ReBBR: Reproducing BBR Evaluation Results

Luke Hsiao
Stanford University
lwhsiao@stanford.edu

Jervis Muindi
Stanford University
jmuindi@stanford.edu

ABSTRACT

This report describes an attempt to reproduce one of the key findings from the BBR Paper by Cardwell et al. The primary result we pursue is performance of BBR compared to CUBIC in networks that have non-negligible packet loss. As reported in the original paper, we find that in our preliminary exploration, CUBIC has poor performance for loss rates above 0.1% whereas BBR is able to provide higher utilization of the network up to much high loss rates.

1 INTRODUCTION

In this report, we attempt to validate the experimental results of "BBR: Congestion-based Congestion Control" [1]. We start by looking at the goals, motivations and results from the BBR paper.

Goals *What was the original paper trying to solve?*

The original paper was trying to find a congestion control approach that would stay as close as possible to optimal network operating point in various network conditions. The network link is operating at the optimal point when bandwidth utilization is maximized and latency and loss are minimized, as shown by Leonard Kleinrock in 1979 [3]. Furthermore, Jeffrey Jaffe proved that a distributed algorithm that converged to this optimal point was an impossibility [2].

Loss-based congestion control operates by delivering full bottleneck bandwidth at the cost of high delay and frequent packet loss. Furthermore, Cardwell et. al. observed that because RTprop and BtlBw could be inferred from traces, measurements over time could result in an algorithm which converged with high probability to the optimal operating point. BBR is an algorithm that seeks to converge to this optimal point.

Motivation *Why is the problem important/interesting?*

At a high level, the Internet could be moving data much more efficiently. For example, cellular users can experience delays of seconds to minutes and even multi-Gbps infrastructure may deliver data at a few Mbps when transmitting over intercontinental distances. In addition, traditional congestion control algorithms such as CUBIC have a tough time operating efficiently when there is non-negligible packet loss in the network. This limitation comes from the CUBIC's use of packet loss as a signal for congestion, which can unnecessarily hinder throughput leading to poor performance.

Importantly, these limitations are not fundamental, but rather are a result of particular implementation choices made in designing TCP congestion control. In the original paper, the BBR authors seek to find an alternative to loss-based congestion control, in an effort to mitigate many of these issues.

The problem is particularly interesting because, if successful, we will be able to utilize the network infrastructure we have much more effectively, while reducing latency experienced by users, without needing to upgrade or change the network itself.

Results *What did the original authors find?*

The BBR paper describes a new form of congestion control that is based on actual congestion in the network. The insight from the authors is that their approach estimates bottleneck bandwidth and the minimum propagation delay so as to be able to get as close as possible to the ideal operating point of maximum bandwidth and minimal latency. We highlight several of the key results below.

The first finding is that BBR can quickly adopt to changes in bottleneck link bandwidth. During operation, BBR spends most of its time continuously probing if more bandwidth has become available. Figure 1 shows BBR reacting to a 2x increase and a 2x decrease to bottleneck bandwidth within about 2 seconds.

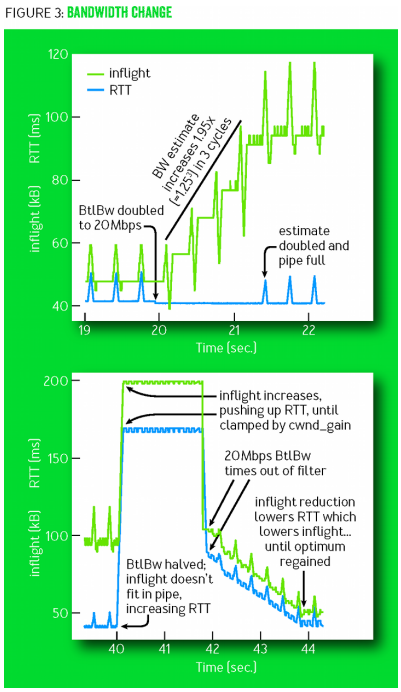

Figure 1: BBR converges to new bottleneck bandwidth at exponential rate. This figure is from the original paper.

