engineering proceedings

Proceeding Paper

Improved Control Mechanism of Bottleneck Bandwidth and Round-Trip Propagation Time v3 Congestion with Enhanced Fairness and Efficiency [†]

Hung-Chi Chu * and **Hao-Chu Chiang**

Department of Information and Communication Engineering, Chaoyang University of Technology, Taichung 413310, Taiwan; acbhui4061@gmail.com
* Correspondence: hcchu@cyut.edu.tw

† Presented at the 2024 IEEE 7th International Conference on Knowledge Innovation and Invention, Nagoya, Japan, 16–18 August 2024.

Abstract: The widespread adoption and popularity of various applications have led to large and frequent data transmissions, resulting in network congestion, high packet delays, and packet loss. In 2016, Google proposed the Bottleneck Bandwidth and Round-trip propagation time (BBR) algorithm to mitigate network congestion. However, its network fairness is poor. Consequently, BBRv2 and BBRv3 were introduced in 2018 and 2023 as improved versions. Although BBRv2 exhibited enhanced fairness, its bandwidth utilization rate was lower than that of other existing methods. Meanwhile, BBRv3 still lacked bandwidth fairness in its initial transmission. Therefore, we have improved the fairness based on BBRv3 by considering the maximum sending rate and utilizing connections at different times. Good fairness and bandwidth utilization are maintained on the bottleneck bandwidth with the improved method. The method outperforms Cubic Binary Increase Congestion Control (CUBIC) and BBRv3 in terms of bandwidth utilization and network usage fairness.

Keywords: BBR; congestion control

Academic Editors: Teen-Hang Meen, Chun-Yen Chang and Cheng-Fu Yang

Published: 24 February 2025

Citation: Chu, H.-C.; Chiang, H.-C. Improved Control Mechanism of Bottleneck Bandwidth and Round-Trip Propagation Time v3 Congestion with Enhanced Fairness and Efficiency. *Eng. Proc.* **2025**, *89*, 11. https://doi.org/10.3390/engproc2025089011

1. Introduction

With the widespread adoption of the Internet, people's demand for and reliance on it has increased significantly. The proliferation of real-time and multimedia transmissions has exposed the limitations of traditional network structures. Although Transmission Control Protocol/Internet Protocol (TCP/IP) [1,2] has successfully addressed the challenge of connecting data and messages between different computers, the growing network demand and increased data transfer from connected devices have caused network overload. This results in congestion or even paralysis, causing high latency, packet traffic congestion, and packet loss.

The TCP congestion control mechanism has evolved to address these congestion issues. Early TCP implementations used a timeout-based mechanism, assuming that packet loss occurred when a packet took too long to traverse the network, reducing the transmission rate. However, this approach proved too conservative, causing the TCP transfer rates to drop too rapidly when real congestion occurred, thus undermining the overall network performance. As time progressed, congestion control mechanisms such as Tahoe, Reno, CUBIC, and their related extended versions were developed to better address the growing demands of modern networks. However, these congestion control mechanisms continue to compete with one another in terms of bandwidth utilization and fairness.

In this context, Google proposed BBR in 2016, a new congestion control algorithm. BBR aims to maximize network transmission rate utilization while keeping latency low to provide a better user experience. One of BBR's key features is its ability to proactively probe the network environment. It monitors the network's bandwidth and round-trip propagation time in real time, adjusting the transmission rate based on this information. This active probing capability allows BBR to adapt quickly to network changes and effectively avoid congestion. BBR incorporates advanced congestion control algorithms, including ProbeBW and ProbeRTT, to better handle various network environments in steady-state situations. The successful application of BBR in Google's services, such as YouTube and Google Cloud, demonstrates its excellent performance in modern network environments. Due to its effectiveness, BBR is gradually gaining attention from other ISPs and has become an important trend in network congestion control.

However, BBR has shortcomings in terms of fairness during transmission. In BBRv1, connections with different congestion control mechanisms such as CUBIC resulted in almost 10 s of near-maximum bandwidth for each side, leading to unfair behavior. BBRv2 improved the ProbeBW stage and introduced Explicit Congestion Notification (ECN) support and 3% packet loss tolerance, resulting in better fairness but with much lower bandwidth utilization compared to that of CUBIC. BBRv3 reduces the sending rate and adjusts parameters to better optimize fairness, but unfair situations still occur in the early and mid-term stages of connections.

