

# Incremental deployment of new ECN-compatible congestion control

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

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

## ABSTRACT

Congestion control can be improved by using more accurate congestion feedback from the network. However, when new protocols either do not obtain information from all congested network elements, or share resources with existing congestion control protocols, it is possible for one or other to obtain unfairly low throughput. This paper investigates the performance of Binary Marking Congestion Control when deployed in conjunction with SACK and Drop-Tail or ECN routers. We propose and evaluate solutions that allow for a fairer bandwidth sharing between BMCC and SACK flows in these cases. The proposed solutions can also be applied to other protocols which use explicit feedback from the network.

## 1. INTRODUCTION

Current implementations of the Transmission Control Protocol (TCP) respond to congestion once it has become bad enough to overflow network buffers, or at least to form significant standing queues. It is desirable for congestible devices, such as routers, to signal the TCP agents about incipient congestion before it becomes bad enough to affect the quality of service. Many protocols have been proposed for this purpose, but most have problems with incremental deployment.

Deployment of protocols that use explicit feedback from the network requires changes in the routers. These changes, however, cannot be done overnight. Thus, it is important that a new protocol is able to run over existing infrastructure, and share routers with existing protocols. Moreover, it is also important that a protocol performs well when congestion occurs at a device, such as a firewall, which does not provide any feedback.

Since many new protocols aim to avoid queueing, they are likely to be starved by Reno-like flows which continue to increase their rate until the buffer overflows. Conversely, if a new protocol decreases its rate less (or increases more) than Reno, then it is likely to starve Reno-like flows if a congested resource does not provide feedback.

It is often proposed that the former problem be solved by having separate queues for packets which do and don't support the new protocol [15]. However, that raises complex management issues and unnecessarily increases the cost. Many proposals for high bandwidth-delay products include a "compatibility mode" which revert to Reno-like behavior when the bandwidth-delay product is low. Similarly, it is possible for a new protocol to seek to detect the presence of congestion at points which do not provide explicit feedback, and revert to a loss-based mode. Problems may arise if the detection process fails, but such mode switching may have a useful role in the migration towards more efficient congestion control.

This paper evaluates and improves the incremental deployability of Binary Marking Congestion Control (BMCC), a recently pro-

posed ECN-compatible congestion control protocol [10, 11]. The proposed solutions allow BMCC to safely coexist with standard TCP traffic in the same queue. These solutions have applicability beyond BMCC.

The rest of the paper is organized as follows. In Section 2, we discuss the motivation behind BMCC. In Section 3, we describe the BMCC protocol in detail. We analyze the performance of BMCC under different deployment scenarios in Section 4. In Section 5, we propose and evaluate solutions to overcome incremental deployment issues. We offer concluding remarks in Section 6.

## 2. WHY BMCC?

BMCC is one of the many protocols which use network feedback, but it is one particularly suited to deployment in the Internet because its signalling is compatible with the existing Internet Protocol (IP) packet header.

Many protocols, such as XCP [19], RCP [15], MaxNet [14] and MLCP [12] require that routers send a (quantized) real number indicating the amount of congestion. This requires additional fields, either in an IP option, a TCP option [14] or modified header [19], or a "shim layer" [15]. None of these can be universally deployed because many routers are configured to drop packets containing IP options, and IP payloads may be encrypted.

An alternative, used by ECN [13] and VCP [18] is to squeeze two extra bits into the IP header. Because they signal only binary [13] or ternary [18] congestion indication, these schemes provide minimal benefit, although [18] allows low mean queue size at the expense of slow convergence [11].

BMCC uses these bits to send a continuous congestion signal by coding the value over multiple packets, using ADPM [2]. This is an enhancement of random marking [7, 3, 1] and of Thommes and Coates's scheme [16] to achieve a wider dynamic range of signals using few packets. This allows BMCC to be IP-compatible while achieving fast convergence. BMCC achieves efficient and fair bandwidth allocations on high bandwidth-delay product paths while maintaining low queues, negligible packet loss rate and small average flow completion times [10, 11].

## 3. BMCC

The IP header contains two Explicit Congestion Notification (ECN) bits for providing feedback from the routers to the sources. An arriving packet with ECN bits set to  $(00)_2$  indicates that the source is not ECN compatible, whereas the symbols  $(01)_2$  and  $(10)_2$  signify an ECN-capable transport. The ECN bits on an unmarked packet are initially  $(10)_2$  and the routers set these bits to  $(11)_2$  in order to indicate congestion [13].

