Modest BBR: Enabling Better Fairness for BBR Congestion Control

Yuxiang Zhang*, Lin Cui*, Fung Po Tso†
*Department of Computer Science, Jinan University, Guangzhou, China
†Department of Computer Science, Loughborough University, LE11 3TU, UK
Email: samuelzyx0924@gmail.com; tcuilin@jnu.edu.cn; p.tso@lboro.ac.uk;

***Abstract*—As a vital component of TCP, congestion control defines TCP's performance characteristics. Hence, it is important for congestion control to provide high link utilization and low queuing delay. Recent BBR tries to estimate available bottleneck capacity to achieve this goal. However, its aggressiveness characteristics generate a massive amount of packet retransmission which harms loss-based congestion control protocol such as Cubic. In this paper, we first dive into this issue and reveal that the aggressiveness of BBR can degrade the performance of Cubic, as well as the overall Internet transmission. Then we present *Modest BBR*, a simple yet effective solution based on BBR, by responding to retransmission less aggressively. Through extensive testbed experiments and Mininet simulation, we show *Modest BBR* can preserve high throughput and short convergence time while improve the overall performance when coexisting with Cubic. For example, *Modest BBR* gets similar throughput compared to BBR, while it improves 7.1% of the overall throughput and achieves better fairness to loss-based schemes.**

***Index Terms*—Congestion control, Retransmission, Inter-protocol fairness**

I. Introduction

Congestion control protects the Internet from persistent overload situations and provides high utilization and low queuing delay. Since its invention and Internet-wide deployment, congestion control has grown out of its infancy, but is still a hot topic of today's research. In general, congestion control mechanisms try to determine a suitable amount of data to transmit at a certain point in time in order to utilize the available transmission capacity, but also to avoid a persistent overload of the network [1][2]. BBR [3], which was proposed and developed by researchers in Google recently, determines the pacing rate based on bandwidth-delay product (BDP). With extensive experimental evaluations, it is demonstrated that BBR can achieve high throughput and low latency [3]. Some studies show that BBR work quite well for a single flow at a bottleneck [1][3][4].

However, it is observed that BBR can lead to the problem of a massive amount of packet retransmission [1][5]. BBR's mechanism inherently leads to a sustained overload of the bottleneck, resulting in a steadily increasing amount of inflight data, queuing up at the bottleneck buffer and occurring excessive retransmission. Besides, BBR has no mechanism to drain this unintentionally built-up queue, except *ProbeRTT* which is triggered at a preset period [1][3]. On the other hand, loss-based congestion controls (e.g., Cubic [6], BIC [7]) are widely deployed in Internet service which are sensitive to packet retransmission. Cubic is very popular and configurated as the default congestion control in Linux. Nevertheless, with such considerable retransmission, the effect of the presence of BBR on loss-based schemes' performance remains unclear.

Hence, with extensive testbed experiments, we unveil that the performance of Cubic is degraded sharply when competed with BBR. Our experiments demonstrate: 1) **Overwhelming throughput**. BBR does maintain high throughput for data transfer, but it would suppress Cubic's performance and cause unfairness; 2) **Overmuch retransmission**. With the aggressive mechanism, BBR can occupy most of bottleneck's capacity and would eventually saturate the bottleneck and cause Cubic flow experiencing massive retransmission either.

Motivated by aforementioned issues, we seek for a solution that can preserve the good properties of BBR, while alleviate the degradation of performance on other coexisting congestion congestion schemes, e.g., Cubic. To this end, we present *Modest BBR*, a simple yet effective solution that achieves our goal. First, *Modest BBR* would response to retransmission in a moderate way, rather than too aggressive like BBR. It would adjust its pacing rate according to the network congestion. Then, at its heart, *Modest BBR* sacrifices a small amount of bandwidth compared to BBR and maintain a relatively high throughput.