This study aims to address the drawbacks of BBRv3, which struggles to achieve fair bandwidth allocation under different congestion control scenarios. The proposed method involves restricting various parameters during the transmission process. Specifically, the StartUP stage is used to probe the bandwidth status parameters, which are then adjusted to optimize the network bandwidth fairness. This method mitigates the deficiencies observed in BBRv3.

2. Related Work

Congestion controls based on packet loss and active probing are developed in this study.

2.1. Congestion Control Based on Packet Loss

CUBIC is a congestion control algorithm stemming from BIC [3]. It differs from the previously mainstream Reno version as its primary feature is the use of a cubic function to adjust the congestion window. This adjustment is conducted based on the time elapsed since the last decrease in the congestion window. The cubic function equation is defined as shown in (1):

$$W(t) = C \times (t - K)^3 + W_{max} \tag{1}$$

where C is the CUBIC parameter, t is the time elapsed since the last window reduction, and K is the time required to increase W to W_{max} when no further packet loss occurs.

When congestion occurs, CUBIC records the current congestion window and decreases it in a multiplicative manner to alleviate network congestion. Then, it rapidly increases the window to W_{max} using a concave segment of a cubic function to restore the congestion window to the maximum traffic level that existed before packet loss occurred. Once the congestion window reaches W_{max}, CUBIC maintains this level for a period, carefully probing for packet loss to determine if the limits of the available bandwidth have been reached. If no packet loss occurs, CUBIC uses convex functions to start increasing the congestion window until another packet loss event occurs. Notably, CUBIC's design is not related to the Round-Trip Time (RTT) but to the time interval between two packet loss events, as shown in Figure 1. CUBIC has three phases: to find current bandwidth, to avoid congestion, and to probe more bandwidth. When a timeout occurs, packets will be lost, and the process ends the "to find the current bandwidth" phase and enters the "to avoid congestion" phase. At this time, W is W_{max}. In the "to avoid congestion" phase, when $W = W_{max}/2$, it enters the "to probe more bandwidth" phase. If a Timeout occurs during this phase, the process returns to the "to avoid congestion" phase. The threshold is defined as the difference between the congestion window values at the end of the most recent "to avoid congestion" phase.

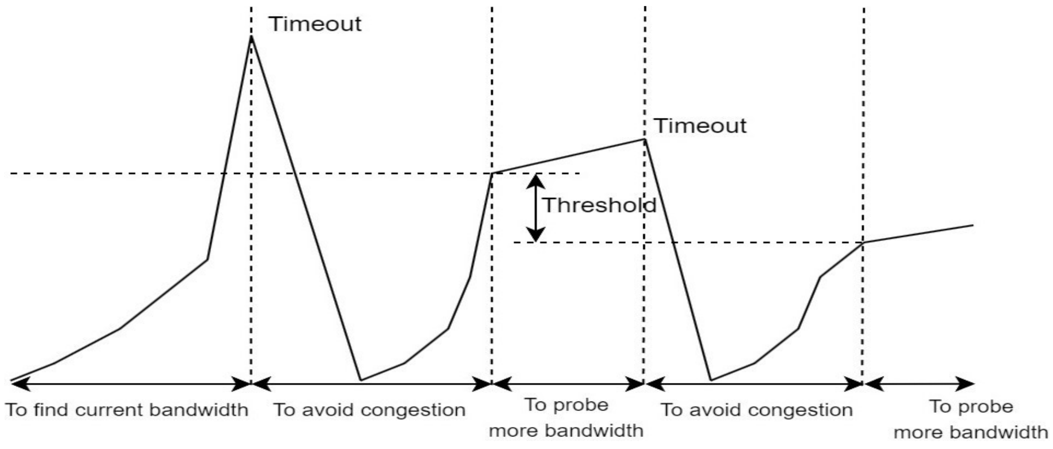

Figure 1. Schematic of CUBIC congestion control.

2.2. Congestion Control Based on Active Probing

BBRv1 probes the maximum available bandwidth and delay state to constantly detect the maximum bandwidth (maxBw) and the minimum RTT (minRTT) of packet transmission [4–6]. It controls the pacing_gain and cwnd_gain parameters to increase the packet transmission rate and expand the congestion window, allowing more packets to be transmitted. BBRv1 permits a 3% packet loss rate to probe for the maximum bandwidth. The transmission bandwidth size is determined based on the ratio of the total transmission to delay to reduce buffer congestion and increase the transmission speed. This approach enables better adaptation to various network environments.