BMCC [11] employs ADPM, a packet marking scheme, which uses these bits 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.

### 3.1 BMCC Router

During every time interval  $t_p = 200$  ms, each BMCC router computes the load factor on each of its output links as

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

where  $\lambda$  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 [18, 3, 5, 6].

### 3.2 BMCC Receiver and ADPM

The router conveys its load factor to the sender by applying ADPM [2] to the ECN bits. Let  $u$  be the maximum value of  $f$  that ADPM can signal. If  $f \geq u$  or the packet already contains a mark  $(11)_2$ , then BMCC marks the packet with  $(11)_2$ . 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 of the received packet. Note that the receiver's estimate will lag behind the true value [17], 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  $\eta_0 = 15\%$  are rounded up to  $\eta_0 = 15\%$ , and the hash is such that 1/4 of packets hash to values in  $(\eta_0, \eta)$  for some design parameter  $\eta$ , 1/4 of packets hash to  $(\eta, 1)$  and 1/2 hash to  $(1, u)$ .

### 3.3 BMCC Sender

BMCC uses the following control laws based on whether the most loaded link is lightly-, heavily- or over-loaded, corresponding to  $f \in [0, \eta)$ ,  $[\eta, 1)$  or  $[1, \infty)$ , where  $\eta = 75\%$ .

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

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

where  $T$  is the RTT of the flow,  $\xi(\hat{f}) = \kappa_2(1 - \hat{f})/\hat{f}$  and  $\kappa_2 = 0.35$ . As shown in [11], this rule yields a concave negative exponential window growth function. BMCC aims to give equal rate to flows with different RTTs. Since flows with large RTTs update less often, the rule

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

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

**High Load** ( $\eta \leq \hat{f} < 1$ ): When the system has achieved high utilization, the algorithm must seek fairness. This is achieved using

AIMD. In high load, sources apply AI:

$$w(t+T) = w(t) + \alpha, \quad (4)$$

with  $\alpha = (T/t_p)^2$  chosen to cause the equilibrium window to be proportional to the flow's RTT, giving RTT fairness [18].

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

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

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

varies linearly in  $[\beta_{\min} = 0.65, \beta_{\max} = 0.875]$ ,  $u = 1.2$ , and  $\Delta\beta = \beta_{\max} - \beta_{\min}$ .

**Packet loss:** The original description of BMCC [11] did not specify its response to packet loss, since BMCC does not induce self-loss like Reno does. However, this response is important when operating over non-BMCC equipment. The response to a loss is the same as the response to a  $(11)_2$  mark, except that the window is reduced by a factor of 2. SACK is used for loss recovery.

### 3.4 Illustration

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 three BMCC flows that arrive with an inter-arrival time of 100 s and have round-trip propagation delays of 150 ms, 200 ms and 250 ms, respectively. Observe that the three flows rapidly achieve rates that are within 70% of their fair share before converging to a fair ( $\approx 50$  Mbps each) and efficient ( $\approx 100$  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.

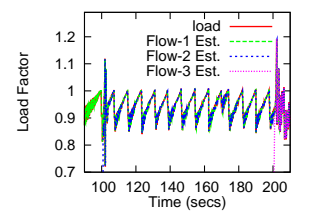
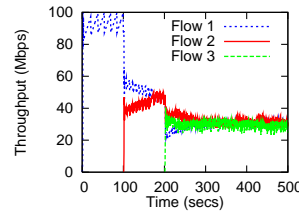


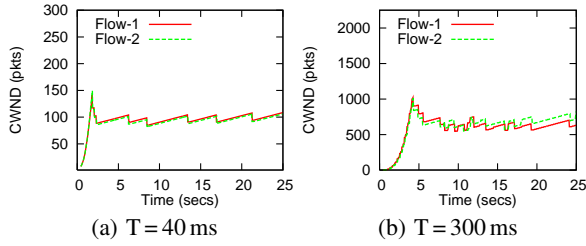
Figure 1: Throughput of three flows on a 100 Mbps link

Figure 2: Comparison of the load factor at the bottleneck and the flows' estimates of it

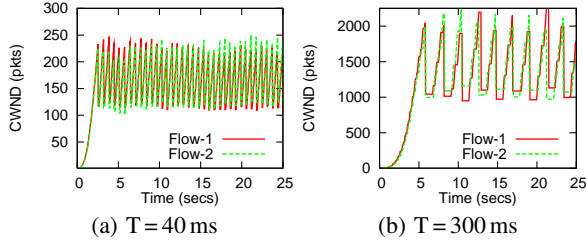
## 4. DEPLOYMENT EVALUATION

In this section, we study the performance of BMCC under different partial deployment scenarios and protocol mixes using ns-2 simulations. First, we study the performance of BMCC under Drop-Tail and RED (with ECN support) routers. We then add SACK flows to the existing BMCC traffic and study their interaction under different bottlenecks. Second, we study the performance of a mix of protocols under BMCC-enabled bottleneck.