The second finding is that BBR is able to avoid bloating router buffers unnecessarily and thus is able to keep RTT at a minimum. This is in contrast to CUBIC which continues to bloat a the bottleneck buffer until it observes a packet loss event from overflowing the buffer. Figure 2 illustrates hoe CUBIC and BBR behave in this regard.

Third, and one of the most interesting results, is that BBR outperforms CUBIC for non-negligible loss rates. Figure 3 shows a graph showing the performance comparison between BBR and CUBIC for varying loss rates.

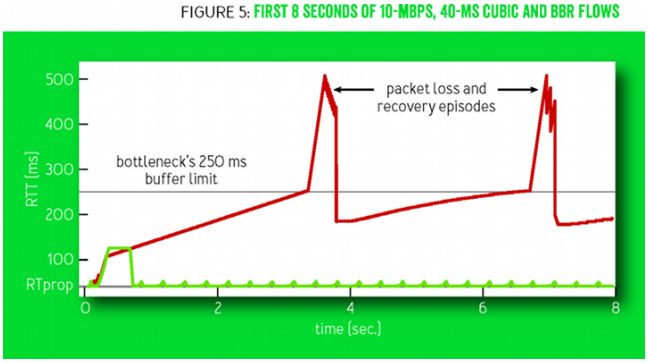

Figure 2: BBR avoids bloating router buffer unlike CUBIC. Figure from original paper.

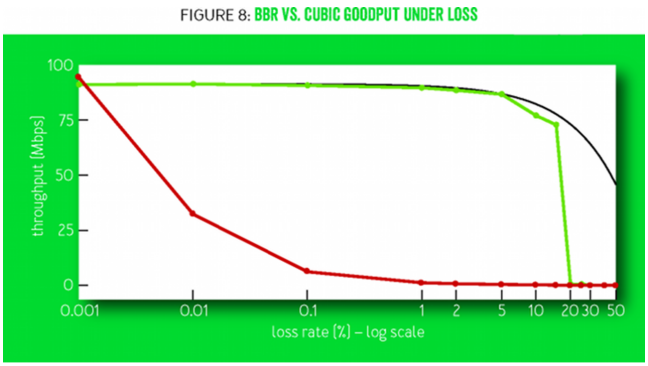

Figure 8 shows BBR vs. CUBIC goodput for 60-second flows on a 100-Mbps/100-ms link with 0.001 to 50 percent random loss. CUBIC's throughput decreases by 10 times at 0.1 percent loss and totally stalls above 1 percent. The maximum possible throughput is the link rate times fraction delivered (= *1 – lossRate*). BBR meets this limit up to a 5 percent loss and is close up to 15 percent.

Figure 3: Throughput vs loss rate comparing CUBIC vs BBR. Figure from original paper.

The authors report that Google has had a very good experience deploying BBR on B4—Google's data center to data center high-speed Wide Area Network built primarily from commodity switches that have shallow/small memory buffers. The limited amount of buffering means that there is the occasional packet loss when there are bursts of incoming packets that arrive at the same time. After switching to BBR from CUBIC, Google saw throughput improvements ranging from 2× to 25×.

Finally, the authors do acknowledge that there is still work to be done in terms of how BBR interoperates with existing loss-based congestion controls algorithms such as CUBIC. In cases where routers have large buffers, CUBIC senders can drown out BBR. Gracefully mitigating this is an ongoing area of research.

2 REPRODUCING BBR RESULTS

Goal *What subset of results did you choose to reproduce?*

The main result we focused on reproducing is the comparison between CUBIC and BBR and how their performance varies across different loss rate (shown in Figure 3, which appears as Figure 8 in the original paper).