The main contributions of this paper are as follows:

1) We designed *Modest BBR*, which is a variant of BBR with similarly high throughput, but more amicable by adjusting pacing rate based on both on packet retransmission and bottleneck capacity.
2) A prototype of *Modest BBR* is implemented based on Linux kernel 4.14. *Modest BBR* is light-weighted and effective, containing only about 100 lines of code, and pluggable into Linux kernel.
3) Extensive test-bed experiment results show that *Modest BBR* achieves comparable throughput compared to BBR, while improving overall throughput by 7.1% and offering better fairness when coexisting with loss-based mechanisms.

Corresponding author: Dr. Lin Cui

This work has been partially supported in part by Chinese National Research Fund (NSFC) No. 61772235 and 61402200; the Fundamental Research Funds for the Central Universities (21617409); the UK Engineering and Physical Sciences Research Council (EPSRC) grants EP/P004407/2 and EP/P004024/1.

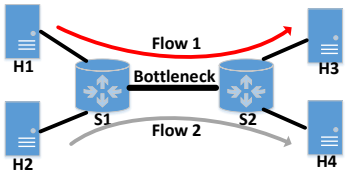
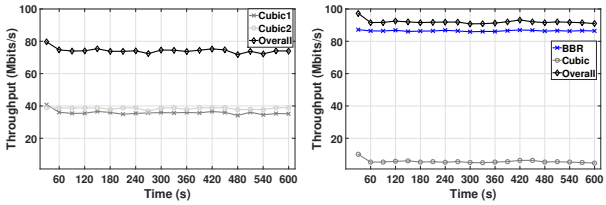


Fig. 1. Standard dumbbell topology with two senders and two receivers.



(a) Two Cubic flows (b) One Cubic and one BBR.

Fig. 2. The Performance of Cubic and BBR when they coexist

The remainder of this paper is organized as follows. We present our motivations for this research in Section II. We describe the design of *Modest BBR* in Section III, followed by presenting the evaluation *Modest BBR* in both testbed and simulation environments in Section IV. Related works are surveyed in Section V. Finally, we conclude the paper in Section VI.

II. Motivation

Many studies have shown BBR congestion control would introduce considerable packet retransmission [1][3]. On the other side, Cubic, the default congestion control in Linux kernel, is a loss sensitive scheme which would degrade its performance when encountering lots of retransmission. Thus, a key question arises: Does Cubic degrade its performance with the presence of BBR?

In order to quantify such impairment, we built a small testbed consisting of 4 Linux 4.14 servers connected to two routers, as shown in Figure 1, and generated flows using *iperf3* from H1 and H2 to H3 and H4 respectively.

A. Impact on throughput

First, we compared the throughput of one Cubic flow and one BBR flow (depicted in Figure 2(b)) and the throughput of two Cubic flows (depicted in Figure 2(a)). Besides, these two groups of flows would share the bottleneck capacity respectively. And we also measure the overall performance which is also shown in both figures respectively. From the Figure 2(b), it is clear that BBR outperforms Cubic. The throughput of BBR achieve almost 13X compared to Cubic's performance (around 88.1Mbits/s to around 6.3 Mbits/s). BBR's aggressiveness help it obtain most of the bottleneck capacity. Meanwhile, without the presence of BBR flow, both two Cubic flows can get about 36 Mbits/s throughput shown in Figure 2(a), which compete for bottleneck capacity. By comparing these two figures, we can conclude that Cubic is affected by BBR heavily (around 36 Mbits/s to 6.3Mbits/s). Moreover, the average throughput of each flows above which we repeated this experiments ten times is depicted in Figure 3. Clearly, the presence of BBR lead to worse Cubic's performance. To be

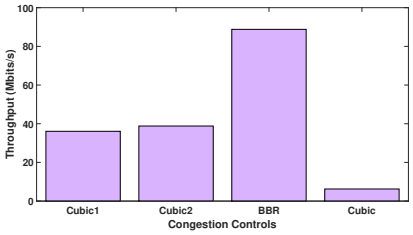


Fig. 3. The average throughput of each flow.

more specific, the overall throughput for the group of BBR flow and Cubic flow is up to 91.7 Mbits/s, while the total throughput of two Cubic flows is up to 74.1 Mbits/s.