BBRv2 optimizes the bandwidth-hogging in multiple connections to achieve higher fairness and reduce transmission delay [7]. Key additions include the calculation of the packet loss rate and the incorporation of ECN in congestion determination. It allows for a packet loss of 3% and an ECN of 50% to minimize the accumulation in the buffer queue. BBRv2 introduces four stages to ProbeBw and adds limits on the minimum (inflight_lo) and maximum (inflight_hi) number of packets transmitted. These changes reduce bandwidth-hogging issues when interacting with other congestion-controlled connections, thus improving bandwidth sharing. The modifications address the problem of slow convergence to optimal fairness in larger buffer zones. However, while BBRv2 makes significant improvements, it does not fully realize optimized fair sharing in all scenarios.

Based on the result of Ref. [8], BBRv3 makes the following adjustments. The packet sending rate gain in the StartUP stage is changed from 2.89 to 2.77, the packet loss threshold is modified, and the ECN threshold parameter is adjusted. Additionally, the congestion window gain is altered from 2.89 to 2.0, along with parameter settings for bandwidth stabilization. Upon exiting the StartUP stage, the inflight_hi parameter is set based on the greater value between the estimated Bandwidth-Delay Product (*BDP*) and the maximum number of packets successfully transmitted in the last RTT. The criteria for exiting the StartUP stage due to packet loss have also been revised.

3. Improved BBR

3.1. Congestion Control Based on BBR

Congestion control relies on actively detecting the bottleneck bandwidth and estimating the round-trip propagation delay. The first version of BBR uses four stages to adjust congestion control. There are two important gain parameters, pacing_gain and cwnd_gain. Pacing_gain is used to control the delivery rate, while cwnd_gain manages the number of packets being transmitted in flight. When the StartUP stage begins detecting the bottleneck bandwidth and estimating the round-trip propagation delay, BBR sets both pacing_gain and cwnd_gain to $2/ln\ 2$, which is approximately 2.89. BBR considers the maximum transmission rate in the last 10 RTTs as the bottleneck bandwidth (*BtlBw*) and the minimum latency measured within the last 10 s as the round-trip propagation time (*RTprop*). BBR controls its transmission behavior using pacing_gain and cwnd_gain, exponentially increasing the sending rate to fully utilize the bottleneck link. If the sending rate is not increased significantly (less than 25% change within three RTTs) after three sending attempts, the actual sending rate reaches *BtlBw*. The bottleneck bandwidth is defined as shown in (2):

$$BtlBw = \max\{dt\}\ \forall t \in [T - WB\ ,\ T] \tag{2}$$

where d_t is the delivery rate, W_B is 6 to 10 RTTs, and T is the transmission time during probing. The final result obtained is the BDP as the baseline for determining the congestion control. It is assessed whether congestion is present. The congestion control operation is expressed as shown in (3):

$$BDP = BtlBw\ \times RTprop \tag{3}$$

where $RTprop$ is a physical characteristic of the connection.

The approximate value is obtained indirectly by monitoring the RTT of the packets, taking the maximum transmission among 10 RTTs as the criterion. After confirming the congestion control point, in the Drain stage, pacing_gain is set to the reciprocal of $ln\ 2/2$ from the StartUP stage since buffering occurs during detection and there is also the problem of packet loss. This clears the queue that appears during reprobing. When the number of inflight packets matches the BDP, it is determined if the queue is emptied, allowing BBR to leave the Drain stage and enter the ProbeBW stage.

In the ProbeBW stage, BBR remains in this stage for an extended period. The value of cycle_gain = pacing_gain is used to detect the bottleneck bandwidth and adjust the sending rate with values of 1.25, 0.75, and 1 to achieve continuous detection. To detect more bandwidth, the sending rate is increased by 1/4. If $RTprop$ is not updated for more than 10 s, BBR enters the ProbeRTT stage. In this stage, cwnd is reduced to four packets and maintained for 200 ms or one RTT. If the time exceeds this value, the network channel becomes full.

The key feature of BBRv2 is the enhancement of ProbeBW and the addition of four stages to adjust congestion control under steady-state conditions, as shown in Figure 2. Different pacing_gain values are used in various sub-stages of ProbeBW, and the bandwidth detection time is adaptive. This improves the coexistence of BBR and CUBIC. Additionally, an ECN threshold greater than 50% is used to alert the receiving end that congestion is imminent. A 3% packet loss rate is used to determine whether to exit the StartUP stage early and inflight_hi.

