Fig. 9. (a) Probability of overload detection in one t_p with ADPM and (b) Average number of packets needed to detect overload as a function of the number of flows ($k = 1000$ pkts). $T \in [25\text{ ms}, (N-1)30\text{ ms}]$.

In the next set of experiments, we vary the number of long-lived flows while keeping the BDP of the path fixed at 1000 pkts. The generated flows have round-trip times that vary in [25 ms, $(N-1)30$+25 ms]. Figures 8 and 9 show that $P(R > d)$ remains below 0.2 for a wide range of per-flow BDPs, following closely the trend predicted by the model.

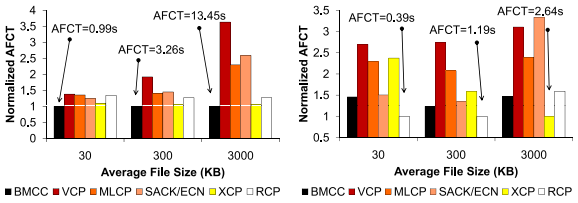
VI. Reverse Signalling Overhead

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 are a significant overhead. Rather than simply 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, and sends $\hat{f}$ every ith ACK. The counter is reset to 1 each time the price changes and incremented by 1 each time $\hat{f}$ is sent. This scheme is robust against losing a small number

TABLE I
OVERHEAD OF SIGNALLING FROM RECEIVER TO SENDER.

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



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

Fig. 14. 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.

of ACKs, but if $\hat{f}$ changes on average once per n packets, the overhead is only $\log(n)/n$ times that of naïvely echoing on every ACK.

We conducted ns2 simulations to evaluate the reduction in overhead for a dumbbell topology with a bottleneck link of capacity 100 Mbps, carrying ten long-lived flows in each direction with T=80 ms. Table I 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.

VII. PERFORMANCE EVALUATION

In this section, we evaluate and compare the performance of BMCC with other protocols using the packet-level simulator ns2 [19], which we have extended with a BMCC module. A wide range of network scenarios are considered, including varying link capacities in the range [100 kbps, 1 Gbps], round-trip times in the range [1 ms, 2 s], number of long-lived, FTP-like flows in the range [1, 1000], and short-lived, web-like flows with offered load and average transfer sizes in the range [0.1C_l Mbps, C_l Mbps] and {30KB, 300KB, 3000KB}, respectively. All simulations use a dumbbell topology with a single bottleneck link. The basic setting is a 155 Mbps link with 80 ms RTT where the forward and reverse path each has 5 FTP flows unless stated otherwise. TCP SACK is always used with RED and ECN enabled at the routers. The bottleneck buffer size is set to the BDP, or two packets per-flow, whichever is larger. The data packet size is 1000 bytes, while the ACK packet size is 40 bytes. All simulations are run for at least 100 s. The statistics neglect the first 5% of the simulation time.

a) Varying Bottleneck Capacity: We vary the bottleneck capacity from 100 Gbps to 2 Gbps while keeping everything else fixed. Figure 10 shows that BMCC is able to maintain high utilization ($\geq$90%) while maintaining low persistent queue length ($<$20% BDP) and negligible packet loss rate across a range of link capacities. While average utilization for SACK reduces to 60$\sim$70% for link capacities $>$10 Mbps, VCP, MLCP, XCP and RCP all achieve $\geq$85% utilization across a range of link capacities. RCP and SACK result in high loss rates for small capacities whereas all other schemes experience negligible loss rates. Average queue length with XCP (5$\sim$40% BDP) is much higher than other schemes for link capacities higher than 1 Mbps.

b) Varying Feedback Delay: We now vary the round-trip time from 1 ms to 2 s while keeping everything else fixed. Figure 11 shows that BMCC, MLCP and VCP are able to maintain high utilization ($\geq$80%) while maintaining low persistent queue length ($\leq$25% BDP) and negligible packet loss rate across a range of RTTs. Observe that with SACK, average utilization reduces to $<$20% for large RTTs. XCP results in a higher average queue length (20$\sim$35% BDP) than other schemes. For low RTTs (e.g., $<$2 ms) the average queue length for BMCC, MLCP and VCP rises to about 5$\sim$25% BDP. This happens because the AI parameter value used in these schemes is large for such small BDP paths. Note that for large RTTs, RCP results in a loss rate of up to 15%; a consequence of its aggressive rate allocation scheme.