Next, we trigger one cubic flow and one BBR flow which start at 0s and 300s respectively and last for 600 seconds. As we can see from Figure 4 that, in the first 300 seconds, Cubic can obtain quite high throughput while get relatively low throughput in later 300 seconds when BBR flow involved (from about 90Mbits/s to around 6.2Mbits/s). Hence, the degradation of Cubic's performance is demonstrated that is caused by the presence of BBR.

Observation 1. *The throughput of Cubic is overwhelmed by BBR.*

B. Impact on retransmission

Thus, as Cubic is a loss-based scheme, a simple idea arises in our mind: does BBR introduces massive retransmission so that degrade Cubic's performance? In order to testify our guess and quantify the cause of such phenomenon, we measured the number of retransmission of each flows with repeating triggering both one Cubic flow and one BBR flow and two Cubic flows respectively.

Figure 5 and Figure 6 show the results. From Figure 5, we can observe that the retransmission of BBR flow arrives at 3X compared to Cubic flow's (around 1600 per 30 seconds to around 600 per 30 seconds) in whole experimental period. Meanwhile, when only Cubic flows exist, the numbers of retransmission of both flows are just around 250 every second. Besides the average of ten runs is depicted in Figure 6. Clearly, retransmission of Cubic flows with the presence of BBR flows is larger than the opposite (1600 per 30 seconds to 615 per 30 seconds).

Thus, since BBR's pacing rate would not be affected by retransmission signal, their aggressiveness would help it obtain higher throughput. However, the large amount of retransmission which BBR creates and occupying most of the bottleneck bandwidth, thereby does affect Cubic flows which treats them as congestion signal. Therefore, overmuch retransmission introduced by BBR flow is the main cause of degradation of Cubic's performance.

Observation 2. *Cubic causes massive retransmission because of BBR's aggressive meachanism.*

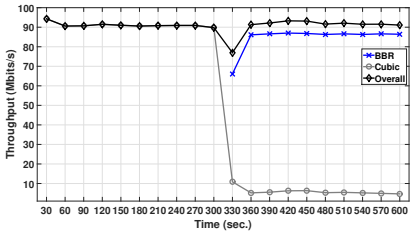
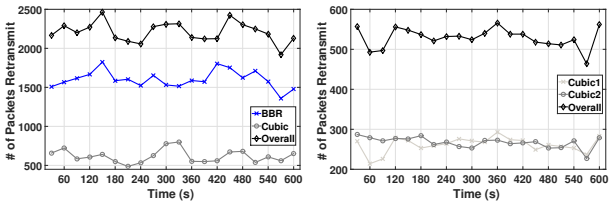


Fig. 4. Cubic starts at the beginning and BBR starts at 300s.



(a) One Cubic flow and one BBR flow. (b) Two Cubic flows

Fig. 5. The packet retransmission of Cubic and BBR when they coexist

III. *Modest BBR* DESIGN

A. Design Goals

Motivated by the above problems, we aim to design a moderately aggressive, BBR based scheme for alleviating the degradation of Cubic when coexisting with Cubic with the following properties:

- **Less aggressive:** Our scheme should alleviate the degradation of Cubic's performance when they coexist. In the other word, we need a BBR which is sensitive to loss either and reduces several orders of magnitude of retransmission.
- **High throughput:** Our scheme should maintain a relatively high throughput as BBR does, though it can sacrifice a small amount of throughput. Meanwhile, it should be able to fully utilize the network capacity as highly as possible.
- **Readily deploy:** Our scheme should be compatible to current OS kernel as well as deployed into a commodity web server easily.