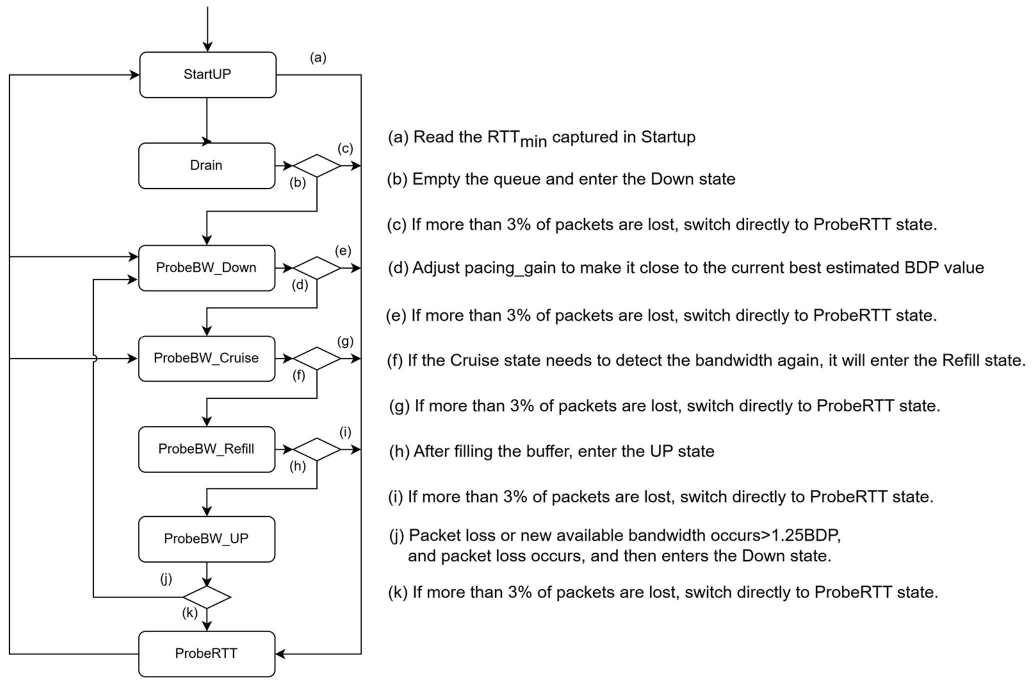
StartUP
Drain
ProbeBW_Down
ProbeBW_Cruise
ProbeBW_Refill
ProbeBW_UP
ProbeRTT
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)
(k)
(a) Read the RTT$_{min}$ captured in Startup
(b) Empty the queue and enter the Down state
(c) If more than 3% of packets are lost, switch directly to ProbeRTT state.
(d) Adjust pacing_gain to make it close to the current best estimated BDP value
(e) If more than 3% of packets are lost, switch directly to ProbeRTT state.
(f) If the Cruise state needs to detect the bandwidth again, it will enter the Refill state.
(g) If more than 3% of packets are lost, switch directly to ProbeRTT state.
(h) After filling the buffer, enter the UP state
(i) If more than 3% of packets are lost, switch directly to ProbeRTT state.
(j) Packet loss or new available bandwidth occurs>1.25BDP, and packet loss occurs, and then enters the Down state.
(k) If more than 3% of packets are lost, switch directly to ProbeRTT state.

Figure 2. BBRv2 ProbeBW and ProbeRTT detailed status diagram.

The down stage is illustrated in Figure 2(b,c). In the Cruise stage, several bandwidths are reserved for other connections, with inflight kept between inflight_lo and inflight_hi (Figure 2(f,g)). In the Refill stage, the sending rate gradually increases to approach the upper limit of the estimated bandwidth (Figure 2(h,i)).

In the ProbeBW_UP stage, the sending rate is accelerated until it reaches the upper limit of the estimated bandwidth to maintain steady-stage behavior. If packet loss occurs, the status transitions to the ProbeRTT stage (Figure 2(j,k)). This allows BBRv2 to maximize the use of the available bandwidth. If an ECN mark is detected or a packet is lost, the sending rate is reduced by 30%. BBRv2 uses an exponential growth method, starting slowly and then accelerating, to detect the available bandwidth. The inflight_hi value is not updated until packet loss occurs or the newly available bandwidth is greater than 1.25 times the estimated bandwidth, coinciding with packet loss.

In the adjustments mentioned, although BBRv2 sets inflight_hi and inflight_lo, the detection bandwidth cannot be effectively increased due to the influence of inflight_hi when striving for bandwidth. With a larger buffer, CUBIC detects the bandwidth based on packet loss, but BBRv2 sets inflight_hi to limit the maximum detection bandwidth. As a result, BBRv2 deviates further from fairness and cannot converge back.