In all simulations, the non-bottleneck routers are assumed to be Drop-Tail. Unless explicitly stated otherwise, the bottleneck capacity,  $C$ , and the round-trip propagation delay,  $T$ , are set to 45 Mbps and 40 ms, respectively. The capacity of the non-bottleneck links is set to  $10 \cdot C$  Mbps. The buffer size of all routers is set to the bandwidth-delay product. We always maintain bidirectional cross



**Figure 3: Congestion window size of two BMCC flows passing through a BMCC router**



**Figure 4: Congestion window size of two BMCC flows passing through a Drop-Tail router**

traffic, with an offered load equal to 10% of the bottleneck capacity. The cross traffic arrives according to a Poisson process and has sizes that follow the Pareto distribution with an average file size of 30 kB and a shape parameter of 1.2 [8]. All flows use a data packet size of 1 kB.

## 4.1 Performance over non-BMCC routers

We now evaluate the performance of BMCC under different bottlenecks while considering a single bottleneck, dumbbell topology. To aid comparison under different bottleneck types, we first analyze the performance of BMCC under BMCC-enabled bottleneck routers. Figure 3 shows the variations in the congestion window size of two BMCC flow traversing a BMCC-enabled bottleneck router. The source uses the precise load factor estimates it receives via ADPM to adjust its congestion window size. Observe that BMCC introduces little variations in the flows' rates.

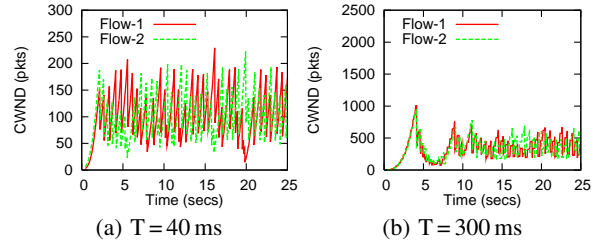
### 4.1.1 BMCC over Drop-Tail

Figure 4 shows the variations in the congestion window size of two BMCC flows, passing through a Drop-Tail bottleneck, for  $T = 40$  ms and  $T = 300$  ms. In this case, sources increase their windows by a factor of 3 per  $t_p$  until a packet gets dropped at the bottleneck (since new BMCC flows assume the initial value of  $f$  to be 15%). This cycle gets repeated because there is no change in the receivers' estimates<sup>1</sup>. In other words, BMCC employs MIMD with a MI factor of  $3^{T/t_p}$  per round-trip time and a MD factor of 0.5. In the presence of synchronous feedback, it has been shown that MIMD may not converge to a fair bandwidth allocation [4].

### 4.1.2 BMCC over RED+ECN

Figure 5 shows the variations in the congestion window size of two BMCC flows traversing a RED (with ECN support) bottleneck router. In this case, BMCC flows increase their congestion windows by a factor of  $3^{T/t_p}$  per  $T$  until overload. In overload, BMCC

<sup>1</sup>Note that with drop-tail routers, all received packets will be unmarked. Hence, there won't be any change in  $\hat{f}$  because  $h > \hat{f}, \forall h$ .



**Figure 5: Congestion window size of two BMCC flows passing through a RED router with ECN support**

sources apply MD with a factor of 0.65 (if there is no packet loss) because RED routers mark the ECN bits with the  $(11)_2$  symbol that is interpreted by BMCC as severe overload. After which, receiver's estimate decreases until it approaches 15%. Observe that congestion window sizes assumed by BMCC flows are lower than in the case of Drop-Tail routers. This is because RED signals congestion earlier than via packet drops. However, note that higher congestion window sizes in case of Drop-Tail routers do not necessarily imply higher throughput. In fact, increasing the congestion window beyond a certain threshold (depending on the aggressiveness of the source control laws) only increases the RTT of flows without changing the flow throughput. This hurts the performance of other flows due to increased queueing delays.

