Replication: "When to Use and When Not to Use BBR"

Soumyadeep Datta
sdatta@nyu.edu
New York University
Brooklyn, New York, USA

Fraida Fund
ffund@nyu.edu
New York University
Brooklyn, New York, USA

ABSTRACT

We replicate the paper, "When to Use and When Not to Use BBR: An Empirical Analysis and Evaluation Study" by Cao *et al*, published in IMC 2019 [2], with a focus on the relative goodput of TCP BBR and TCP CUBIC for a range of bottleneck buffer sizes, bandwidths, and delays. We replicate the experiments performed by the original authors on two large-scale open-access testbeds, to validate the conclusions of the paper. We further extend the experiments to BBRv2. We package the experiment artifacts and make them publicly available so that others can repeat and build on this work.

CCS CONCEPTS

• **Networks** → *Data path algorithms*; *Network control algorithms*; **Transport protocols**.

KEYWORDS

TCP Protocol Evaluation, BBR, CUBIC, Testbeds

ACM Reference Format:
Soumyadeep Datta and Fraida Fund. 2023. Replication: "When to Use and When Not to Use BBR". In *Proceedings of the 2023 ACM Internet Measurement Conference (IMC '23), October 24–26, 2023, Montreal, QC, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3618257.3624837

1 ORIGINAL PAPER AND MOTIVATION FOR REPLICATION

In [2], Cao *et al* compare the performance of TCP BBR [3] and TCP CUBIC [6] in 640 different emulated network scenarios (with different bandwidth, bottleneck buffer size, and RTT).

They find that:

- For shallow buffers, BBR significantly outperforms CUBIC in terms of goodput, especially in the high bandwidth-delay product (BDP) regime, despite a higher number of retransmissions, as shown in Figure 5(a) of the original paper.
- For deep buffers, CUBIC slightly outperforms BBR in terms of goodput, except for an extremely high bandwidth-delay product (BDP) regime, as shown in Figure 5(b) of the paper.

Since the paper was published in 2019, however, BBR has grown in popularity, to the extent that a recent census found that it makes up almost 40% of all Internet traffic [9]. It is therefore crucial to re-examine the conclusions of [2]. We are also interested in extending the evaluation to BBRv2 [4], which differs from the original BBR in many implementation details. Our focus in this paper is to replicate the results of the original paper *independently*. Therefore, rather than contacting the authors of the original paper, we implemented scripts to replicate the experiments ourselves. Cao *et. al.* provide a detailed description of the system specifications (such as the specific Linux OS and kernel version), network settings (such as buffer sizes, bandwidths, and delays), and other implementation details in the original paper [2], which we used to realize a comparable experimental setting.

To independently verify the claims in this paper, users of the CloudLab [5] or FABRIC [1] platforms can use the artifacts and instructions in our repository [1] to reserve testbed resources, run the experiments, and visualize the results.

2 EXPERIMENT METHODOLOGY

In this section we describe the details of our experiment methodology, including the experimental platforms, topology, network scenarios, flow generation, socket buffer configuration, Linux kernel versions, and validation of the experiment setting. Where relevant, we compare our choices to those in the original paper [2].

Experimental platforms: The comparable experiments performed by Cao *et al* in [2] were conducted their own LAN testbed. To support easy sharing of experiment artifacts for further replications and extensions of this work, we replicate these results on Cloudlab [5], a shared testbed facility that provides bare metal servers, and on FABRIC [1], which provides virtual machines connected by high-speed links.