BBRv3 [8,9] adjusts cwnd_gain to 2.0 during the StartUP stage and also modifies pacing_gain from 2.89 to 2.77. When exiting the StartUP stage, BBRv3 sets the inflight_hi parameter based on the maximum value between the estimated BDP and the highest number of successfully transmitted packets during the last RTT. Additionally, the criteria for exiting the StartUP stage due to packet loss are updated to reduce queuing delays during both the StartUP stage and subsequent stages, thereby decreasing the packet loss rate. Bandwidth detection continues until the packet loss rate or ECN exceeds the predefined 1% threshold. Detection stops if no packets are transmitted and the available bandwidth reaches the maximum inflight_hi.

While CUBIC congestion control offers excellent fairness, it falls short compared to the various versions of BBR in terms of bandwidth utilization. Each version of BBR actively detects and increases the bandwidth, but this aggressive approach leads to significantly poorer fairness compared to CUBIC. Although BBRv2 improves fairness compared to

BBRv1 by setting inflight_hi to limit the maximum transmission bandwidth, it does not achieve optimal fairness in larger buffers. In BBRv3, pacing_gain and cwnd_gain are reduced in the StartUP stage, and bandwidth detection is stopped if inflight_hi is not reached. This approach balances fairness and bandwidth utilization better, as illustrated in Figure 3.

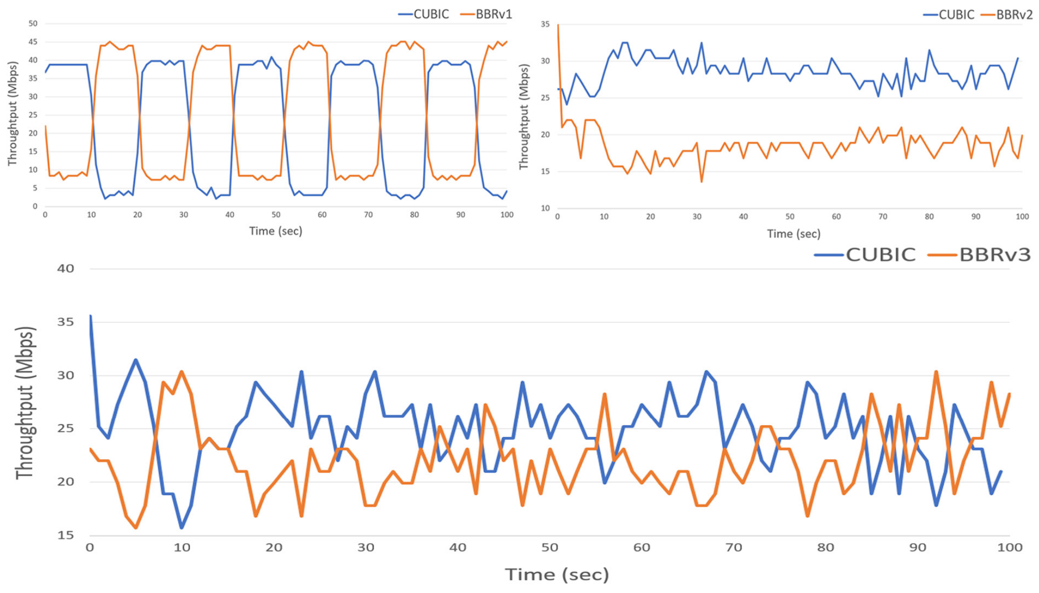

Figure 3. Comparison of throughput between CUBIC and BBR congestion control.

3.2. Fair Congestion Control Mechanism

BBRv3 and CUBIC are used in this study because CUBIC is based on packet loss and offers superior bandwidth detection and fairness compared to other packet loss-based congestion controls. To obtain better bandwidth fairness, we adjust pacing_gain to observe its impact on fairness with the generally low bandwidth of BBRv3. In the StartUP stage, BBRv3 sets pacing_gain to $4ln(2)$ for bandwidth detection. Increasing pacing_gain to $2/ln(2)$ significantly reduces fairness. The proposed pacing_gain setting in the StartUP stage is shown in Algorithm 1.

