Enhanced BBR Congestion Control Algorithm for Improving RTT Fairness

Geon-Hwan Kim, Yeong-Jun Song, Imtiaz Mahmud and You-Ze Cho*
School of Electronics Engineering, Kyungpook National University, Daegu, Korea
*Corresponding Author
kgh76@ee.knu.ac.kr, syj5385@knu.ac.kr, imtiaz.tee@gmail.com, yzcho@knu.ac.kr

***Abstract*—**Google has proposed the Bottleneck Bandwidth and Round-trip propagation time (BBR) as a new congestion control algorithm. The difference between it and the existing congestion control algorithm is that it is derived from actual congestions. BBR measures the bottleneck bandwidth and round-trip propagation time sequentially and sends data based on bandwidth delay product (BDP). The mere dependence on the BDP unfairly allocates a larger part of the bandwidth to a flow with long RTT than a short RTT flow. This nature of the BBR entirely conflicts with the trend of traditional TCP where a short RTT flow is more vibrant than long RTT flow. To solve this problem, we previously proposed the bottleneck queue buildup suppression method and verified that through NS-3. However, during the testbed experiments, we found that the previous method was unstable and led to unexpected results. Therefore, in this paper, we propose an improved version of the previously proposed algorithm. We confirmed through extensive testbed experiments that our proposed BBR-E improves fairness between the flows with different RTTs and ensures better throughput.

Keywords— BBR; congestion control; round-trip time; fairness; testbed;

I. INTRODUCTION

Recently, Google had proposed the BBR (Bottleneck Bandwidth and Round-trip propagation time) congestion control algorithm [1]. BBR is called a congestion-based congestion control algorithm because it is derived from actual congestions of the bottleneck link, unlike the existing loss-based or delay-based congestion algorithms. BBR aims to minimize the creation of queues in the bottleneck by measuring the maximum bandwidth and minimum RTT. It is designed to operate at Kleinrock's optimal operating point with minimum delay at maximum bandwidth [2].

Since BBR is still going through immense improvement, a lot of issues have been reported about the initially proposed BBR version 1[3]. One of the significant issues is unfair competition between short RTT and long RTT BBR flows. Long RTT flows tend to excessively overwhelm the short RTT flows by occupying more bandwidth because of their extended time delay. Some authors claim that the persistent queue created in the bottleneck is the main reason behind the fairness issue [4].

In this paper, we propose Bottleneck Bandwidth and Round-trip propagation time - enhanced **(**BBR-E**),** to improve the fairness between different RTT flows of the BBR. The proposed BBR-E prevents excessive data transmission by the long RTT flows and achieves an overall balance. We also confirmed that it facilitates greater throughput than the original BBR. The performance comparison was executed through actual testbed experiments.

II. STAGE OF BBR

The operation of the BBR can be roughly classified into the following two states [1]:

A. Startup State

BBR uses a *pacing_gain* of about 2.88 to fill the bandwidth rapidly and to find the bottleneck bandwidth at the beginning of the start of a flow. If the growth of probing bandwidth decreases, BBR exits the Startup phase and enters the Drain phase. The *pacing_gain* is set to approximately 0.34 to eliminate queues that were created due to excessively transmitted data during the Startup phase. The probe bandwidth (ProbeBW) phase begins and the Drain phase terminates when the amount of inflight data becomes one BDP (bandwidth-delay product).

B. Steady State

In this state, BBR spends most of its time in the ProbeBW phase. In ProbeBW, it cruises with an eight RTT cycle while pacing with specific values during each RTT. In the first RTT, it probes for bandwidth with a pacing rate of 1.25. In the next, it decreases the pacing rate to 0.75 to erase the sent extra data during the previous RTT. For the next six RTTs, it maintains a *pacing_gain* of 1.0 to maintain the measured maximum bandwidth.

In the steady state, BBR also has a ProbeRTT phase for measuring the minimum propagation time (RTprop). This phase is utilized if RTprop is not updated to a lower value during the last 10 seconds. By default, BBR enters into ProbeRTT every 10 seconds, and during this phase, only 4 packets are transmitted. This greatly reduces the flow of data and enables the measurement of the RTprop of the link.

III. ENHANCED BBR ALGORITHM

There are two main reasons behind the RTT fairness issues of the BBR flows [4].

(1) The behavior of probing more bandwidth creates long unexpected queues.

(2) Owing to the short RTT, BBR flow has lower BDP values than the long RTT flow, it has a smaller amount of inflight data.

To eliminate the bias towards the long RTT flows caused by its occupation of more buffers, previously, we proposed queue

buildup suppression method for both flows [5]. Excessive data transmission was reduced, and the creation of queues was minimized by limiting the *cwnd_gain* to less than 2 BDP (default) in the ProbeBW phase. Assuming that the measured RTT exceeds $\alpha \times \text{RTprop}$, we considered that the pipe was already full and that a queue was growing, therefore, the *cwnd_gain* was limited to one BDP to release that queue. Algorithm 1 describes the previously proposed method.