c) Varying Number of Long-lived Flows: The number of long-lived flows is now varied from 1 to 1000 while keeping everything else fixed. Figure 12 shows that BMCC, MLCP, VCP, XCP and RCP are able to maintain high utilization ($\geq$90%). While average queue length for BMCC, MLCP and RCP remains less than $<$10% BDP, it is higher for XCP ($\sim$10%$-$40%), VCP (in some cases $\sim$20%$-$40%) and SACK ($\sim$10%). A higher average queue length for VCP in some cases is due to the usage of a higher MD factor than MLCP and BMCC. While loss rate for SACK rises to $\sim$8% for 1000 flows, it becomes more than 20% with RCP.

d) Pareto-Distributed Traffic: To study the performance of BMCC in the presence of variability and burstiness in flow arrivals, we generate web-like flows whose transfer sizes obey the Pareto distribution (shape=1.4) and arrive according to a Poisson process [5].

i) Varying Average File Size: We vary the average file size from 30 KB to 3000 KB for bottleneck capacities of 10 Mbps and 100 Mbps and measure the AFCT of flows. Figure 14 shows the AFCT (normalized by the smallest AFCT) as a function of the average file size. Observe that on a 10 Mbps link, BMCC outperforms all schemes across a range of file sizes. VCP and SACK stretch the AFCT of flows by a factor of up to $\sim$3.5 and $\sim$2 over BMCC, respectively. On a 100 Mbps link, RCP performs best when the average file size is 30 KB and 300 KB. XCP, however, outperforms other schemes when the average file size is 3 MB. BMCC is the second best performing scheme in all cases. For an average transfer size of 30 KB, VCP, XCP and MLCP stretch the AFCT of flows by factors of up to $\sim$2.7, $\sim$2.4 and $\sim$2.3.

VCP results in the highest AFCT because it uses a conservative MI factor of 1.0625, a consequence of quantizing the load factor information into only three levels. BMCC, in contrast, obtains load factor estimates of up to 16-bit resolution, allowing larger MI factors in low-load. BMCC also allows new flows to use MI for longer than VCP or MLCP,

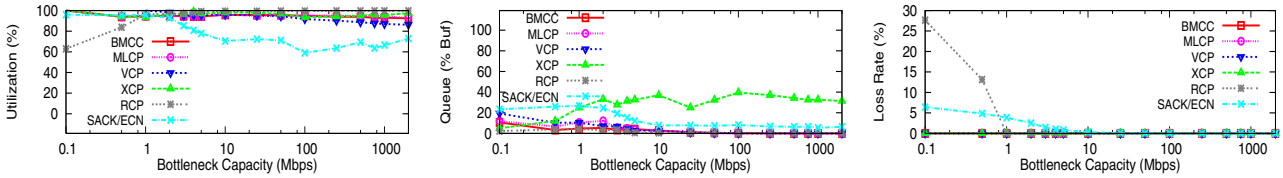


Fig. 10. Impact of varying the bottleneck capacity from 100 kbps to 2 Gbps.

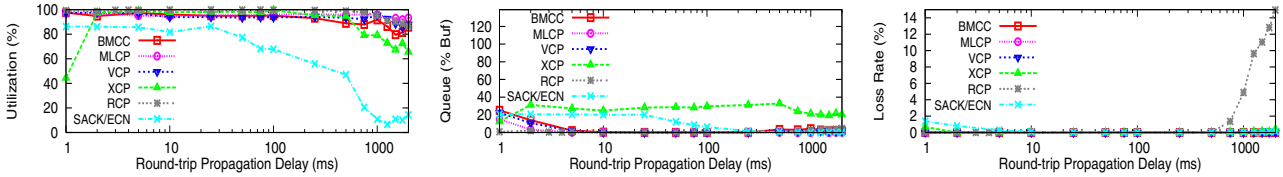


Fig. 11. Impact of varying the round-trip propagation delay from 1 ms to 2 s.

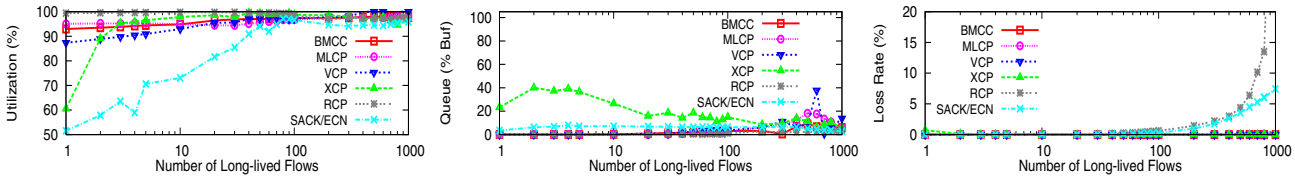


Fig. 12. Impact of varying the number of long-lived, FTP-like flows from 1 to 1000.