Algorithm 1: Pacing_gain setting in StartUP stage

```
Set $\alpha = 4$, $\beta = 2$, $\gamma = 1$    //Initial Setup for BBRv3
Flag = False
$Pacing\_gain = \alpha \times (\ln \beta)^{\gamma}$
Do {
Increasing/decreasing Pacing_gain with different $\alpha$, $\beta$, $\gamma$
If Fairness of network increase and it closed to 1 then
Store $\alpha$, $\beta$ and $\gamma$
Accepted the Paceing_gain with $\alpha$, $\beta$ and $\gamma$Flag = True
End if
} While (Flag == True)
Return Paceing_gain
```

4. Results

The network setup includes four virtual machines (two sending nodes and two receiving nodes) and two switches (SW1 and SW2), connected through bottleneck links for congestion control experiments (Figure 4). The GNS3 network simulator is used for the experiment. The bottleneck bandwidth is 50 Mbps, and the transmitter sending rate is 150 Mbps. Additionally, iperf3 is used for transmitting packets.

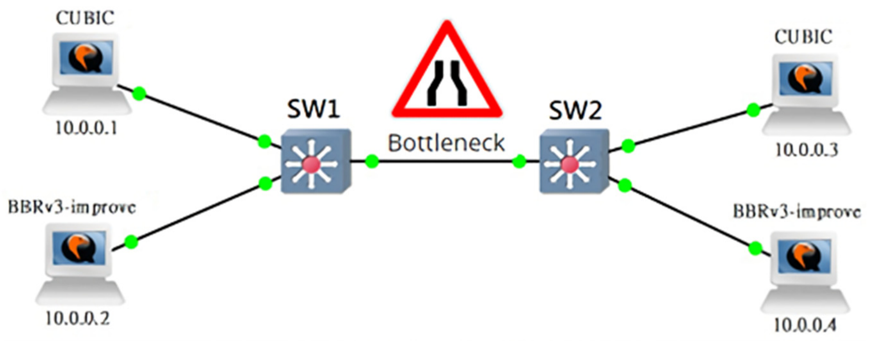

Figure 4. Network topology diagram.

In the first stage of the experiment, Flow1 and Flow2 start transmitting at the same time. The pacing_gain is set to $3ln2$, approximately 2.07, in the StartUP stage to obtain better bandwidth fairness. As shown in Figure 5, when there are two flows, Flow1 uses CUBIC and Flow2 uses BBRv3+ simultaneously in stage 1, and the bandwidth of the two flows is close to 25 Mbps with similar performance (Figure 4).

In the second stage of the experiment, Flow1 begins transmission at the 0 s mark, while Flow2 commences transmission 2 s later. The pacing_gain is set to $4ln2$, using BBRv3 in the StartUP stage of Flow2. The throughput of the two flows is shown in Figure 6. At the 0 s mark, only Flow1 uses the connection, fully utilizing the bandwidth. When Flow2 begins with the 2 s delay, bandwidth competition starts. An obvious fairness issue is visible at the 40 s mark. By the 100 s mark, Flow2 utilizes the entire bandwidth (Figure 6).

Similarly, the proposed BBRv3+ is implemented in Flow2, which began transmission two seconds after Flow1 started. At the 0 s mark, Flow1 exclusively utilizes the connection, maximizing bandwidth usage. The competition for bandwidth begins at the 2 s mark when Flow2 initiates. The proposed congestion control mechanism enables improved bandwidth fairness between the flows. As Flow1 concludes its transmission at the 100 s mark, Flow2 subsequently occupies the entire bandwidth (Figure 7).

The experimental results are shown in Table 1. With a bandwidth of 50 Mbps, each flow is allocated 25 Mbps if the bandwidth is fairly distributed between the two flows. BBRv3+ produces an output of nearly 25 Mbps, regardless of whether the transmissions start simultaneously or with a 2 s delay. This demonstrates that BBRv3+ shows a better output and fairness performance compared to the other methods.