Algorithm 1. Queue buildup suppression method

```
if (lastRTT < α × propRTT)
    cwnd_gain = 2;
else
    cwnd_gain = 1;
```

Queue buildup suppression method worked well in network simulator three (NS-3), but the results of testbed experiments were different. It led to some performance improvements, but occasionally showed a sharp increase/decrease in the throughput. Therefore, we modified Algorithm 1 to stabilize the behavior of the BBR flow in the testbed experiments.

Algorithm 2. BBR-E

```
if ( lastRTT ≤ propRTT )
    cwnd_gain = 2.0;
else if ( propRTT < lastRTT < α × propRTT )
    cwnd_gain = 1.5;
else
    cwnd_gain = 1.0;
```

In the testbed experiments of the queue buildup suppression method, we observed that during the steady state, after exiting ProbeRTT, a long RTT flow often overwhelms the short RTT flow during the following 10 seconds before entering the next ProbeRTT. Therefore, we reduced the cap of *cwnd_gain* from 2.0 to 1.5. Also, we set the *cwnd_gain* to 2.0 only when the measured RTT is lower than the RTprop. With this change, two BBR flows that had large RTT differences also showed reasonable fairness in the testbed experiment.

IV. EVALUATION

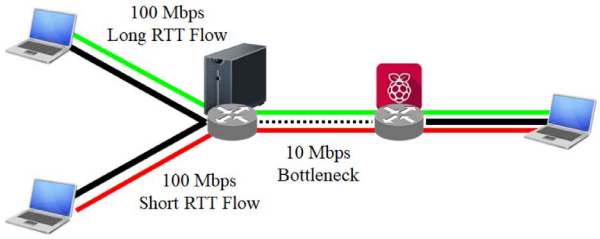

Fig. 1. Testbed setup.

The testbed experiment environment for evaluating the fairness in the situation where two BBR flows share a common bottleneck link is constructed as shown in Fig. 1. All the end hosts of the testbed used the Linux kernel version 4.14, BBR v1.0. The Linux-based PCs and Raspberry pi acted as the intermediate router for implementing a bottleneck link. At the two sender hosts, we changed the queuing discipline to "fq", to enable pacing. We configured various RTTs through "qdisc" and different bandwidths of bottleneck links through "ethtool". Each sender transmitted data to the receiver using "iperf3". To consider a large bottleneck buffer situation, we set the bottleneck's buffer size as 1.5 MB, which was approximately 1.2 s queuing delay at 10 Mbps. It was enough for the two flows to work competently.

A. Original BBR – 10 ms vs 10 ms & 10 ms vs 100 ms

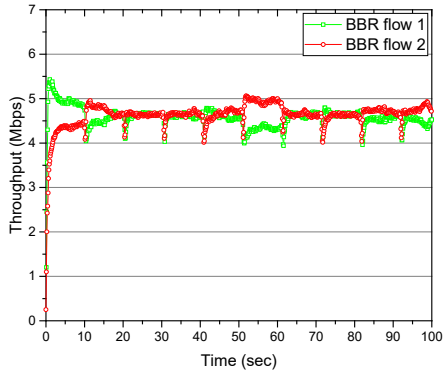

Fig. 2. A throughput comparison – 10 ms vs 10 ms BBR flows.

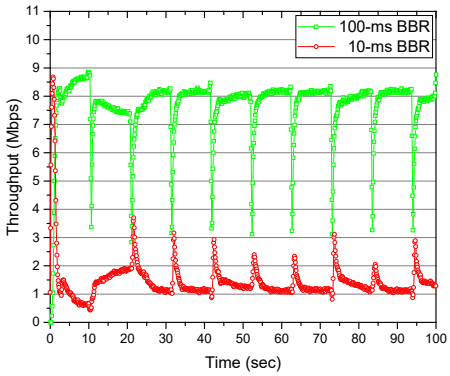

Fig. 3. A throughput comparison – 10 ms vs 100 ms BBR flows.

Fig. 2 shows the results of original BBR. Two BBR flows with a 10 ms RTT shared the bandwidth fairly. However, when competing with 100 ms RTT BBR flow, 10 ms BBR flow was suppressed as shown in Fig. 3.

B. BBR-E – 10 ms vs 10 ms & 10 ms vs 100 ms

The BBR-E showed not only same trend of fairness as the original BBR but also enhanced the stability further as can be observed in Fig. 2.

Fig. 5 shows the performance improvement of the BBR-E. During the first 20 seconds, the two flows started at the same time, a 100 ms RTT flow dominated. The 10 ms RTT flow gradually increased in bandwidth through continued probing. After the ProbeRTT phase at 20 s, both flows used the bandwidth almost fairly. The 10 ms BBR flow had very low throughput in competition with 100 ms BBR flow, but with the BBR-E both flows had similar throughput after stabilization.