B. Modest BBR Mechanism

After understanding the existing BBR-Cubic coexistence problem, we revisit the design of BBR congestion control that it is too aggressive when the bottleneck is fully utilized, which causes the inter-protocol unfairness. We note that too much retransmission is the core issue. Hence, merging loss-sensitive approach into BBR congestion logic is a plausible solution. Here, we propose a Moderately aggressive BBR congestion control, *Modest BBR*, for alleviating retransmission and better inter-protocol fairness. The core idea of our *Modest BBR* is sacrificing a small amount of bandwidth for alleviating retransmission and moderate aggressiveness, in order to obtain better fairness with loss-sensitive approaches (e.g., Cubic). In other word, the core idea can be described as equation 1. In BBR's implementation, the size of inflight packets is equal to $BtlBw \times RTprop$. Thus, we decrease this pacing rate to $BtlBw \times RTprop \times Proportion$ which *Proportion* is a decimal

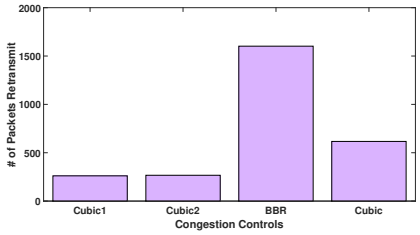


Fig. 6. The average number of retransmission of each flow.

between 0 to 1. By using such decreasing approach the pacing rate can be less aggressive.

$$Pacing\ rate = BtlBw \times RTprop \times Proportion \quad (1)$$

Next, we would detail how to compute a suitable *Proportion*. Here, similar to BIC [7], binary search phase and linear increase phase are adopted. At binary search phase, we view computing *Proportion* as a searching problem in which network gives yes/no feedback through retransmission as to whether the current pacing rate (or window) is too aggressive. The starting points for this search are based on W_{min} and W_{max}[1]. Specifically, W_{max} represents the upper bound of *Proportion* while W_{min} is the lower bound. Later, this phase repeatedly computes the midpoint between W_{max} and W_{min}, sets the current window size to the midpoint and checks for feedback. Based on this feedback, the midpoint is taken as the new W_{max} if retransmission occurs and as the new W_{min} if not. The process repeats until the difference between W_{min} and W_{max} falls below a preset threshold, called the minimum increment (S_{min}).

At linear increase phase, *Modest BBR* would increase the pacing rate (or window) in a linearly additive strategy. If no retransmission occurs, window would be increased linearly in step S_{min}. Otherwise, *Modest BBR* would fall back to binary search phase.

We present the pseudo-code of *Modest BBR* in Alg. 1 which is implemented as a modification of BBR congestion control and the parameters in Table I are used in the *Modest BBR* congestion control. Especially, *Modest BBR* only works on the steady phase of BBR since BBR flow spends the vast majority of its time and modifying window size in other phases would impact on bottleneck capacity measurement. Furthermore, once entering the *Modest BBR* mechanism, the current window size would be stored in $last\_cwnd$ (in Alg. 1 line 3). While switching to other phases except steady phase, the window size would be restored to $last\_cwnd$.

C. Parameters selection

First, we discuss some parameters' value setting. The first one is W_{max}. If setting W_{max} larger than 1, which means the outcome congestion window would be larger than the BBR's, would risk congestion leat to "bufferbloat" [8]. Otherwise,

[1]Although, Alg. 1 do the binary search based on min_win and max_win rather than W_{min} and W_{max}, the purpose of Alg. 1 is to find a appropriate *proportion* either

TABLE I
Modest BBR CONGESTION CONTROL PARAMETERS

Parameter	Description
W_{min}	Minimum window scale factor
W_{max}	Maximum window scale factor
max_win	Upper bound of window size in binary search phase
min_win	Lower bound of window size in binary search phase
bin_p	Indicating in binary search phase
add_p	Indicating in linear increase phase
S_{min}	the minimum increment step