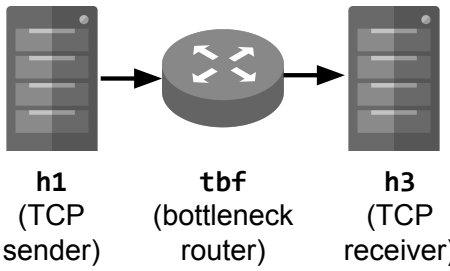


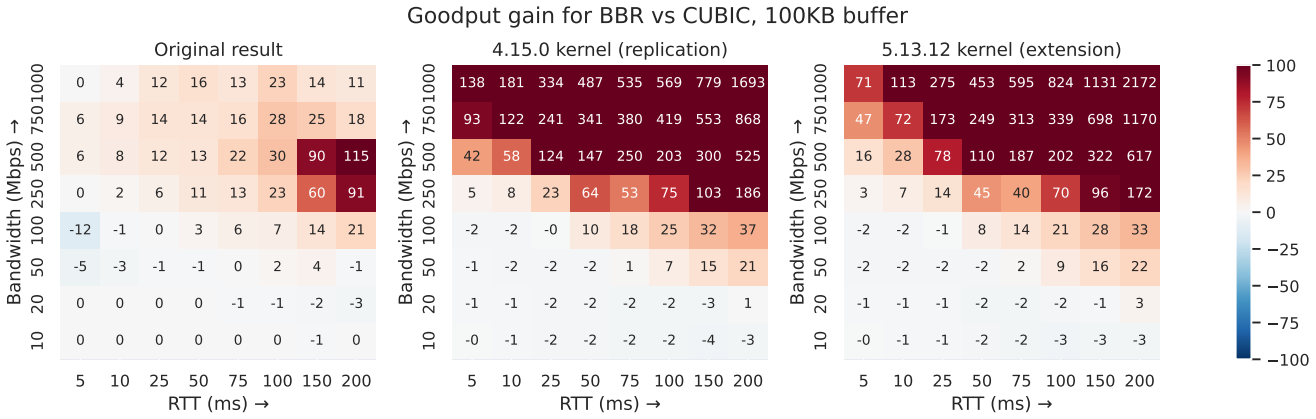
Figure 1: Network topology and direction of data flow.

Topology: We replicate the topology proposed by the authors in the original paper [2], with one host (denoted as `h1`) sending data to another host (denoted as `h3`) via an intermediate router (denoted as `tbf`) as shown in Figure 1. All nodes in the topology

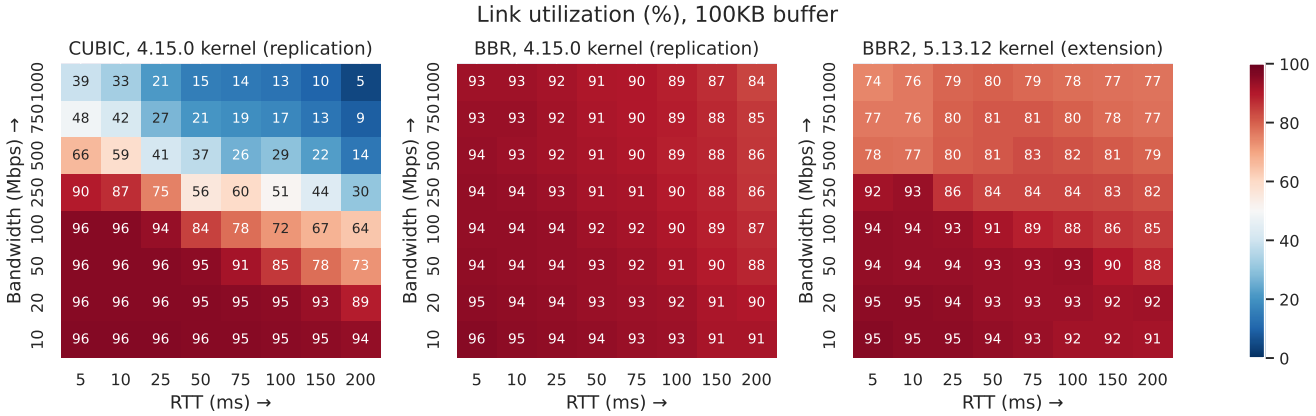
[1] Experiment artifacts: https://github.com/sdatta97/imcbbrrepro

IMC '23, October 24–26, 2023, Montreal, QC, Canada

ACM ISBN 979-8-4007-0382-9/23/10…$15.00
https://doi.org/10.1145/3618257.3624837



(a) (A positive value indicates an advantage to BBR over CUBIC.) Compared to the original result, which found that the relative advantage of BBR vs. CUBIC was on the order of 10x in high-BDP scenarios, we observed an advantage on the order of 100x or 1000x, for both BBRv1 (replication) and BBRv2 (extension).