### 4.1.3 Mix of Protocols over Drop-Tail and RED

We now mix three BMCC flows with three SACK flows and analyze their performance under different bottlenecks. Figure 6 shows the congestion window sizes under Drop-Tail and RED routers. With Drop-Tail routers, BMCC flows grab a much large share of bandwidth than SACK since SACK uses AIMD whereas BMCC uses a much more aggressive MIMD. Note that SACK flows get completely starved when  $T = 300$  ms.

In case of RED (with ECN support) routers, performance is similar to the Drop-Tail case. SACK flows achieve small congestion window sizes when  $T = 40$  ms and get completely starved when  $T = 300$  ms.

## 4.2 Performance over BMCC routers

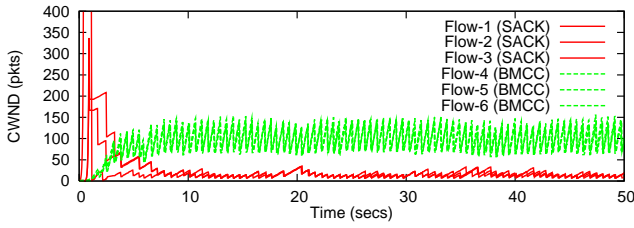
We now consider a mix of protocols while assuming a BMCC-enabled bottleneck router.

### 4.2.1 BMCC and SACK

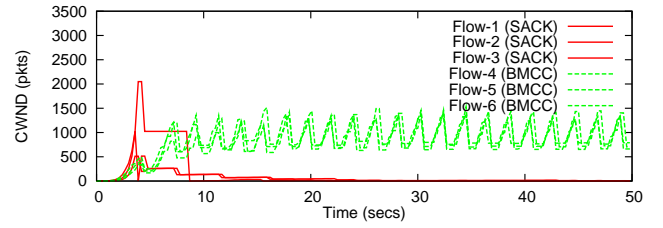
We generate 3 SACK and 3 BMCC flows while retaining all the settings as in the previous experiments. The SACK flows arrive between 10 s and 12 s. Observe that the SACK flows completely starve BMCC flows (see Figure 7). This happens because SACK fills router buffers until a packet loss and thus maintains a high bottleneck queue. Since BMCC does not rely on packet loss as a signal of congestion, this gives rise to a sustained increase in the load factor, causing BMCC flows to back-off very frequently.

### 4.2.2 BMCC and SACK+ECN

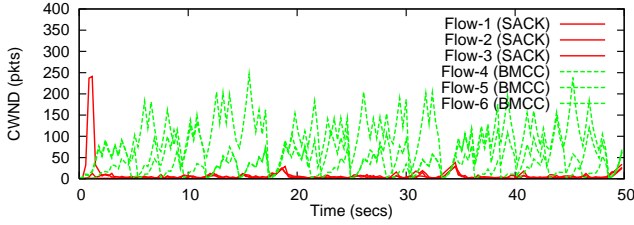
We now generate 3 ECN-compatible SACK flows and 3 BMCC flows while retaining all the settings as in the previous experiments. Figure 8 shows that in this case also, the BMCC flows get completely starved by SACK flows. In this case, when the load factor exceeds 100%, BMCC flows back-off probabilistically using ADPM. Since ADPM allows quick detection of overload, this causes BMCC flows to back-off. However, SACK flows continue to grab more bandwidth by increasing their rate until load factor exceeds



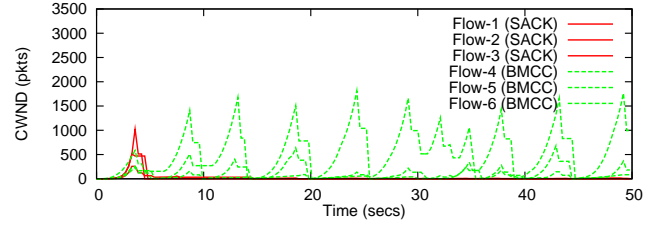
(a)  $T = 40$  ms, Bottleneck = Drop-Tail



(b)  $T = 300$  ms, Bottleneck = Drop-Tail



(c)  $T = 40$  ms, Bottleneck = RED+ECN



(d)  $T = 300$  ms, Bottleneck = RED+ECN

Figure 6: Congestion window size of 3 BMCC and 3 SACK flows sharing a common, non-BMCC bottleneck

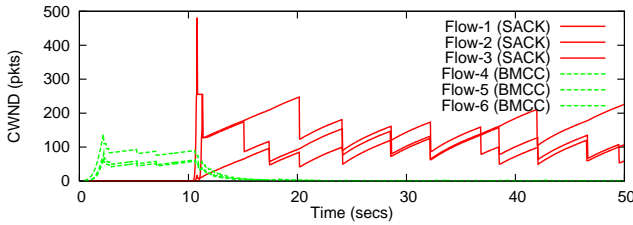


Figure 7: 3 SACK and 3 BMCC flows sharing a BMCC-enabled bottleneck link

120% (after which SACK flows receive the  $(11)_2$  mark and back-off). When  $\hat{f} \in [100, 120]$ , BMCC flows back-off repeatedly, leading to very low congestion window sizes.