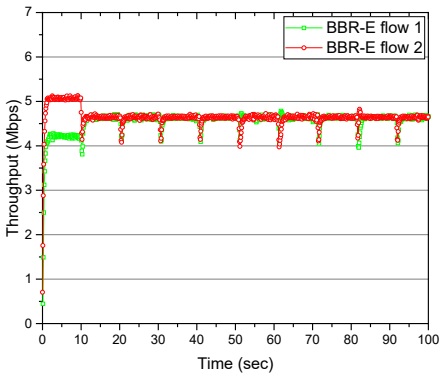

Fig. 4. Throughput comparison – 10 ms vs 10 ms BBR-E flows.

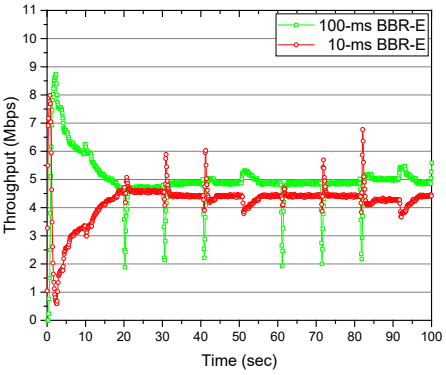

Fig. 5. Throughput comparison – 10 ms vs 100 ms BBR-E flows.

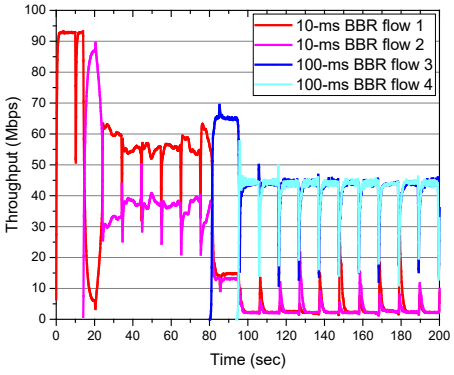

Fig. 6. Throughput of multiple BBR flows competing.

C. *Multiple BBR flows with different RTT*

We examined the behavior of multiple flows with different start times and RTTs having a common 100 Mbps bottleneck. Two 10 ms RTT flows started at 0 and 14 seconds, respectively, and two 100 ms RTT flows started at 80 and 94 seconds, respectively.

In a competition between original BBR flows with different minimum RTTs, the BBR-1 flow that was started earlier occupied a higher bandwidth as shown in Fig. 6. However, at 80 s, when the flow with the 100 ms RTT (flow 3) started, the two 10 ms (flow 1 and 2) flows lost the bandwidth instantly. After that, two 100 ms RTT flows occupied almost half of the bandwidth suppressing the other two.

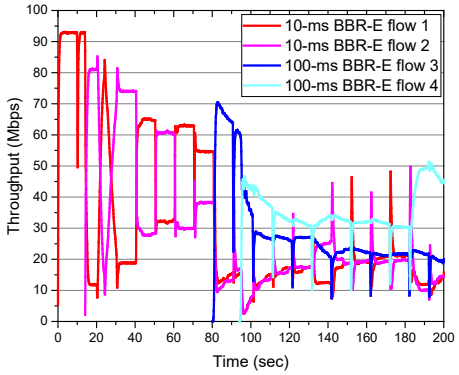

Fig. 7. A throughput of multiple BBR-E flows competing.

Fig. 7 shows the multiple BBR-E flow's competition. In the comparison between the 10 ms RTT flows, they reverse each other and gradually decrease their gap. When the 100 ms RTT flows intervene, the 10 ms RTT flows lose a large amount of bandwidth, but they still occupy some bandwidth through persistent probing. Although, the 10 ms BBR-E flows did not occupy their ideal share, they showed a sufficient amount of throughput as compared to 10 ms original BBR flows.

V. CONCLUSION

In this paper, we proposed an improved bottleneck queue buildup suppression method by applying and experimenting in real testbed experiments. The proposed BBR-E showed better throughput than BBR in comparison with different RTT flows. Also, through the experiment of multiple flows concurrent competition, we confirmed that the BBR-E occupied sufficient bandwidth when compared with the BBR.

ACKNOWLEDGMENT

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (No. NRF-2017M3C4A7083676) and by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019R1A2C1006249).

REFERENCES

[1] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh et al., "BBR:congestion-based congestion control," Commun. ACM, vol. 60, no. 2, pp. 58–66, 2017.

[2] L. Kleinrock, "Power and deterministic rules of thumb for probabilistic problems in computer communications," in Proc. of IEEE ICC, 1979.

[3] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of BBR congestion control," in Proc. of ICNP, 2017.

[4] S. Ma, J. Jiang, W. Wang, and B. Li, "Towards RTT Fairness of Congestion-Based Congestion Control," CoRR, vol. abs/1706.09115, 2017. [Online]. Available: http://arxiv.org/abs/1706.09115.

[5] G. H. Kim, I. Mahmud, and Y. Z. Cho, "Fairness Improvement of BBR Congestion Control Algorithm for Different RTT Flows," in Proc. of ICEIC, 2019.