(b) From the link utilization, we find that the relative advantage of BBR over CUBIC in our experiment is mainly driven by the severe underutilization of CUBIC in high-BDP scenarios with a very shallow bottleneck buffer. The utilization of BBR is generally good in these scenarios, although slightly lower in BBRv2.

Figure 2: Original result, replication, and extension to a newer BBR protocol version with a shallow buffer (100KB). The top right corner of each figure is of interest, showing the network scenarios with a high BDP.

are implemented in Linux servers. (On FABRIC, where resources are virtualized and not bare metal servers, we use VMs with 4 cores and 16 GB RAM.)

Network scenarios: The router (`tbf`) uses a token bucket filter (TBF) mechanism on its egress interface towards the host `h3` to limit the bottleneck bandwidth and buffer size using the Linux `tc-htb` utility. The router also applies the `tc-netem` utility on its ingress interfaces towards the host `h1` to emulate network delays (with a sufficiently large `netem` buffer size to ensure that no ACKs are dropped due to delay emulation). Following the original paper, we consider these network conditions on the intermediate router:

- Round trip times (ms): 5, 10, 25, 50, 75, 100, 150, 200
- Bottleneck bandwidths (Mbps): 10, 20, 50, 100, 250, 500, 750, 1000
- Bottleneck buffer sizes: 100 KB, 10 MB

Flow generation: For each of the network configurations above, we generate a single TCP flow using the `iperf3` utility for 60 seconds, and record the goodput reported by the receiver and the number of retransmissions reported by the sender. We repeat each experiment 5 times and use the average of the 5 trials.

Socket buffer configuration: In the original paper, the authors report that they used `sysctl` to set the TCP read and write memory to $2^{31} - 1$ bytes, the maximum allowed value, to avoid having the socket buffer size be a limiting factor. We set the same at the system level. However, they did not specify the socket buffer size configured at the application level by the `iperf3` "window size" parameter. Following what we understand to be the intent of the original paper, we use 100 MB as the socket buffer size, and confirmed by experiment (using the `ss` utility) that this was sufficient to avoid limiting the connection by socket buffer size.

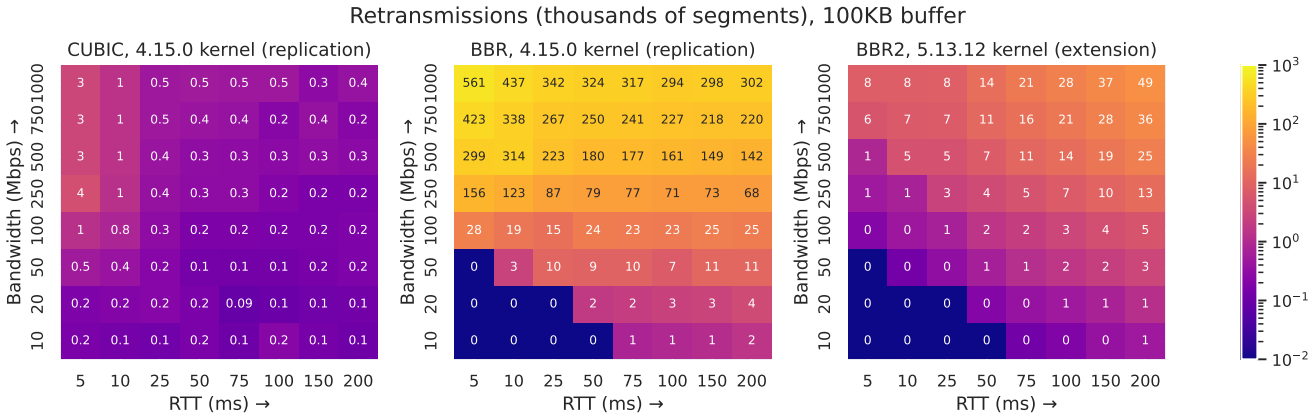


Figure 3: (Colors in log scale.) As in the original, we find that a 60 second BBR flow may experience hundreds of thousands of retransmitted segments, but that CUBIC has far fewer retransmissions. Beyond the original result, we observe that the newer BBRv2 protocol version has an order of magnitude fewer retransmissions.