Algorithm 1 *Modest BBR* Congestion Control

1: **if** $retransmission\ occurs$ *and* $bin\_p = 0$ **then**
2: $bin\_p = 1, add\_p = 0$
3: $last\_cwnd = cwnd$
4: $min\_win = cwnd * w_{min}$
5: $max\_win = (cwnd + min\_win)/2$
6: $cwnd = max\_win$
7: **else if** $bin\_p = 0$ **then**
8: **if** $retransmission\ occurs$ **then**
9: $max\_win = (max\_win + min\_win)/2$
10: $cwnd = max\_win$
11: **else**
12: $min\_win = (max\_win + min\_win)/2$
13: $cwnd = min\_win$
14: **end if**
15: **if** $(max\_win - min\_win) <= S_{min}$ **then**
16: $add\_p = 1$
17: $bin\_p = 0$
18: **end if**
19: **end if**
20: **if** $add\_p = 1$ *and* $cwnd <= last\_cwnd$ **then**
21: **for** $each\ ACK$ **do**
22: $cwnd = cwnd + S_{min}$
23: **end for**
24: **end if**

if setting W_{max} smaller than 1 may not approximate the optimal utilization. Next, we discuss the value of W_{min}. W_{min} determines the lower bound of binary search phase, thus, finding an appropriate value is quite important for alleviating retransmission. If W_{min} is set to a very low value, the binary search phase would be prolonged and affect the convergence. Meanwhile, if W_{min} is set to a relatively high value, it may not be a retransmission free value which is useless for alleviating retransmission. Besides, with this W_{min}, *Modest BBR* would oscillate between binary search phase and linear increase phase. Hence W_{min}'s setting is relatively important. Furthermore, S_{min} relates to the aggressiveness in linear increase phase. If S_{min} is set to a very low value, the procedure of approximating the actual network capacity would be prolonged and affect the performance. Meanwhile, if S_{min} is set to a relatively high value, it may fill the bottleneck buffer with the excess data which would result in packet retransmission. Thus S_{min} is vital for interacting with changing network condition.

In current *Modest BBR*'s implementation, W_{max} is set to 1 since BBR is good at measuring bottleneck capacity. Besides, line 20 in Alg. 1 restrains the pacing rate cannot be larger than BBR estimated. W_{min} is set to be 0.75 which is the value of the cwnd_gain in drain phase of BBR. This is a heuristic setting. And S_{min} is set to be 1, which is a conservative value. We want *Modest BBR* would reduce retransmission effectively. Besides, We leave optimal parameters tuning as an important future work.

D. Convergence and intra-protocol fairness

Then we briefly discuss the intra-protocol fairness of *Modest BBR*. Let's consider *n Modest BBR* flows, which share the same bottleneck. As the original BBR paper [3] demonstrated that BBR flows can learn their bottleneck fair share, these *n* flows can converge to fair share before entering binary search phase and linear increase phase. Next, once the bottleneck is overloaded, the congestion information would be transmitted to each sender through occurring retransmission. Since all flows encounter the same shortage of bottleneck capacity, these *n* flows would receive the same amount of retransmission signal. Thus, these *n* flows may experience similar both binary search phase and linear increase phase processing. Therefore, these *n* would get similar congestion control value or converge to the fair share.

IV. EVALUATION

A. Experimental Setup

Most experiments are conducted on a physical testbed with 4 HP Z230 workstations (4-core Intel i7-4790 3.6GHz CPU and 16 GB memory as well as 100M NIC). Our switches are Huawei HG255d, each with a buffer of 32MB shared by four 100M ports. Besides, 5 client machines are adopted for evaluating the performance in real network. These VMs located in Singapore (Sin), Shenzhen (SZ), Beijing (BJ), London (Lon) and New York (NY). The Mininet simulation is conducted on one of the workstations, which runs Linux Kernel 4.14. Furthermore, Apache2 is deployed on this workstation for offering web service.

We run Linux kernel 4.14 which implements BBR and Cubic as a pluggable module. We set *tcp_sack* and *tcp_low_latency* to 1 and increase the maximum receive and send window sizes of TCP up to 16MB, to support experiments for getting rid of buffer insufficiency. Results are obtained with MTU sizes of 1.5KB, as current desktop networks typically use this setting. Moreover, we enabled the queuing discipline "fq" at senders which BBR implements its packet pacing feature based on it. Besides, "fq" was not enabled for flows of Cubic.

