Poster Abstract: Analysis and Experimental Investigation of BBR

Tongyu Dai[1], Xinggong Zhang[1,2,§] and Zongming Guo[1,2]
[1]Institute of Computer Science & Technology, Peking University, Beijing, P.R. China, 100871
[2]Cooperative Medianet Innovation Center, Shanghai, China
{dai_tongyu, zhangxg, guozongming}@pku.edu.cn

I. Introduction

With the advance of mobile network and video technology, increasing video applications are emerging with the requirements of low latency and high bitrate. However, nowadays Internet suffers from high delay or low bandwidth utilization, leading congestion control to be a hot topic again.

Conventional TCP is the primary cause of these issues, such as Cubic [1] and Compound [2]. It interprets packet loss as a congestion signal, which has been out-of-date now [3], resulting in "bufferbloat" [4] and unacceptable packet delay. Besides, in wireless networks, low channel utilization also results from the stochastic loss unrelated to congestion [5]. There are also some delay-based algorithms such as TCP Vegas [6], TCP Fast [7], LEDBAT [8] and so on. They all perform well in constraining packet delay, but will be starved when sharing a bottleneck link with loss-based flows [9]. Therefore, it is still a challenge to design an effective congestion control algorithm to achieving low latency, high channel utilization and good fairness at the same time.

For addressing these issues, Google proposed a new algorithm named BBR [3] in 2016, which has been integrated in Linux core 4.9. The key idea of BBR is to make the number of inflight packets (i.e. data sent but not yet acknowledged) converge to bandwidth-delay product (BDP). To this purpose, BBR estimates bottleneck bandwidth and round-trip propagation time to calculate BDP to adjust sending rate. BBR claims it achieves high channel utilization, low latency, good intra-fairness and a fair share with loss-based flows. Although it shows some experiment results in [3] to prove its performance, there are still many issues worth exploring. How much better is BBR than other algorithms? To what extent BBR is able to fairly share the bottleneck with concurrent flows?

In this paper, we gave a further insight on BBR through analysis and extensive experiments, including emulation and real-world tests over Ethernet and 4G. Through analysis and comparison with Cubic and Vegas, we found that 1) BBR adopts a syntonization mechanism to achieve intra-protocol fairness, but does not achieve a fair share of bandwidth with loss-based flows, which is contrary to what it claims. It starves Cubic in the case of small network queue, and is starved by Cubic when queue size is large. 2) Over real-world Ethernet and 4G networks, BBR does not perform as well as expected, even worse than Vegas. To the best of our knowledge, this is the first experimental investigation of BBR.

In summary, the contributions of this paper can be summarized in two-folds: First, we reveal the flaws of BBR in TCP-fairness, which is not achieved as it claimed. Last, we carried out extensive experiments to show that BBR performs not well as expected in complex real network.

II. Analysis of BBR

BBR claims that it *"converges toward a fair share of the bottleneck bandwidth whether competing with other BBR flows or with loss-based congestion control"* [3]. It really achieves intra-protocol fairness through a syntonization mechanism. However, we found that the TCP-fairness of BBR is only achieved under specified conditions. The queue size of the bottleneck link plays an important role.

When the queue size is small, TCP is starved by BBR. The small queue size limits TCP to accumulate packets so that packet loss occurs quickly. Once TCP halves its congestion window in front of packet loss, BBR grabs remainded bandwidth to starve it. In contrast, when the queue size is large, TCP starves BBR. TCP is always trying to fulfill network queue until packet loss occurs. Even it halves its window size sometimes, the number of inflight packets are still much larger than BDP, resulting in starving BBR.

Due to this weakness, BBR does not perform well in real environment, because there are a variety of background traffic including lots of TCP flows to compete with it. Especially in the environment with large buffer size like wireless networks, the aggressive TCP flows will starve BBR.

III. Experimental Result

A. Testbed and Scenario

Emulation testbed: To evaluate the performance of BBR in the emulation testbed, we have established two nodes, where Node 1 plays as a sender and Node 2 is a receiver, connected through Gigabit fiber. Each node runs a configurable number of BBR and Cubic sources with *Linux core 4.9*. We employed the *NetEm* linux module along with the traffic shaper *tc* to configure the link parameters, such as bottleneck bandwidth, propagation delay, packet loss and maximum queue size.

§Corresponding author.

This work was supported by National Natural Science Foundation of China under contract No. 61471009 and Beijing Culture Development Funding under Grant No.2016-288.

978-1-5386-5979-3/18/$31.00 ©2018 IEEE

Real-world test: In the case of Ethernet, *Planetlab* is employed to establish nodes in five countries respectively, including America, China, Japan, New Zealand and Switzerland. Our host is used as a receiver while these nodes are senders. In 4G mobile network, we drove on the Fourth Ring Road of Beijing and employed a laptop connected with a 4G mobile to investigate BBR, Cubic and Vegas.