Linux kernel versions: The original result [2] is based on the TCP congestion control implementations in Linux kernel 4.15. We use the same kernel (in Ubuntu 18.04), and also consider the new BBRv2 protocol implementation in kernel version 5.13 (in Ubuntu 20.04).

Validation of experiment setting: To validate that the underlying host and network capabilities on the testbed platform is not a limiting factor, after instantiating resources on each testbed we confirmed using `iperf3` and `ping` that the bandwidth between `h1` and `h3` was close to 10 Gbps, and the round trip time (RTT) was less than 1 ms. This is substantially better network performance than the maximum bandwidth in our experiment, which is 200 Mbps, or the minimum RTT, which is 5 ms. (On CloudLab, a variety of different bare metal servers are available - to ensure that the experiment is valid, we select server types that have at least 10 Gbps Ethernet NICs.)

3 EXPERIMENT RESULTS

In this section, we describe the results of our replication and of our extension to newer TCP implementations. To evaluate the performance improvement of TCP BBR over TCP CUBIC in terms of goodput, we use the goodput gain (GpGain) metric, defined in the original paper [2] as follows:

$$\text{GpGain}_{\text{CUBIC}}^{\text{BBR}} = \frac{\text{goodput}|_{\text{BBR}} - \text{goodput}|_{\text{CUBIC}}}{\text{goodput}|_{\text{CUBIC}}} \times 100. \quad (1)$$

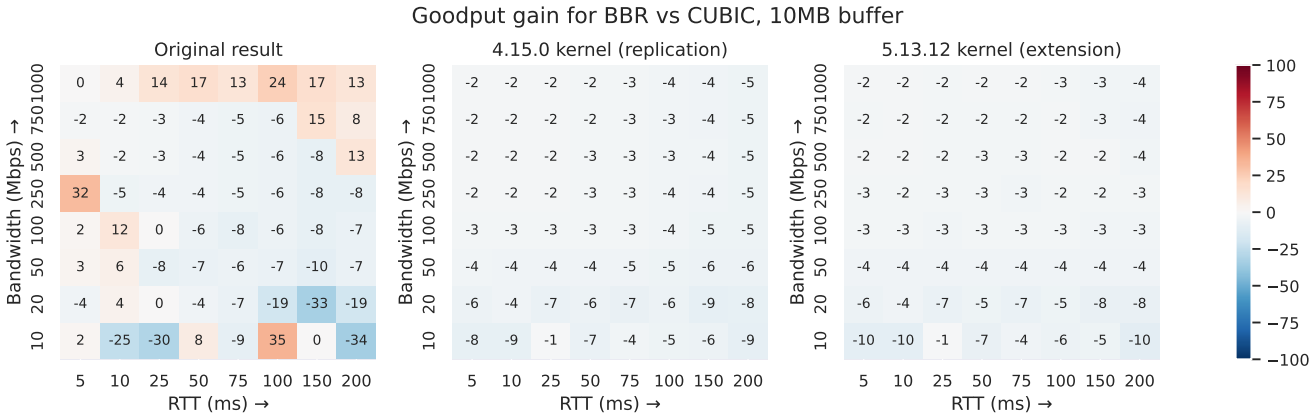
We also report the number of retransmitted segments per 60 second flow. We organize these results according to the two key findings that we seek to replicate, one from the experiment with a shallow buffer (100 KB) and one from the experiment with a deeper buffer (10 MB).