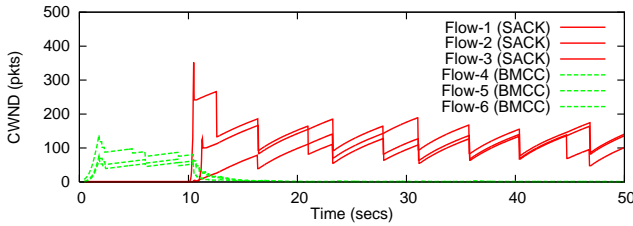


Figure 8: 3 SACK+ECN and 3 BMCC flows sharing a BMCC-enabled bottleneck link

## 5. ENABLING DEPLOYMENT OF BMCC

In the previous section, we observed that SACK flows (with and without ECN support) starve BMCC flows. This is due to the fact that SACK flows maintained high persistent queue length, leading to sustained value of load factor above 100%. This caused BMCC flows to back-off multiple times, leading to starvation. In this section, we discuss a number of solutions and their shortcomings. We

then propose one solution and show that it is effective in handling the issues raised in the previous section.

### 5.1 Deployment over BMCC bottlenecks

**Deterministically marking/dropping packets in overload:** One possible solution to the above problems is to disable ADPM in overload and mark ECN-capable transports with the  $(11)_2$  symbol and drop packets from non-ECN-capable transports. However, this change will force BMCC flows to use a single back-off factor. This will remove the benefit that BMCC has of dynamically adapting the back-off factors depending on the load at the bottleneck, which has a significant impact on network utilization and convergence rates. Moreover, this solution is likely to starve or result in very low throughput for SACK flows with  $T \ll 200$  ms. The reason is that BMCC routers keep load factor estimates for  $t_p = 200$  ms. A small RTT flow (e.g., with  $T = 40$  ms) will reduce its congestion window multiple times in one  $t_p$ , leading to very small windows. Note that reducing  $t_p$  is not likely to help because that would cause large variations in traffic loads due to the burstiness induced by sources with  $T \gg t_p$ .

**Probabilistically marking packets with the  $(11)_2$  symbol:** We can prevent starvation of BMCC flows when sharing a link with SACK+ECN flows by probabilistically marking (or dropping) packets with the  $(11)_2$  symbol when  $f \in [100\%, 120\%]$ . Packets can be marked (or dropped) if a packet was marked (with  $(01)_2$ ) using ADPM. However, this would cause most BMCC flows to back-off by a factor of 0.65, thus reducing the benefit of small decreases in low overload. Also, ADPM yields a marking/dropping probability close to 0.5 which is too large and is likely lead to multiple window reductions as in the above case. To remedy this situation, one could mark/drop packets with low probability when overload is small and increase this probability as  $f$  approaches 120%.

### 5.2 Modified BMCC router

To retain the benefit of using ADPM in overload for BMCC traffic while preventing starvation of SACK and BMCC flows when traversing a common BMCC-enabled bottleneck, the following ingredients are desirable in a solution: (1) packets should be marked (or dropped) probabilistically rather than deterministically and (2) the fact that BMCC routers keep load factor estimates for one  $t_p$

should not impact other flows, which suggests that the marking procedure should be decoupled from load factor computation.

To achieve the above ends, we make the following modifications to the BMCC router. In overload, we replace the packet marking policy in BMCC routers with the Adaptive RED algorithm<sup>2</sup> while continuing to mark packets with ADPM using load factor computation. Recall, ADPM *only* marks packets with the  $(01)_2$  symbol and therefore, doesn't interfere with the RED algorithm or the standard ECN behavior. This ensures that BMCC sources continue to obtain precise load factor estimates. To retain the benefit of multiple back-off factors, BMCC sources are made to back-off only when they receive the  $(11)_2$  symbol and  $\hat{f} \geq 1$ , where the back-off factor depends solely on the load factor estimate at the receiver. Now since the marking (or dropping) probability is the same for SACK and BMCC flows, they back-off roughly the same number of times leading to a more fair bandwidth sharing.