To understand *Modest BBR* performance, three different congestion control configuration are considered: Cubic, BBR and *Modest BBR*. Meanwhile, the overall performance of one *Modest BBR* flow and one Cubic flow would be compared to the two group flows evaluated in section II. The overall performance of two Cubic flows is denoted as "Overall*" and "Overall+" represents the overall performance of one BBR flow and one Cubic flow.

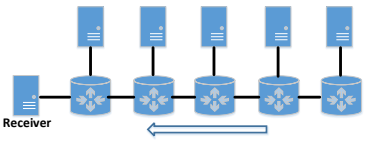
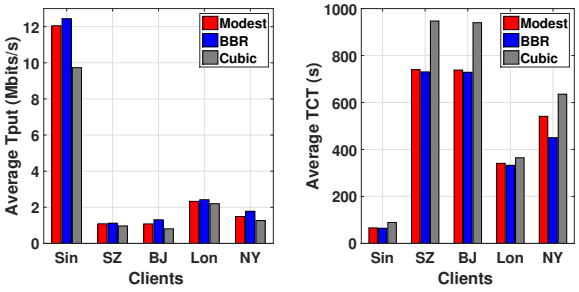
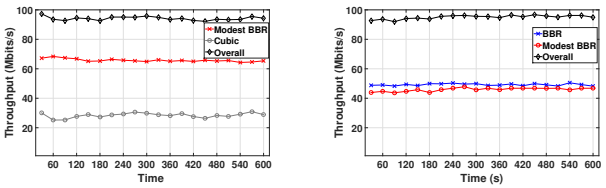


Fig. 7. Multi-hop, multi-bottleneck (parking lot) topology.



(a) Average throughput. (b) Average TCT

Fig. 8. The Performance of *Modest BBR*, BBR and Cubic



(a) One *Modest BBR* and one Cubic. (b) One *Modest BBR* and one BBR.

Fig. 9. The Performance of *Modest BBR*, Cubic and BBR when they coexist

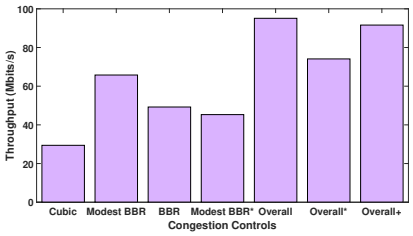


Fig. 10. The average throughput of each flow.

The main metrics used are: Transfer completion time (TCT, measured by *CURL*), Throughput (measured by *CURL* or *iperf*) and the number of retransmission (measured by *CURL*). To be more specific, TCT is measured for 100 MB data transfer. Fairness to inter-protocol and convergence experiments are conducted on topologies shown in Figure 1 (physical testbed experiment) and Figure 7 (Mininet [9] based simulation), respectively.

B. Performance in real network

We first evaluate *Modest BBR*'s performance in Internet circumstance. We deployed one Apache2 web server in our laboratory (Guangzhou, China) and instructed all clients to pull 100 MB data from web server simultaneously. Figure 8 shows the average throughput and TCT of these three congestion controls and the results presented are averaged over 10 runs of each congestion control. From the figure, we can tell that *Modest BBR* and BBR outperform Cubic. BBR gets the best performance and Cubic obtain the worst among three congestion controls. In addition, *Modest BBR* and BBR can achieve up to 23.8% and 27.8% higher throughput compared to Cubic, respectively. Furthermore, *Modest BBR* can reduce the average TCT by up to 25.8% compared to Cubic, while BBR can reduce about 27.4% average TCT.

Both *Modest BBR* and BBR function under the mechanism which continually measures bottleneck capacity. Hence they can achieve good TCT as well as high throughput in all clients' network environments. Besides, the performance gap between *Modest BBR* and BBR is relatively small which *Modest BBR* just get 3% lower throughput and 7% larger TCT compared to BBR. In other word, the amount of performance *Modest BBR* sacrifices is relatively small and acceptable.