3.1 Shallow buffer result

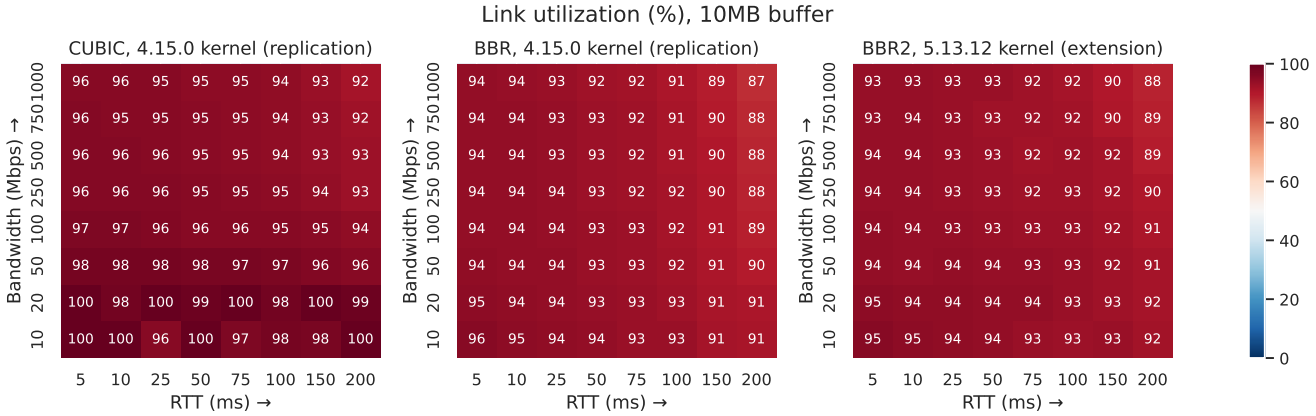
In the original paper [2] Figure 5a (first panel of Figure 2a here) shows that with a 100 KB bottleneck buffer and a high BDP, BBR significantly outperforms CUBIC in terms of goodput. Our experiment results, shown in Figure 2a, validate this overall conclusion with respect to goodput gain. However, where the original result finds that the advantage of BBR vs. CUBIC is about 10x in most high-BDP scenarios, in our experiments we measured an advantage for BBR on the order of 100x or 1000x, for both the original version of BBR and for the more recent BBRv2 protocol version. To explain this result, we also show the link utilization of all three protocol versions - CUBIC, BBR, and BBRv2 - in Figure 2b. We observe that in our experiments, CUBIC severely underutilizes the link on high-BDP paths, with link utilization less than 10% in the most extreme case.

It is difficult to attribute a specific reason to the difference in magnitude between our result and the original, since in the original, only the goodput *gain* is reported and not the individual congestion control variants' goodput or link utilization. However, we observe that under some experimental settings that were not specified explicitly in the original paper, BBR had much worse link utilization. For example, the authors did not specify if they changed the `iperf3` socket buffer size from the default value, under which we observed that the TCP flow could be limited by socket buffer size. Similarly, the authors did not report whether they increased the default buffer size limit in `netem` to ensure that no ACKs were dropped. Under the default settings for `iperf3` socket buffer size and `netem` buffer limit, the BBR link utilization for the highest BDP path (1000 Mbps bandwidth, 200 ms delay) was only 12%, rather than the 84% we report in Figure 2b.

The original paper [2] further clarifies in Figure 5c and 5d that BBR's goodput advantage comes at the cost of many more retransmissions, with a 60 second flow having hundreds of thousands of retransmitted segments in BBR and far fewer in CUBIC. Our experiments validate this conclusion as well, with results shown in Figure 3. As in the original, we see that the BBR flows experience hundreds of thousands of retransmissions over high-BDP paths,



(a) (A positive value indicates an advantage to BBR over CUBIC.) The original result suggests an advantage of about 10x for BBR in high BDP regimes (top right of each panel) and a similar advantage for CUBIC in low-bandwidth scenarios (bottom of each panel) with deep buffers. In contrast, in our experiments we consistently see a very small advantage for CUBIC.



(b) In our experiments, CUBIC, BBR, and BBRv2 generally have good link utilization, so none of these congestion control variants has a substantial goodput advantage over any other.

Figure 4: Original result, replication, and extension to a newer BBR protocol version for a deep buffer (10MB).

while CUBIC flows experience far fewer. We also consider the extended result to BBRv2, where we see an order of magnitude fewer retransmissions than the original BBR protocol version.