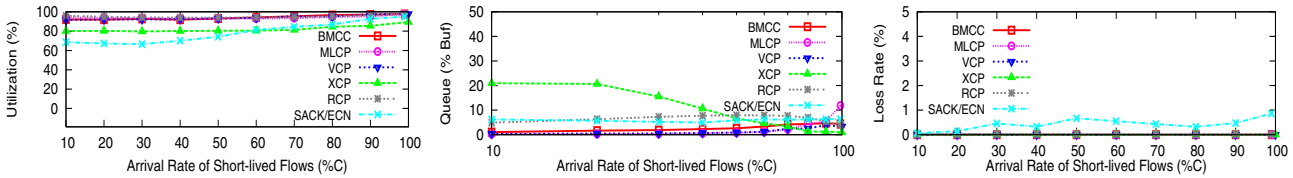


Fig. 13. Impact of varying the offered load of short-lived, web-like traffic from $0.1C_l$ Mbps to C_l Mbps, where $C_l = 155$ Mbps.

which reduces the AFCT. With SACK's slow-start algorithm, flows use a fixed MI factor of two. In high load, this is too aggressive, inducing a high loss rates which increases the AFCT. XCP increases the AFCT flows because new flows apply AIMD. RCP gives higher rates to new flows which helps short flows to finish quickly.

ii) Varying Traffic load: Assuming an average file size of 30 KB for web-like flows, we now vary their offered load from 0% to 100% of the bottleneck capacity while maintaining 5 long-lived flows in either direction. Figure 13 shows that BMCC, MLCP, VCP and RCP are able to maintain high utilization ($\geq$90%) while maintaining low persistent queue length ($\leq$5% BDP) and negligible packet loss rate. Average utilization with XCP is less $\sim$10% when compared with BMCC and for SACK the difference is as large as 25% under low loads. Average queue length for XCP is higher (10$\sim$25% BDP) for loads $<$50%. The relatively small loss rate for SACK $\sim$0.5%-1% is due to the presence of RED/ECN at the routers.

e) Fairness: We now compare the fairness properties of BMCC with other schemes. We consider a single bottleneck link of capacity 60 Mbps with 20 long-lived flows in either direction. Each forward flow j's RTT is chosen according to $T_j = 40 + 4j\delta$ ms for $j = 0,\ldots,19$, where δ is the one-way propagation delay for a non-bottleneck link. We perform

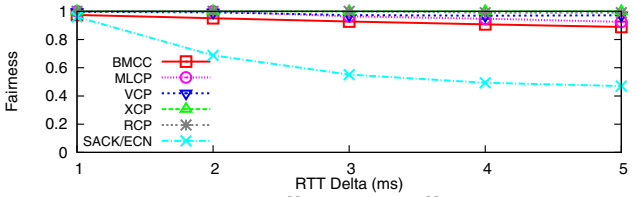


Fig. 15. Jain's fairness index $[(\sum_{i=1}^{N} x_i)^2/(N\sum_{i=1}^{N} {x_i}^2)$, where x_i is the throughput of flow i and $i \in \{1,..,N\}]$ as a function of δ.

simulations with δ varying from 1 ms to 5 ms. 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 and RCP achieve high level of fairness ($\geq$0.9) across a large range of RTT variations. SACK, however, becomes very unfair in the presence of RTT heterogeneity.

VIII. Related Work

ADPM and BMCC build on existing packet marking schemes and congestion control protocols. Notable related work is as follows.

a) Packet Marking Schemes: The first schemes to convey a continuous value using a single bit per packet [10], [20], [21] randomly mark packets with a probability dependent on the

value to be sent. These schemes require at least $n-1$ packets to signal n different values, like unary coding, and implicitly sum the values at routers on the path.

The "side information" contained in the IP header was first used for setting the ECN bits in [11], which considers the time-to-live (TTL) field. This idea was extended by Thommes and Coates [12], who provided an efficient, deterministic marking algorithm, using the 16-bit IP packet identifier ("IPid") to allow the value to be encoded base-two. Following that, [22] proposed a similar scheme for estimating the maximum value, appropriate for max-min flow control. These deterministic marking schemes provide estimators that potentially have a much lower MSE than the random marking schemes [12], [22].

All these schemes must specify *a priori* how to trade resolution for agility. Random marking schemes must choose the interval over which to average random marks, while deterministic schemes use a fixed quantizer, which means that the MSE is poor until sufficient packets have been processed [9].