B. Inter-protocol fairness

We used above emulation testbed to investigate TCP-fairness of BBR, with one BBR flow against one standard TCP flow (Cubic) over a bottleneck. According to the analysis in Section II, we considered a bottleneck with five different queue sizes, i.e. $q_i \in [25, 50, 100, 300, 1000]ms$, and a constant capacity equal to $2000kbps$.

Figure 1 shows the overall channel utilization of BBR and Cubic. Five groups of bars are shown, each group for a different value of queue size. It clearly shows that BBR grabs a higher bandwidth at small queue size. A fair share is only achieved in the case of queue size equal to 300ms. This also verifies our previous analysis.

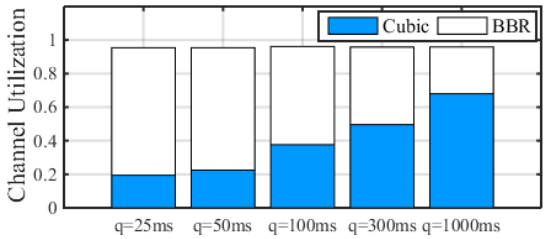

Fig. 1: Channel Utilization when one BBR flow shares the bottleneck with a Cubic flow.

C. Global Ethernet experiments

Employing Planetlab, we investigate the performance of BBR on Ethernet. For each node, we run ten tests to eliminate the impact of exceptions and every test lasts 300 seconds.

TABLE I: Average throughput [Kbps] and RTT [ms] to America, China, Japan, New Zealand and Switzerland.

	BBR		Cubic		Vegas	
Country	Rate	RTT	Rate	RTT	Rate	RTT
USA	1381	270	1557	260	662	314
CHN	6040	88	6344	68	4074	167
JPN	2599	137	2997	168	2602	150
NZL	802	342	1095	365	660	449
CHE	901	321	1097	316	848	315

Table I shows the average throughput and RTT of BBR, Cubic and Vegas to the five countries. As for throughput, there is a common regularity for all tests, i.e. $Cubic > BBR > Vegas$. Besides, BBR lost the ability to constrain delay and its RTT is close to that of Cubic, mainly due to the competition with TCP in real network. The bufferbloat caused by TCP results in the high delay. So overall, on the real-world Ethernet network, BBR performs worse than Cubic.

D. 4G mobile network

Deploying the scheme in Section III-A, we investigated the performance of BBR, Cubic and Vegas in 4G network. For each algorithm, we have run the test on the same route and started at similar time, with every test lasting 10 minutes.

TABLE II: Average throughput [bps] and RTT [ms] in 4G mobile network.

	Ave. Throughput	Ave. RTT
BBR	3256	128
Cubic	2997	129
Vegas	4285	94

Table II displays the average throughput and RTT, illustrating that BBR performs similarly to Cubic, in terms of both throughput and delay. However, to our surprise, Vegas performs best in 4G mobile network, obtaining 1.3× throughput vs. BBR and Cubic. It is mainly because the defect of BBR in TCP fairness limits its performance, while the random loss in 4G mobile network reduces the throughput of TCP.

IV. Conclusions

In this paper, we first analyzed the BBR algorithm proposed by Google and extensive experiments are carried out, including emulations and real-world tests over Ethernet and 4G networks, to evaluate its performance. The experimental results show that: (1) BBR does not achieve the professed fairness when competing with loss-based flows. BBR starves TCP in the case of small queue size, but is starved when queue size is large. (2) Over real-world Ethernet and 4G mobile networks, BBR does not perform well as expected.

References

[1] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.
[2] K. Tan, "Compound tcp : A scalable and tcp-friendly congestion control for high-speed networks," *Pfldnet Feb*, 2006.
[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] J. Gettys and K. Nichols, "Bufferbloat: dark buffers in the internet," *Communications of The ACM*, vol. 55, no. 1, pp. 57–65, 2012.
[5] G. Xylomenos, G. C. Polyzos, P. Mahonen, and M. Saaranen, "Tcp performance issues over wireless links," *IEEE Communications Magazine*, vol. 39, no. 4, pp. 52–58, 2001.
[6] L. SBrakmo and L. LPeterson, "Tcp vegas: end to end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, 1995.
[7] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "Fast tcp: motivation, architecture, algorithms, performance," *IEEEACM Transactions on Networking*, vol. 14, no. 6, pp. 1246–1259, 2006.
[8] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind. (2013) Low extra delay background transport (LEDBAT). Internet Draft draft-ietfledbat-congestion. [Online]. Available: http://tools.ietf.org/html/draft-ietf-ledbat-congestion
[9] L. A. Grieco and S. Mascolo, "Performance evaluation and comparison of westwood+, new reno, and vegas tcp congestion control," *Computer Communication Review*, vol. 34, no. 2, pp. 25–38, 2004.