This difference between BBR a BBRv2 is briefly explained as follows. In BBR and BBRv2, the sending rate may be limited either by the pacing rate (which is determined by BBR's estimate of the bottleneck bandwidth) or by CWND (which is set as a multiple of the BDP, so it is determined by BBR's estimate of the bottleneck bandwidth and minimum RTT of the path). In our experiments, when the sender is limited by pacing rate, as with deep buffer or low-BDP paths, BBR's estimate of the bottleneck bandwidth and minimum RTT (as observed using the `ss` utility) is similar in the original and in BBRv2. Thus, in these settings, both protocol versions achieve similar (high) link utilization and minimal retransmissions. However, when the sender is CWND limited, as in shallow buffer high-BDP paths, the behavior is very different because the CWND in BBRv2 is set much lower by design [4]. For example, in the network with 200 ms RTT and 1000 Mbps bottleneck bandwidth, the median CWND of the BBR flow is approximately 400 Mbits, while the median CWND of the BBRv2 flow is only 157 Mbits. BBRv2 is also less aggressive in bandwidth probing, with the explicit goal of reducing loss rate in shallow buffers [4], so in high-loss scenarios it also estimates a smaller bottleneck bandwidth - 881 Mbps median bottleneck bandwidth estimate for BBRv2 vs. 992 Mbps bottleneck bandwidth estimate for the original BBR in the network with 200 ms RTT and 1000 Mbps bottleneck bandwidth. We elaborate further on the differences between BBR and BBRv2 in Section 4.

Summary: We validate the overall findings of the original, although the magnitude of the result is different. We also extend the result and show that for high-BDP paths with shallow buffers, BBRv2 has lower goodput but fewer retransmissions than the original BBR.

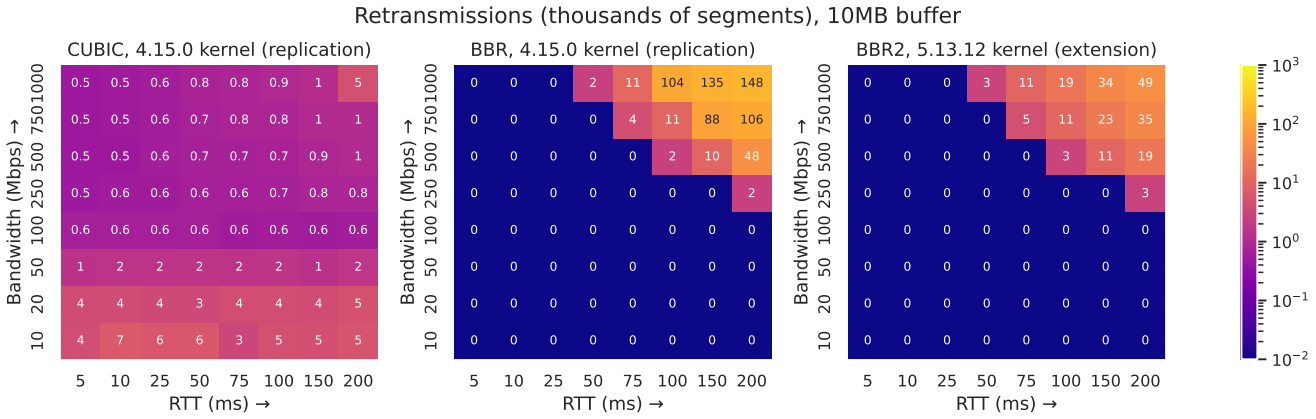


Figure 5: (Colors in log scale.) The original paper, while not reporting specific values, states that in the deep buffer scenario there are far fewer retransmissions for both CUBIC and BBR than in the shallow buffer scenario. We confirm this finding for BBR, but not necessarily for CUBIC.

3.2 Deep buffer result

In the original paper [2] Figure 5b (the first panel of Figure 4a) shows that with a 10 MB bottleneck buffer, CUBIC outperforms BBR, except for very high BDP regimes, where BBR has an advantage. In our experiments, however, we find a consistent but small advantage for CUBIC over both BBR and BBRv2. Furthermore, the utilization of all three congestion control protocols (Figure 4b) is generally very close to 100%, although all three have slightly lower utilization when the BDP is large.