Figure 9 shows the congestion window sizes of three SACK flows sharing a BMCC-enabled bottleneck with a modified BMCC router. Observe that SACK flows are now able to effectively share bandwidth with BMCC traffic. The BMCC flows obtain the benefit of multiple MDs by applying small back-off factors in low overloads whereas SACK behavior remains largely unchanged. Figure 10 shows the congestion window sizes of flows on a path with  $T = 300$  ms. The bandwidth-delay product of this path is about 1700 packets, which is roughly eight times larger than the previous scenario. In this case, BMCC obtains much larger throughput than SACK flows because it acquires the spare bandwidth much faster than SACK flows.

In order to achieve max-min fairness, BMCC flows scale their MI and AI parameters. This scaling depends on the ratios  $T/t_p$  and  $(T/t_p)^2$  for MI and AI, respectively. This implies that on long RTT paths, BMCC flows will achieve higher throughput than SACK flows due to larger  $\alpha$ ,  $\beta$ , and  $\xi$  parameter values<sup>3</sup>. On the other hand, it is well-known that SACK flows achieve throughput proportional to  $1/RTT^z$ , where  $1 \leq z \leq 2$  [21]. This implies that with SACK, short RTT flows can achieve much higher throughput than long RTT flows. We now compare the unfairness in these two cases. In the first case, we characterize the throughput achieved by BMCC and SACK flows when they share a bottleneck and in the second case, we characterize the throughput achieved by two SACK flows with different RTTs.

To quantify unfairness, we measure the gain,  $G$ , in the throughput of a more aggressive flow at the expense of a loss,  $L$ , in the throughput of a lesser aggressive flow. In the first experiment, called *BASELINE*, we run two SACK flows sharing a 45 Mbps link with a round-trip propagation delay of  $T$ . In the second experiment, we run one SACK flow and one BMCC flow, each RTT,  $T$ . In the third experiment, we run one SACK flow with RTT,  $T$  and another SACK flow with an RTT of 10 ms. The gain achieved by the aggressive flow and the loss incurred by a less aggressive flow is given by

$$G = T(B)_{Mix} / T(B)_{Baseline},$$

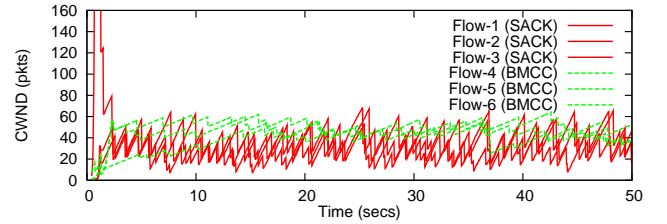
$$L = T(A)_{Mix} / T(A)_{Baseline}$$

respectively, where  $T(A)$  is the throughput of a less aggressive flow and  $T(B)$  of a more aggressive flow [20].

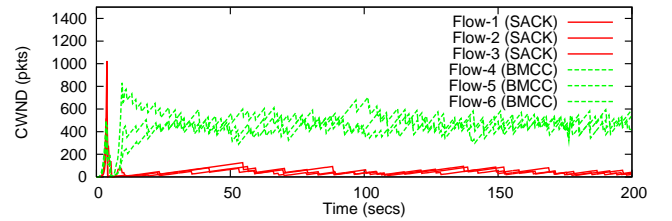
<sup>2</sup>We chose RED because it is the most widely deployed AQM scheme. However, any AQM could be used.

<sup>3</sup>Note that this does not necessarily mean that BMCC flows do not achieve higher than SACK throughput on very small RTT paths. The fast MI phase and high  $\beta$  values used by BMCC allows it to achieve high throughput in such cases.

Figure 11(a) shows that the bandwidth gained by a BMCC flow over a SACK flow is less than the bandwidth gained by a SACK with  $T = 10$  ms, across a range of round-trip times. Figure 11(b) shows that the bandwidth lost by SACK flows is roughly an order of magnitude larger when the bottleneck is shared by a 10 ms RTT flow compared to a BMCC flow.