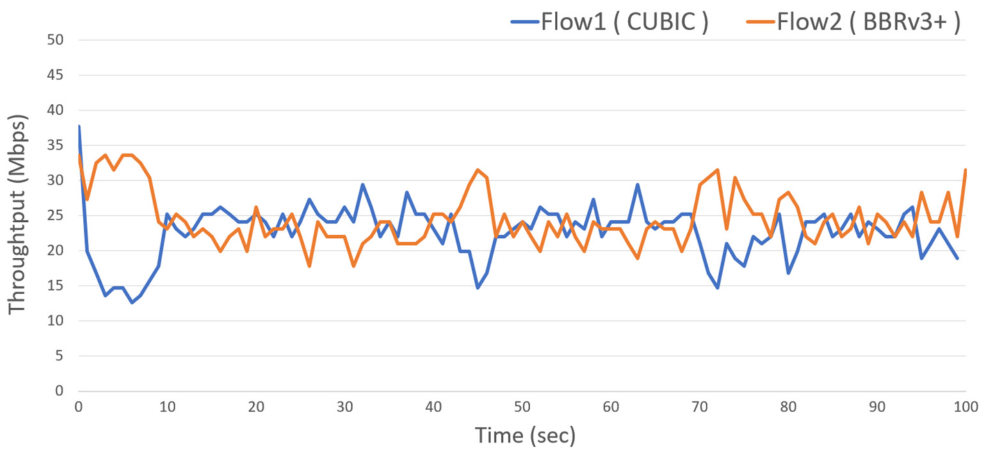

Figure 5. Comparison of throughput between CUBIC and BBRv3+ (1st stage).

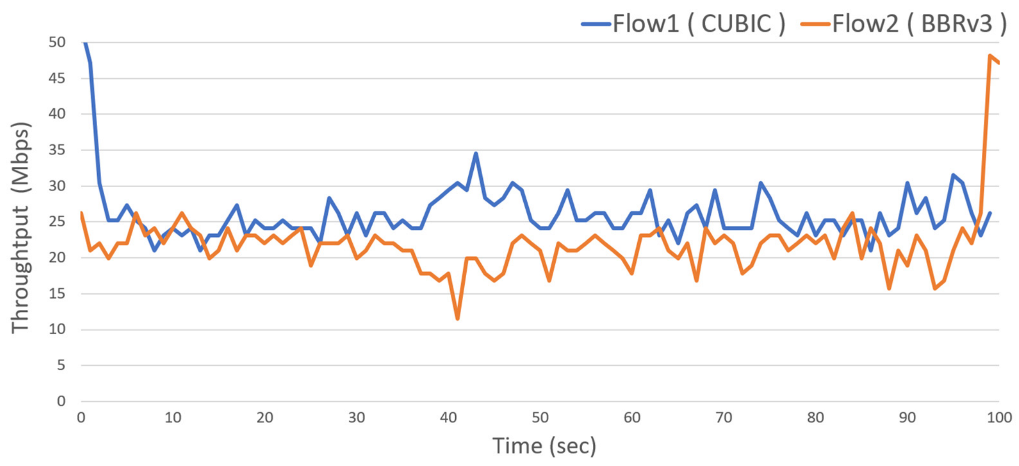

Figure 6. Comparison of throughput between CUBIC and BBRv3 (2nd stage).

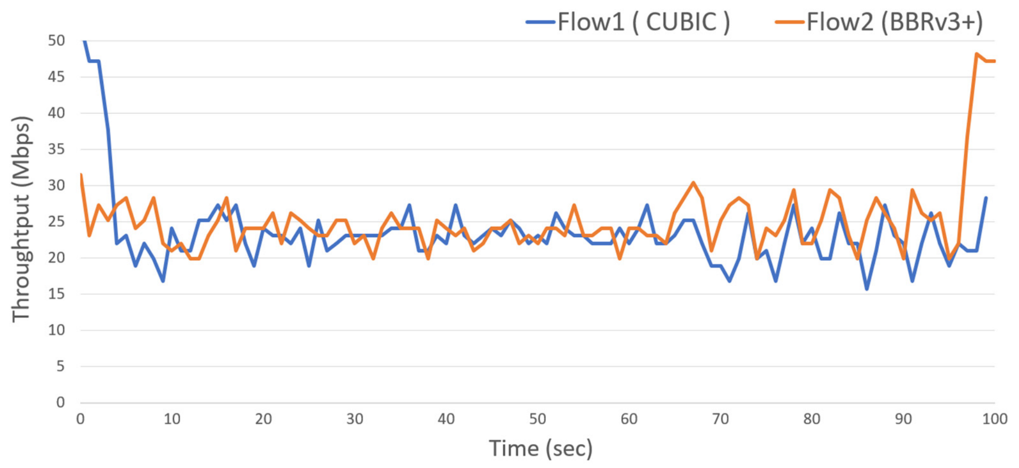

Figure 7. Comparison of throughput between CUBIC and BBRv3+ (2nd stage).

Table 1. Throughput with different congestion controls.