The original paper [2] does not report specific values for retransmissions in the deep buffer scenario, but does say that far fewer retransmissions occur for both CUBIC and BBR. In our observation (Figure 5), BBR indeed has many fewer retransmissions, with zero retransmissions in all but the highest-BDP paths. For CUBIC, however, we observe more retransmissions in the deep buffer case when either the BDP is high or the link bandwidth is small.

Summary: While the original finding suggests a goodput advantage for BBR in high-BDP paths and a goodput advantage for CUBIC in low-bandwidth paths, we observe a slight advantage for CUBIC in all of the deep buffer scenarios. We also have mixed findings relative to the original for number of retransmissions in the deep buffer scenarios.

4 DISCUSSION

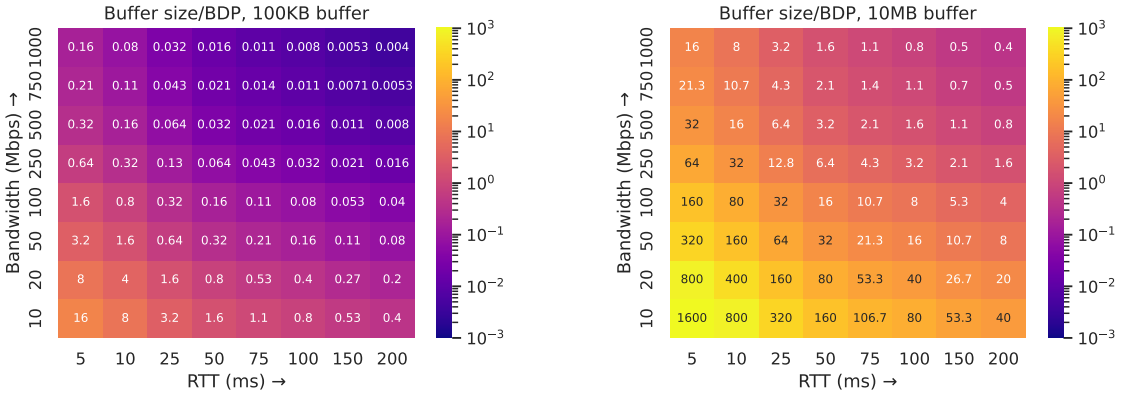
Understanding the results with respect to other published work: The results of our replication are largely in line with other related work. For example, [8] shows that for a single flow, TCP CUBIC requires a buffer size at least 0.4 BDP to achieve full link utilization. In fact, we see in Figure 2b and Figure 6a that CUBIC utilization is degraded when the bottleneck buffer size is smaller than 0.4 BDP. Similarly, our results in Figure 3, Figure 6a, Figure 5, and Figure 6b agree with the previous finding that to avoid excessive retransmission in BBR, the buffer size should be at least 1 BDP [7]. We see zero retransmissions for BBR when the buffer size is sufficiently large relative to the BDP.

Understanding the effect of experiment and environment settings: In some cases, our results were not necessarily in line with the original paper [2], with either a difference in magnitude or a difference in trend. However, we observe in our experiments that many experiment settings can interact with the congestion control protocol, including:

- The socket buffer size setting in the application, which if too small may limit the sending rate (instead of CWND or other congestion control parameters),
- The setting of the queues that were not explicitly specified, e.g. the default egress queue at the sending host,
- The speed of the network hardware (the original authors used 1 Gbps NICs, in our experiments the NICs were at least 10 Gbps) - even if the maximum bottleneck bandwidth in the experiment does not exceed the NIC rate, the burst behavior is affected,
- The maximum segment size - whether it is an Ethernet link with 1500 B MTU or one that supports jumbo frames with 9000 B MTU. Also, the segment offload setting on the NIC can change the "effective" MTU,
- The behavior of the operating system with respect to loss detection and retransmission, which is known to have changed in recent Linux kernels with the implementation of RACK [10].

Our experience highlights the importance of validating experiments to be sure that the expected environment is realized, and the benefit of sharing experiment artifacts so that others can easily replicate the work and identify differences.