ADPM implicitly adapts its effective quantization resolution based on the dynamics of the value. Analysis and numerical results in [9], [15] show that static values can be estimated precisely, whilst rapidly changing values can be tracked quickly. These results also show that ADPM provides a MSE that is several orders of magnitude smaller than the estimators based on random marking of packets [10], [20], [21] or deterministic marking with static quantization [12], [22].

b) Congestion Control Protocols: In RCP [5], 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. XCP regulates the sending rate by making routers send precise window increment/decrements in feedback to each flow [4]. 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 [14]. MLCP uses 4-bits for conveying load factor information and achieves near-optimal performance performance in terms of convergence to efficiency [8]. VCP uses two bits to convey load factor information. Hence, its rate of convergence to efficiency is slow, and it considerably stretches the AFCT of short flows.

Pure end-to-end schemes such as HighSpeed TCP [1], FAST [2], PCP [23], CTCP [24] and CUBIC [25] do not require explicit feedback from the routers. Therefore, it is hard for them to remain efficient and fair while keeping low queues and negligible loss rates. BMCC uses the existing ECN bits for congestion-related feedback and is able to achieve all these goals in typical network scenarios.

IX. Conclusion

This paper has presented the design, analysis and simulation results for BMCC; a congestion control protocol that uses a packet marking scheme to obtain congestion estimates of up to 16-bit resolution using only the existing ECN bits. BMCC achieves high utilization, low persistent queue length and negligible packet loss rate on high BDP paths and enables flows with different round-trip times to achieve max-min fairness. BMCC outperforms SACK+RED/ECN, VCP, MLCP, XCP and in some cases RCP in terms of flow completion times for Internet flow sizes.

X. Acknowledgements

This work was supported by NSF grants 010536 and 010684, and the Australian Research Council (ARC), grant DP0985322.

References

[1] S. Floyd, "Highspeed TCP for large congestion windows," in *IETF RFC 3649*, Dec 2003.
[2] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, architecture, algorithms and performance," in *IEEE INFOCOM*, Mar 2004.
[3] 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
[4] D. Katabi, M. Handley, and C. Rohrs, "Internet congestion control for high bandwidth-delay product networks," in *ACM SIGCOMM*, Aug 2002.
[5] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown, "Procesor sharing flows in the internet," in *IWQoS*, Jun 2005.
[6] K. K. Ramakrishnan and S. Floyd, "The addition of explicit congestion notification (ECN) to IP," in *IETF RFC 3168*, Sep 2001.
[7] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One more bit is enough," in *ACM SIGCOMM*, Aug 2005.
[8] I. A. Qazi and T. Znati, "On the design of load factor based congestion control protocols for next-generation networks," in *IEEE INFOCOM*, Apr 2008.
[9] 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.
[10] S. Athuraliya, V. Li, S. Low, and Q. Yin, "REM: Active queue management," in *IEEE Network, 15(3):48-53*, May 2001.
[11] M. Adler, J. yi Cai, J. K. Shapiro, and D. Towsley, "Estimation of congestion price using probabilistic packet marking," in *in Proc. IEEE INFOCOM*, Mar-Apr 2003.
[12] R. W. Thommes and M. J. Coates, "Deterministic packet marking for time-varying congestion price estimation," *IEEE/ACM Trans. Networking*, vol. 14, no. 3, pp. 592–602, June 2006.
[13] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," in *IEEE/ACM Trans. Networking, 1(4):397-413*, Aug 1993.
[14] 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):87-98*, Feb 2000.
[15] L. L. H. Andrew and S. V. Hanly, "The estimation error of adaptive deterministic packet marking," in *Proc. Allerton Conf. Communication, Control and Computing*, Urbana-Champaign, IL, Sept. 2006.
[16] V. Paxson, "End-to-end internet packet dynamics," in *ACM SIGCOMM*, Sep 1997.
[17] 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.
[18] R. Shorten, F. Wirth, and D. Leith, "A positive systems model of TCP-like congestion control: Asymptotic results," *IEEE/ACM Trans. Networking*, vol. 14, pp. 616–629, 2006.
[19] "ns-2 Network Simulator," http://www.isi.edu/nsnam/ns/.
[20] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, 1997.
[21] S. H. Low and D. E. Lapsley, "Optimization flow control I: Basic algorithm and convergence," *IEEE/ACM Trans. Networking*, vol. 7, pp. 861–875, 1999.
[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] T. Anderson, A. Collins, A. Krishnamurthy, and J. Zahorjan, "PCP: Efficient endpoint congestion control," in *NSDI'06*, 2006.
[24] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," in *IEEE INFOCOM*, Apr 2006.
[25] I. Rhee and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," in *PFLDNet'05*, Feb. 2005.