Flow2 \ Throughput		Flow1: CUBIC			Flow2		
		Max.	**Min.**	**Avg.**	**Max.**	**Min.**	**Avg.**
1st Stage	BBRv1	40.9	2.1	23.0	45.1	7.34	24.3
	BBRv2	32.5	24.1	28.5	35.6	13.6	18.6
	BBRv3	35.6	15.7	24.9	30.4	15.7	22.1
	BBRv3+	37.7	12.6	22.7	33.6	17.8	24.5
2nd Stage	BBRv3	52.4	21	26.1	48.2	11.5	21.9
	BBRv3+	52.4	15.7	23.6	48.2	19.9	25.1

5. Conclusions

BBRv3+, the improved congestion control algorithm addresses the unfair bandwidth distribution of BBRv3. When connections are established at different times, fairness in the early and mid-term stages is significantly improved, surpassing the performance of previous BBRs. However, unfair situations persist compared to CUBIC. Therefore, it is required to improve fairness in the early and middle stages of connections by incorporating adaptive adjustments to sending rates and congestion windows across various congestion control scenarios.

Author Contributions: Conceptualization, H.-C.C. (Hung-Chi Chu) and H.-C.C. (Hao-Chu Chiang); methodology, H.-C.C. (Hao-Chu Chiang); software, H.-C.C. (Hao-Chu Chiang); validation, H.-C.C.

(Hung-Chi Chu) and H.-C.C. (Hao-Chu Chiang); formal analysis, H.-C.C. (Hung-Chi Chu) and H.-C.C. (Hao-Chu Chiang); investigation, H.-C.C. (Hung-Chi Chu) and H.-C.C. (Hao-Chu Chiang); resources, H.-C.C. (Hung-Chi Chu) and H.-C.C. (Hao-Chu Chiang); data curation, H.-C.C. (Hao-Chu Chiang); writing—original draft, H.-C.C. (Hung-Chi Chu) and H.-C.C. (Hao-Chu Chiang); writing—review and editing, H.-C.C. (Hao-Chu Chiang); visualization, H.-C.C. (Hung-Chi Chu) and H.-C.C. (Hao-Chu Chiang); supervision, H.-C.C. (Hung-Chi Chu). All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology, Taiwan, under grant NSTC 113-2221-E-324-015.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used in the current study are available from the corresponding author upon request.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of the data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Cerf, V.; Kahn, R. A protocol for packet network intercommunication. *IEEE Trans. Commun.* **1974**, *22*, 637–648. [CrossRef]
2. Rose, K. *Computer Network a Top-Down Approach*, 7th ed.; Pearson: London, UK, 2018.
3. Ha, S.; Rhee, I.; Xu, L. Cubic: A new tcp-friendly high-speed tcp variant. *ACM SIGOPS Oper. Syst. Rev.* **2008**, *42*, 64–74. [CrossRef]
4. Cardwell, N.; Cheng, Y.; Gunn, C.S.; Yeganeh, S.H.; Jacobson, V. BBR: Congestion-based congestion control. *Commun. ACM* **2017**, *60*, 58–66. [CrossRef]
5. Cardwell, N.; Cheng, Y.; Gunn, C.S.; Yeganeh, S.H.; Jacobson, V. BBR Congestion Control. IETF Draft draft-cardwell-iccrg-bbr-congestion-control-00. 2017. Available online: https://datatracker.ietf.org/meeting/101/materials/slides-101-iccrg-an-update-on-bbr-work-at-google-00 (accessed on 15 May 2024).
6. BBR-Congestion Control (draft-cardwell-iccrg-bbr-congestion-control-02). 8 September 2022. Available online: https://datatracker.ietf.org/doc/html/draft-cardwell-iccrg-bbr-congestion-control (accessed on 25 June 2024).
7. Cardwell, N.; Cheng, Y.; Yeganeh, S.H.; Jha, P.; Seung, Y.; Yang, K.; Jacobson, V. BBRv2: A model-based congestion control performance optimization. In Proceedings of the IETF 106th Meeting, Singapore, 16–22 November 2019; pp. 1–32.
8. Cardwell, N.; Cheng, Y.; Yang, K.; David, M.; Yeganeh, S.H.; Jha, P.; Seung, Y.; Jacobson, V.; Swett, I.; Wu, B.; et al. BBRv3: Algorithm Bug Fixes and Public Internet Deployment. In Proceedings of the IETF 117th Meeting, San Francisco, CA, USA, 22–28 July 2023; pp. 1–36.
9. Gomez, J.; Kfoury, E.F.; Crichigno, J.; Srivastava, G. Evaluating TCP BBRv3 performance in wired broadband networks. *Comput. Commun.* **2024**, *222*, 198–208. [CrossRef]