# ReBBR: Reproducing BBR Evaluation Results

Luke Hsiao
Stanford University
lwhsiao@stanford.edu

Jervis Muindi
Stanford University
jmuindi@stanford.edu

## ABSTRACT

This paper describes an attempt to reproduce one of the key findings from the BBR Paper by Cardwell et al. The primary result we are after is performance of BBR and CUBIC in networks link that have non-negligble packet loss. As reported in the original paper, we find that CUBIC has poor performance for loss rates above 0.1% whereas BBR is able to deal with loss much better.

## 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 [1] 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 is minimized.

**Motivation** Why is the problem important/interesting?

This is importannt problem to tackle because traditional congestion control algorithms such as CUBIC have a tough time operating efficiently when there is non-negligible packet loss on the packet. This limitation comes from the CUBIC's use of packet loss as a signal for congestion, which can unnecesarily hinder throughput leading to poor performance.

**Results** What did the original authors find?

The BBR paper descibes a new form of congestion control that is based on actual congestion going in the network. The insight from the authors is that their approach estimates bottleneck bandwidth and then bottleneck latency in succession so as to be able to get as close as possible to the ideal operating point of maximum bandwidth and minimal latency. The paper had severals findings.

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

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

The third and perhaps most important finding 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.

The authors report that Google has had a very good experience deploying BBR on B4. B4 is Google data center to data center high-speed Wide Area Network and it's 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 is a a burst of incoming packets that arrive at the same time. After switching to BBR from CUBIC, Google saw throughput improvements ranging from 2x to 25x.

Lastly, 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 onging area of research.

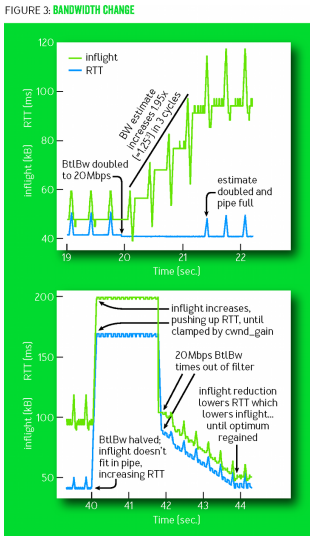


**Figure 1: BBR converges to new bottleneck bandwidth at exponential rate.**

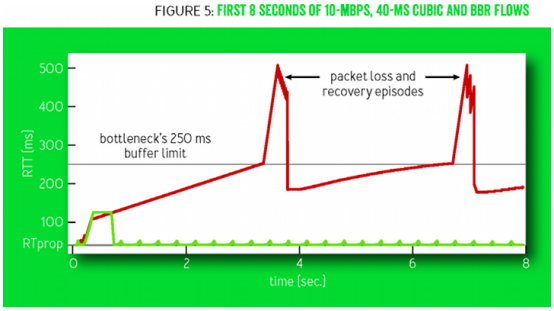


**Figure 2: BBR avoids bloating router buffer unlike CUBIC.**

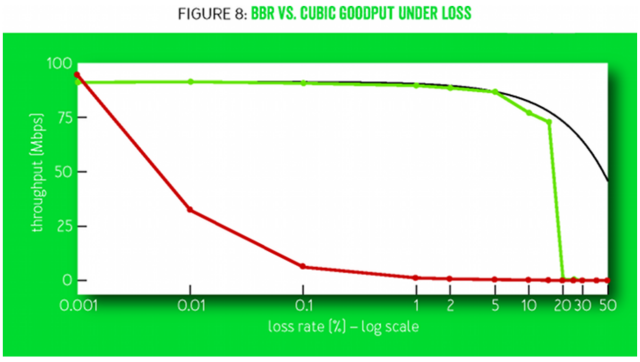


Figure 8 shows BBR vs. CUBIC goodput for 60–second flows on a 100–Mbps/100–ms link with 0.001 to 50 percent random loss. CUBIC's throughput decreases by 10 times at 0.1 percent loss and totally stalls above 1 percent. The maximum possible throughput is the link rate times fraction delivered (= *1 – lossRate*).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

## 2 REPRODUCING BBR EVALUATION

**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.

This result can be seen in Figure 3.

**Motivation** Why that particular choice?

We chose this result because it is a good representative summary that captures the primary essence of the paper. This is that loss-based congestion control schemes such as CUBIC are non-optimal because their use of packet loss as a signal for congestion makes them very sensitive to any (regular) packet loss that occurs on a path and it also contributes to buffer bloat which increases latency.

## 3 PROJECT STATUS

The status of the project is that we have completed the setup of a machine with BBR available. This involved configuring a Linux (Ubuntu) machine with a kernel that included the BBR congestion control algorithm.

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. The network emualtor used is MahiMahi. 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.

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

Our 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 0.01BBR on the other hand is able to deliver decent performance when compared to CUBIC on loss rates as high as 20

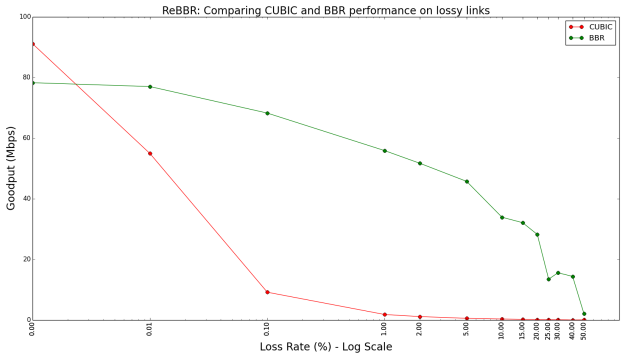


**Figure 4: ReBBR: Throughput vs loss rate comparing CUBIC vs BBR.**

**Plan** For the remaining days, we need to make the experiment setup run on Google Cloud. Running a machine on Google Cloud costs a few dollars, so for convienence and for faster iterating, we have been doing our work in a local Linux Ubuntu VM. The next step is to replicate that setup to run successfully in the Google Cloud.

We would also like to explore how BBR compares to other modern TCP congestion control algorithms such as VEGAS and NEW RENO.

We are also interested on evaluating how does the performance between CUBIC and BBR vary, if at all, when the network link is running an order of magnitude slower. For example at 10Mbps, 1Mbps, 0.1Mbps (100kbps) and 0.01Mbps (10kbps). Similarly, it would also be interesting to see the variation when the RTT changes by an order of magnitude. For example at 0.01s, 0.1s, 1.0s, 10, 60s. Performance on the low bandwidth, high latency links should give insights into potential benefits of BBR when deployed over slow 2G cellular links.

Finally, we need to write up our final report discussing the results we found.

Admittedly, this is quite a bit of material and we'll try to get through as much of it as possible, time permitting.

## 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.