**Figure 9: 3 SACK+RED/ECN and 3 BMCC flows sharing a single modified BMCC bottleneck link ( $T = 40$  ms)**



**Figure 10: 3 SACK+RED/ECN and 3 BMCC flows sharing a single modified BMCC bottleneck link ( $T = 300$  ms)**

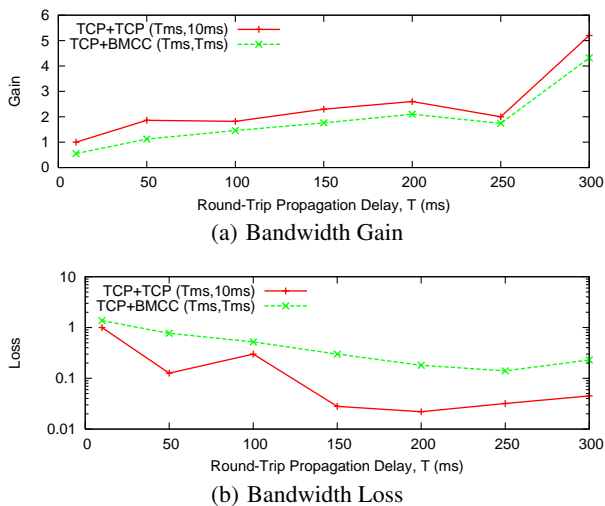
### 5.3 Deployment over non-BMCC bottlenecks

Simulation results in the previous section showed that when the path of a flow contained only non-BMCC routers, BMCC applies MIMD. This leads to degraded throughput and in some cases starvation for SACK flows. To remedy this situation, we now present heuristics for detecting non-BMCC bottlenecks for switching to SACK TCP. When a BMCC router is congested, it signals a high load factor,  $f$ . If other signals of congestion, such as packet loss or  $(11)_2$  marks are detected when the load factor is low, this indicates congestion at a non-BMCC router. We propose the following rules:

- If a BMCC source detects a packet loss or  $(11)_2$  mark when its estimate of the bottleneck load factor is less than 100%, then it falls back to SACK operation.
- If a BMCC source detects a load factor in excess of 100%, it exits SACK compatibility mode and resumes normal operation.

Figure 12(a) shows the congestion window size of two BMCC flows (with this heuristic detection rule) traversing a bottlenecked Drop-Tail router. Note that unlike as in Figure 4, BMCC quickly detects the condition and switches to SACK.

If a BMCC flow halves its window in response to the packet which caused it to enter SACK mode, this rule gives rise to the anomalous situation that the window reduction is *greater* if  $\hat{f} < 1$  than if  $\hat{f} > 1$ . Thus, we propose that the loss or mark which triggers the entrance to SACK mode is not itself interpreted as a congestion signal.



**Figure 11: Bandwidth Gain and Loss for a BMCC (RTT=T ms) and a SACK (RTT=10 ms) flow over a SACK flow (RTT=T ms)**

Note also that the estimate of  $f$  signalled by ADPM lags the value signalled by the routers [17], it is possible for BMCC router to estimate the load factor to be slightly below 100% when the router's value is actually greater, and its RED policy emits a (11)<sub>2</sub> mark. This will cause the flow to mistakenly enter SACK mode. However, it will return to normal mode soon afterwards, as the load factor is already high.

Clearly, this mechanism needs more testing and refinement, but it is a useful approach for allowing co-existence of BMCC and non-BMCC flows running over non-BMCC routers.

## 6. CONCLUSION

In this paper, we have shown that BMCC can be incrementally deployed on the Internet, without changing the IP Header or the addition of a "shim" layer. End-host and router softwares can be gradually updated in a similar way as ECN was deployed. End-hosts using BMCC can immediately get high throughput and faster downloads when traversing BMCC-enabled bottlenecks without significantly hurting standard TCP traffic. This is an important incentive for end-users and router vendors to deploy BMCC. Moreover, research studies have shown that the throughput of most Internet users is limited by their "last mile" or access links. This suggests that BMCC's deployment can begin by updating such links and users can immediately take advantage of BMCC's high performance.

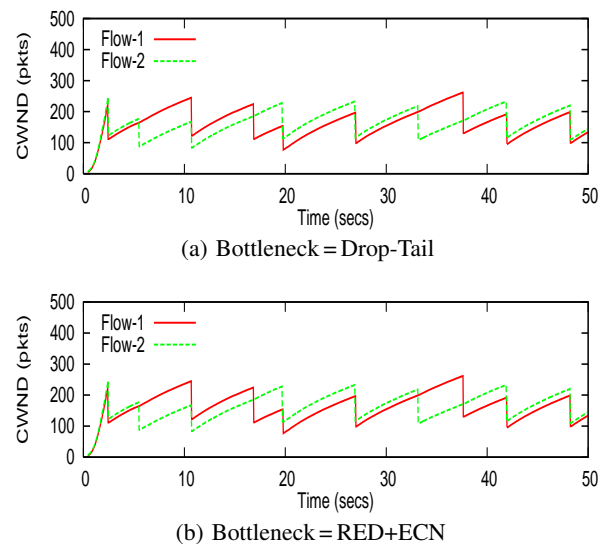
We evaluated the performance of BMCC under different deployment scenarios and a mix a protocols. We showed that when the bottleneck is not BMCC-enabled, SACK flows achieve low throughput and can even get starved. On the other hand, when the bottleneck is BMCC-enabled, SACK flows starve BMCC flows. We proposed and evaluated solutions which prevent starvation and allow for a fairer bandwidth sharing between BMCC and SACK flows.