Understanding the results in light of changes to BBR in BBRv2: In addition to replicating the original result, we extended this study to consider BBRv2. Unlike the original BBR protocol version, which does not consider loss rate at all, BBRv2 has a target loss rate that it tries not to exceed. It keeps track of bounds on estimated bandwidth and number of bytes in flight, with consideration only of samples where a packet loss threshold was not exceeded, and uses these to limit CWND [4]. In our experiments, we observe that BBRv2 has slightly lower utilization but much less retransmission



(a) The "shallow" buffer scenario.

(b) The "deep" buffer scenario.

Figure 6: Buffer size expressed as a ratio of buffer size to path BDP. (Colors are in log scale.)

on network paths where the flow is CWND limited, compared to the original BBR protocol.

5 CONCLUSION

We replicate the experiments to generate Figure 5 in the original paper [2] by Cao *et al* on the large-scale CloudLab and FABRIC testbeds. We were largely able to validate their findings, with some small differences. We also extend the study to BBRv2, and note some differences in performance compared to the original BBR, especially for shallow buffers. Specifically, we observe that BBRv2 has fewer retransmissions, but also has lower utilization on high-BDP paths. Our code and other artifacts (documentation, instructions for instantiating experiments on the testbed facilities, and data analysis scripts) are available publicly[1]. We hope that with these materials future researchers will be able to quickly perform similar studies at scale, not only for TCP CUBIC and BBR but also for other TCP variants.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2226408.

REFERENCES

[1] Ilya Baldin, Anita Nikolich, James Griffioen, Indermohan Inder S. Monga, Kuang-Ching Wang, Tom Lehman, Paul Ruth, and Ewa Deelman. 2019. FABRIC: A National-Scale Programmable Experimental Network Infrastructure. *IEEE Internet Computing* 23, 6 (nov 2019), 38–47. https://doi.org/10.1109/MIC.2019.2958545

[2] Yi Cao, Arpit Jain, Kriti Sharma, Aruna Balasubramanian, and Anshul Gandhi. 2019. When to Use and When Not to Use BBR: An Empirical Analysis and Evaluation Study. In *Proceedings of the Internet Measurement Conference* (Amsterdam, Netherlands) *(IMC '19)*. Association for Computing Machinery, New York, NY, USA, 130–136. https://doi.org/10.1145/3355369.3355579

[3] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-Based Congestion Control. *ACM Queue* 14, September-October (2016), 20 – 53. http://queue.acm.org/detail.cfm?id=3022184

[4] Neal Cardwell, Yuchung Cheng, Soheil Hassas Yeganeh, Ian Swett, and Van Jacobson. 2023. *BBR Congestion Control.* Technical Report. https://www.ietf.org/archive/id/draft-cardwell-iccrg-bbr-congestion-control-02.html

[5] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. 2019. The Design and Operation of Cloudlab. In *Proceedings of the 2019 USENIX Conference on Usenix Annual Technical Conference* (Renton, WA, USA) *(USENIX ATC '19)*. USENIX Association, USA, 1–14.

[6] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS operating systems review* 42, 5 (2008), 64–74.

[7] Mario Hock, Roland Bless, and Martina Zitterbart. 2017. Experimental evaluation of BBR congestion control. In *2017 IEEE 25th International Conference on Network Protocols (ICNP)*. https://doi.org/10.1109/ICNP.2017.8117540

[8] Wolfram Lautenschlaeger and Andrea Francini. 2015. Global synchronization protection for bandwidth sharing TCP flows in high-speed links. In *2015 IEEE 16th International Conference on High Performance Switching and Routing (HPSR)*. 1–8. https://doi.org/10.1109/HPSR.2015.7483103

[9] Ayush Mishra, Xiangpeng Sun, Atishya Jain, Sameer Pande, Raj Joshi, and Ben Leong. 2019. The Great Internet TCP Congestion Control Census. *Proc. ACM Meas. Anal. Comput. Syst.* 3, 3, Article 45 (dec 2019), 24 pages. https://doi.org/10.1145/3366693

[10] Ufuk Usubütün, Fraida Fund, and Shivendra Panwar. 2023. Do Switches Still Need to Deliver Packets in Sequence?. In *2023 IEEE 24th International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 89–95.

ETHICS

This paper does not raise any ethical issues. This paper does not involve research with human or animal subjects.