Our goal is to not only verify the behavior of BBR compared to CUBIC in networks with various loss rates, but also explore the parameters of this particular experiment.

Motivation *Why that particular choice?*

We chose this result because it is one of the primary results of the paper, and is a large contributing factor of BBR's ability to better utilize network infrastructure (as seen in Google's B4 deployment). It is also one of the stronger claims of the paper, which shows that BBR differs significantly from the behavior of CUBIC, a loss-based congestion control algorithm and the current default in Linux.

3 PROJECT STATUS

The status of the project is that we have completed the setup of a VM with BBR available. This involved configuring a Linux (Ubuntu) VM with an updated kernel that included the BBR congestion control algorithm. In addition, we have access to the `tcp_bbr.c`[1] source, which allows us to modify and recompile the kernel module, if needed.

In addition, we have been able to write an experiment script in Python that collects data on BBR and CUBIC throughput performance at varying loss rates. We use MahiMahi[2] as our network emulator. To setup the experiment, we use a delay shell to simulate a 100ms delay, a loss shell to control the amount of packet loss, and a link shell to approximate a 100Mbps link and mimic the conditions of the original BBR results.

Once the data collection is complete, the script generates a plot of the performance between BBR and CUBIC.

A preliminary run produced the graph results in Figure 4. We can see that CUBIC only performs acceptably when the loss rate is extremely small. At loss rates of 0.01% and above, CUBIC performance deteriorates. BBR, on the other hand, is able to deliver a larger percentage of the available throughput when compared to CUBIC on loss rates as high as 50%.

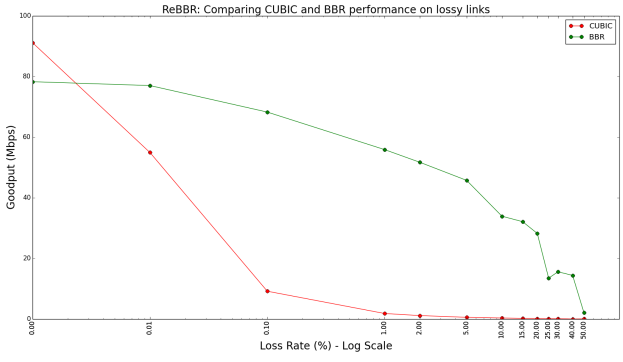

Figure 4: ReBBR: Throughput vs loss rate comparing CUBIC vs BBR.

[1] http://elixir.free-electrons.com/linux/latest/source/net/ipv4/tcp_bbr.c
[2] http://mahimahi.mit.edu/

Plan Currently, we still have some potential bugs to work out in our experimental script, and will need to perform a more rigorous experiment of reproducing the results if Figure 3 (e.g. more runs on longer traces). We would then hope to analyze and comment on the results and discrepancies.

Once the primary result has been reproduced, we would like to explore the different parameters of this experiment, such as which congestion control algorithms are compared against, the characteristics of the network, and perhaps the internal parameters of CUBIC if time permits. For example, we would like to see how BBR compares to other TCP congestion control algorithms such as TCP VEGAS or TCP NewReno, how BBR behaves on a lossy cellular link trace compared to CUBIC, or how the total bottleneck bandwidth available affects the comparison (e.g. running on 1Mbps rather than 100Mbps).

Currently, our experimental scripts are hard-coded for a particular case, and we will need to make some significant modifications to support exploring more of these parameters. We likely will not be able to explore all of these parameters, but hope to evaluate a few in the remainder of the quarter.

REFERENCES

[1] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-Based Congestion Control. *Queue* 14, 5 (2016), 50.

[2] Jeffrey Jaffe. 1981. Flow control power is nondecentralizable. *IEEE Transactions on Communications* 29, 9 (1981), 1301–1306.

[3] Leonard Kleinrock. 1979. Power and deterministic rules of thumb for probabilistic problems in computer communications. In *Proceedings of the International Conference on Communications*, Vol. 43. 1–43.