## 7. ACKNOWLEDGEMENTS

This work was supported by NSF grants 010536 and 010684.

## 8. REFERENCES

[1] Micah Adler, Jin-yi Cai, J. K. Shapiro and Don Towsley. Estimation of Congestion Price Using Probabilistic Packet Marking. In *IEEE INFOCOM*, Mar-Apr 2003.



**Figure 12: Two BMCC flows (with heuristic detection) sharing a non-BMCC bottleneck (T = 40 ms).**

- [2] L. L. H. Andrew, S. V. Hanly, S. Chan, and T. Cui. Adaptive deterministic packet marking. *IEEE Comm. Letters*, 10(11):790–792, Nov 2006.
- [3] S. Athuraliya, V. Li and S. Low and Q. Yin. REM: Active Queue Management. In *IEEE Network*, 15(3):48–53, May 2001.
- [4] Dah-Ming Chiu and R. Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. In *Comput. Netw. ISDN Syst.*, 17(1):1–14, 1989.
- [5] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. In *IEEE/ACM Trans. Networking*, 1(4):397–413, Aug 1993.
- [6] S. 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.
- [7] Frank Kelly. Charging and Rate Control for Elastic Traffic. In *European Trans. Telecommunications*, 8:33–37, 1997.
- [8] M. Crovella and A. Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. In *IEEE/ACM Trans. Networking*, Dec 1997.
- [9] N. Dukkkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown. Processor Sharing Flows in the Internet. In *IWQoS*, Jun 2005.
- [10] Ihsan A. Qazi, L. L. H. Andrew and T. Znati. Two bits are enough. In *ACM SIGCOMM*, Aug 2008 (poster).
- [11] Ihsan A. Qazi, L. L. H. Andrew and T. Znati. Congestion Control Using Efficient Explicit Feedback In *IEEE INFOCOM*, Apr 2009.
- [12] Ihsan A. Qazi and T. Znati. On the Design of Load Factor based Congestion Control Protocols for Next-Generation Networks. In *IEEE INFOCOM*, Apr 2008.
- [13] K. K. Ramakrishnan and S. Floyd. The Addition of Explicit Congestion Notification (ECN) to IP *IETF RFC 3168*, Sep 2001.
- [14] M. Suchara, L. L. H. Andrew, R. Whitt, K. Jacobsson, B. P. Wyrowski and S. H. Low. Implementation of Provably-Stable Explicit Congestion Control. In *Broadnets*, 2008.
- [15] C. H. Tai, J. Zhu and N. Dukkkipati. Making large scale deployment of RCP practical for real networks In *IEEE INFOCOM Mini-Conference*, Apr 2008.
- [16] R. W. Thommes and M. J. Coates. Deterministic Packet Marking for Time-Varying Congestion Price Estimation. In *IEEE/ACM Trans. Networking*, 14(3):592–602, Jun 2006.
- [17] L. L. H. Andrew and Stephen V. Hanly The Estimation Error of Adaptive Deterministic Packet Marking. In *Allerton Conf. Communication, Control and Computing*, Sep 2006.
- [18] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman. One More Bit Is Enough. In *ACM SIGCOMM*, Aug 2005.
- [19] D. Katabi, M. Handley and C. Rohrs. Internet Congestion Control for High Bandwidth-Delay Product Networks. In *ACM SIGCOMM*, Aug 2002.
- [20] L. L. H. Andrew, S. Floyd and W. Gang. Common TCP Evaluation Suite. *Internet draft (work in progress)*, Jul 2008. URL: <http://netlab.caltech.edu/lachlan/abstract/draft-irtf-tmrg-tests-00.html>.
- [21] T. V. Lakshman, U. Madhow, and B. Suter. TCP/IP performance with random loss and bidirectional congestion. In *IEEE/ACM Trans. Networking*, 8:541–555, 2000.