C. Fairness to inter-protocol

The inter-protocol fairness in sharing the bottleneck bandwidth between two competing flows are measured with the similar setting. Resutls are shown in Figure 9 $\sim$ Figure 12.

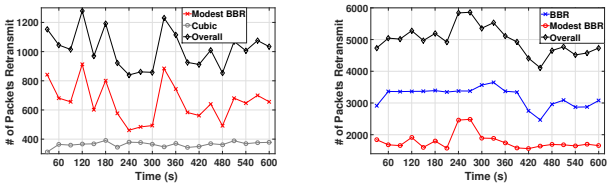
As Figure 9(a) shows, Cubic can increase almost 4X throughput (around 39.5 Mbits/s to 6.3 Mbits/s) when coexists with *Modest BBR* compared to Figure 2(b) which Cubic coexists with BBR. Meanwhile, *Modest BBR* achieves 3X throughput compared to Cubic. Figure 10 shows the average throughput of each flow and the overall of different flow group. The overall throughput of this group, which achieves 3.8% higher than sum of Cubic flow and BBR flow (Overall+) and 28.3% higher than the total of two Cubic flows (Overall*).

Meanwhile, less retransmissions occur for both Cubic and the overall, with the presence of *Modest BBR*. Figure 11(a) shows that *Modest BBR* would get about 2X to 3X packet retransmission compared to Cubic. Besides, the Cubic flow coexisting with *Modest BBR* experience 0.5X to 1X less retransmission, compared to the Cubic flow coexisting with BBR. Furthermore, Figure 12 shows the average number of retransmission of each flow and the total of each flow group. The Overall achieves 120% less retransmission compared to Overall+ while only 39% more retransmission compared to Overall*. Thus, we can conclude that *Modest BBR* is more inter-protocol friendly to Cubic than BBR. And Cubic is a loss sensitive congestion control which less retransmission would improve its performance.

On the other hand, the flow group of one *Modest BBR* and one BBR is adopted to testify the fairness between *Modest BBR* to BBR. The expected result should be that *Modest BBR* gets comparable performance compared to BBR. Figure 9(b) and Figure 11(b) show that *Modest BBR* is a competitive congestion control. *Modest BBR* gets only about 7% lower throughput compared to BBR while can reduce almost 50% retransmission. This demonstrates that the window adjustment responding to retransmission is yet simple and effective mechanism to moderate aggressiveness.

D. Convergence and intra-protocol fairness

Then we demonstrate that *Modest BBR* flows can converge to their fair shares. Similar to the experiments done in [3] and [10], we performed the convergence and fairness experiments for BBR and *Modest BBR* by letting five flows to compete for the bottleneck shares. We added a new flow every 3 seconds on a parking lot topology (starting points:0s, 3s, 6s, 9s, 12s) and



(a) One Cubic and one *Modest BBR*. (b) One BBR and one *Modest BBR*

Fig. 11. The packet retransmission of Cubic, BBR and *Modest BBR* when they coexist.

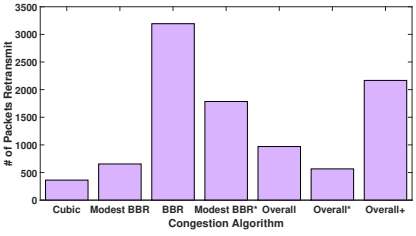


Fig. 12. The average number of retransmissions of each flow.

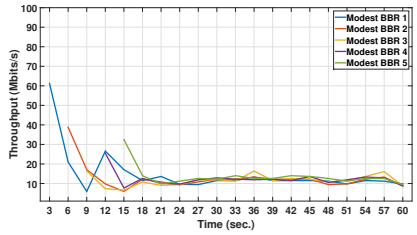
continued the experiment to 60 seconds. Throughput data are collected every 3 seconds. Figures 13(b) and 13(a) show BBR and *Modest BBR* performance, respectively. As the figures show, *Modest BBR* tracks BBR's convergence and fairness which both converge to fair share throughput around 22 second. Thus, the *Modest BBR*'s congestion control mechanism is demonstrated that has little impact on convergence and intra-protocol fairness. In other word, the learning fair share mechanism embedded inside BBR, which *Modest BBR* has inherited, is workable and effective.

V. Related Works

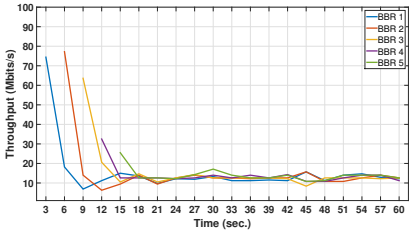
Since the introduction of TCP, a large amount of congestion controls have been developed. Reno [11] is nowadays considered the standard TCP which basically implements the four classical congestion control mechanisms of TCP (i.e., Slow Start, Congestion Avoidance, Fast Retransmission and Fast Recovery). BIC [7] uses two phases to update the bandwidth; linear increase to approach a fair window size, and binary search to improve RTT fairness. Linear increase is similar to additive increase, while binary search essentially uses two window sizes (Wmax and Wmin) that updates these windows and the actual window size to approximate the optimal window size. Once Wmax and Wmin are converging, BIC falls back to linear increase. Cubic [6] is an improvement of BIC, which aims to compensate the aggressive behavior of BIC to more reasonable levels, and simplifies the algorithm. The window growth function of Cubic is a Cubic function, whose shape is very similar to the growth function of BIC. BBR, which is proposed in [3], shows that it can get better performance than current other TCP congestion control algorithms and especially in WAN environment. BBR measures the end-to-end RTT and its bottleneck bandwidth and uses this measurement result to calculate an appropriate congestion window to fully utilize the bottleneck capability.

VI. Conclusion

This paper presents *Modest BBR*, a new congestion control for achieving high throughput and better inter-protocol fairness. It primarily aims at alleviating the packet retransmission and adjusting its pacing rate according to network condition. Our extensive experiments demonstrated that *Modest BBR* can maintain high utilization while affect less on loss-based proposals (e.g., Cubic).



(a) Five *Modest BBR* flows.



(b) Five BBR flows

Fig. 13. The intra-protocol fairness of *Modest BBR* and BBR.

References

[1] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of BBR congestion control," in *Network Protocols (ICNP), 2017 IEEE 25th International Conference on*. IEEE, 2017, pp. 1–10.

[2] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-host congestion control for TCP," *IEEE Communications surveys & tutorials*, vol. 12, no. 3, pp. 304–342, 2010.

[3] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Queue*, vol. 14, no. 5, p. 50, 2016.

[4] Y. Zhang, L. Cui, F. P. Tso, Q. Guan, and W. Jia, "TCon: A Transparent Congestion Control Deployment Platform for Optimizing WAN Transfers," in *IFIP International Conference on Network and Parallel Computing*. Springer, 2017, pp. 49–61.

[5] N. S. Rao, Q. Liu, S. Sen, J. Hanley, I. Foster, R. Kettimuthu, C. Q. Wu, D. Yun, D. Towsley, and G. Vardoyan, "Experiments and Analyses of Data Transfers over Wide-Area Dedicated Connections," in *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*. IEEE, 2017, pp. 1–9.

[6] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[7] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4. IEEE, 2004, pp. 2514–2524.

[8] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the internet," *Queue*, vol. 9, no. 11, p. 40, 2011.

[9] Handigol, Nikhil and Heller, Brandon and Jeyakumar, Vimalkumar and Lantz, Bob and McKeown, Nick, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '12. New York, NY, USA: ACM, 2012, pp. 253–264.

[10] K. He, E. Rozner, K. Agarwal, Y. J. Gu, W. Felter, J. Carter, and A. Akella, "AC/DC TCP: Virtual congestion control enforcement for datacenter networks," in *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 2016, pp. 244–257.

[11] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," *Expires*, vol. 345, no. 2, pp. 414–418, 2004.